ИНФОРМАТИКА

ВАРИАНТ 1

Задача 1

Определите названия ячеек в блоке A1:E1, используя подсказки.

	A	B	C	D	E
1					
2					
3	=ЗНАЧЕН(ПРАВСИМВ(A1;1))	=ЗНАЧЕН(ПРАВСИМВ(B1;1))	=ЗНАЧЕН(ПРАВСИМВ(C1;1))	=ЗНАЧЕН(ПРАВСИМВ(D1;1))	=ЗНАЧЕН(ПРАВСИМВ(E1;1))
4	=ЛЕВСИМВ(A1;1)	=ЛЕВСИМВ(B1;1)	=ЛЕВСИМВ(C1;1)	=ЛЕВСИМВ(D1;1)	=ЛЕВСИМВ(E1;1)
5					

	A	B	C
6	=A3-EXP(0)		=ПОИСК(A4;"bcadfe")
7	=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)		=A4=C4
8	=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3		=КОДСИМВ(B4)-КОДСИМВ(C4)
9	=C3-D3-E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))
10	=C3+D3*E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))

	A	B	C
6	0		3
7	8		ИСТИНА
8	7		1
9	-2		5
10	13		6

Решение:

1. В соответствии с формулами «=ЗНАЧЕН(ПРАВСИМВ(A1;1))», «=ЗНАЧЕН(ПРАВСИМВ(B1;1))» и т.д. в ячейках A3:E3 находятся цифры обозначающие строки определяемых ячеек. В ячейках A4:E4 с формулами «=ЛЕВСИМВ(A1;1)», «=ЛЕВСИМВ(B1;1)» и т.д. соответственно, буквы латинского алфавита, обозначающие колонки определяемых ячеек.

2. Так как значение формулы «=A3-EXP(0)» в ячейке A6 равно 0, то A3 = 0 +EXP(0) = 1.

3. Значение формулы в A7 «=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)» равно 8, поэтому B3 = 8-A3-ОКРУГЛВВЕРХ(ПИ();0) = 8-1-4 = 3.

4. Значение формулы в A8 «=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3» равно 7, поэтому C3 = 7-B3+ ОКРУГЛВНИЗ(EXP(1);0) = 7-3+2 = 6.

5. Значение формулы в A9 «=C3-D3-E3» равно -2, а значение формулы в A10 «=C3+D3*E3» равно 13. Получаем систему уравнений: $\begin{cases} 6 - D3 - E3 = -2 \\ 6 + D3 * E3 = 13 \end{cases}$ $\Rightarrow \begin{cases} -D3 - E3 = -8 \\ D3 * E3 = 7 \end{cases}$, решая которую получим D3 = 7, E3 = 1.

6. Значение формулы в C6 «=ПОИСК(A4;"bcadfe")» равно 3, следовательно в A4 находится символ A.

7. Значение формулы в C7 «=A4=C4» равно «ИСТИНА», следовательно в C4 находится также символ A.

8. Значение формулы в C8 «=КОДСИМВ(B4)-КОДСИМВ(C4)» равно 1, следовательно в B4 находится символ следующий по алфавиту за символом A, а это - B.

9. Значение формулы в C9 «=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 5, а значение формулы в C10 «=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 6. Получаем систему уравнений относительно смещения от начала алфавита символа в ячейке D4 и в ячейке E4. Если обозначим их, соответственно, как Dи E, тополучим: $\begin{cases} E + D = 5 \\ E * D = 6 \end{cases}$. Решая систему уравнений,определяемD = 2, E = 3. Следовательно в ячейке D4 находится символ C, а в ячейке E4 – символ D.

Ответ:

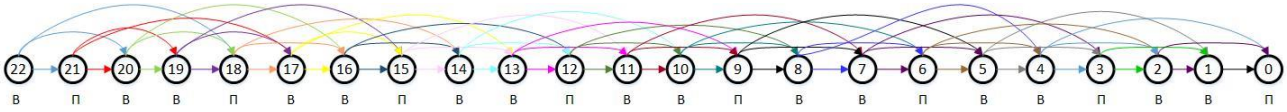
	A	B	C	D	E
1	A1	B3	A6	C7	D1

Задача 2 (10 баллов)

На столе лежит 22 драгоценных камня. Гномы, участвующие в игре, по очереди могут взять 1, 2 или 4 камня. Гном, который не может сделать ход (камней не осталось), – проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Какова должна быть стратегия, выигравшего гнома? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней стратегии выигрывающего гнома. С каким количеством камней этот алгоритм будет работать?

Решение:

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Делаем вывод, что с периодом 3: позиции, где число камней делится на 3 без остатка, будут проигрышными (для того, кто в них оказался), а где не делится — выигрышными. В нашем случае, когда в игре 22 камня, выигрывает первый игрок. Для

безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось кратное трём количество.
Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней (в данном случае игрок 1 может взять 4 камня, чтобы осталось 18 или 1 камень, чтобы осталось 21).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет от 1,2 или 4 камня (в данном случае камней остаётся 17, 16 или 14).

Ход_3: Игрок_1 берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней. И так далее до Победа Игрок_1. Проиграть невозможно.

Алгоритм для Игрок_1 (обобщенный):

Шаг 1 Игрок _1 взял К=1,2 или 4 камня, так чтобы (N-К) кратно 3
Если Остаток = 0 – значит СТОП_Выиграл.

Шаг 2 Игрок_2 взял К=1,2 или 4 камня
Алгоритм подходит в общем случае, для N камней, если N делится на 3 без остатка.

Задача 3 (10 баллов)

Дан фрагмент таблицы истинности и результирующий столбец F(y,x)?G(x,y,z). Укажите пропущенные значения, а также какие переменные в каком порядке указаны в столбцах, а также какой оператор должен стоять между функциями в результирующем столбце.

?	?	?	$\bar{y} \vee z$	$x \to z$	$\bar{y} \wedge x$	F(y, z)?F(x,z)	F(y,x)?G(x,y,z)
0		0	1	1	0	1	0
0		1	1	1	0	1	0
	1		1	0	1	0	0
	1		1	1	1	1	1
1	0	0	0	1	0	0	0

1		1	1	1	0	1	0
		0	0	0	0	0	0
	1	1	1	1	0	1	0

Решение.

$\bar{y} \vee z$ - это F(y,z)

$x \to z$ - это F(x,z)

$\bar{y} \wedge x$ - Это F(y,x)

G(x,y,z) – это F(y,z)&F(x,z)

y	x	z	$\bar{y} \vee z$	$x \to z$	$\bar{y} \wedge x$	F(y, z)**&**F(x,z)	F(y,x)**&**G(x,y,z)
0	0	0	1	1	0	1	0
0	0	1	1	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0
1	0	1	1	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	1	1	0	1	0

Задача 4 (20 баллов)

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

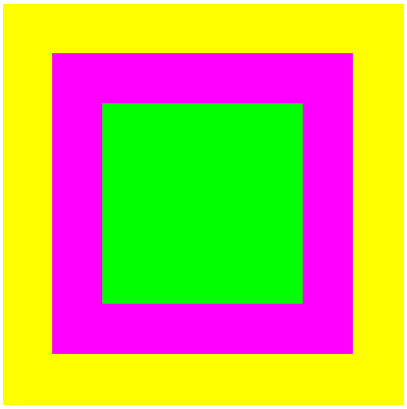


Рисунок 1

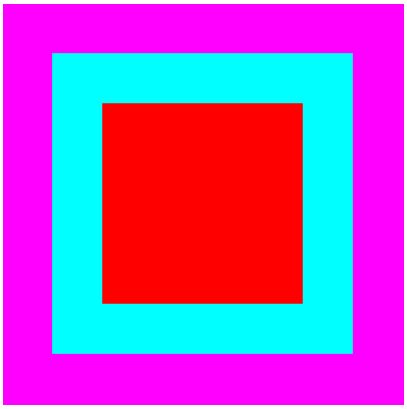


Рисунок 2

Решение:

Как мы видим, в обработанном изображении зелёный цвет стал синим. Делаем предположение, что произошла смена каналов. Значение яркости зелёной составляющей было записано в красный канал.

Жёлтый цвет стал лиловым. Желтый цвет состоит из максимальных значений красной и зеленой составляющих. Так как мы уже знаем, что значение зелёной составляющей было записано в красный канал, делаем вывод, что значение красной составляющей было записано в синий канал. Таким образом был получен лиловый цвет.

Значит, значение синей составляющей было записано в зелёный канал, что и подтверждает смена лилового цвета на голубой.

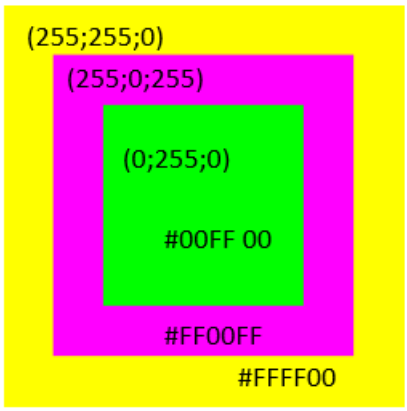
Алгоритм обработки выглядит следующим образом:

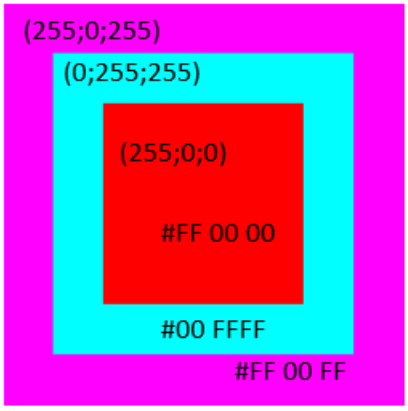
$$Y_R = Y_G$$

$$Y_G = Y_B$$

$$Y_B = Y_R$$

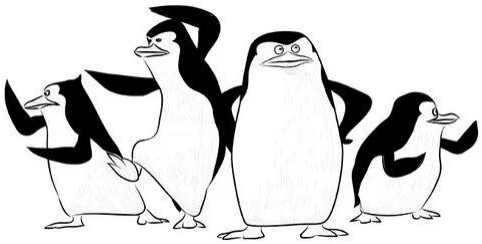
Далее укажем на рисунках значения яркости в RGB и HEX формате





Задача 5 (25 баллов)

ПИНГВИГРА



Однажды гуляя по антарктическим просторам, группа переворачивателей пингвинов (Не удивляйтесь, есть и такая профессия) наткнулась на одно семейство. В течение нескольких дней они наблюдали по очереди за ними. И заметили, необычное поведение пингвинов. Было похоже, что они играют в какую-то игру чтобы не замерзнуть или просто весело провести время.

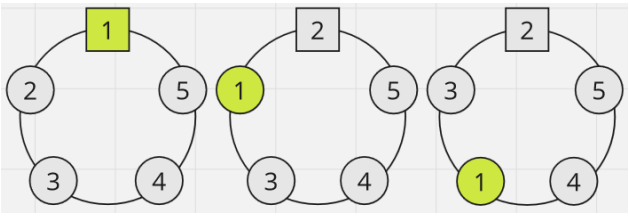
Пингвины вставали в круг лицом к центру рядом с одним из них всегда был небольшой снежный ком. Начиналась игра после того, как пингвин у камня подпрыгивал и похлопывал крыльями, после этого он менялся местами с соседом справа сколько-то раз. Когда первый пингвин перешел в новую позицию, другой пингвин, который оказался у снежного кома также подпрыгивал и тоже менялся с соседом справа, причем количество смен было больше, чем у предыдущего пингвина. Через некоторое количество таких передвижений, вероятно это был конец игры, пингвины разбегались и два которые были рядом с первым, приносили ему рыбку.

Переворачивателей заинтересовал этот необычный обряд у пингвинов, и продолжая наблюдать, они заметили, что количество перемещений совпадают с последовательностью простых чисел. Разумеется, они захотели попробовать эту игру. Для этого каждый из них присвоил себе цифру от 1 до *N*. Перед началом игры они нарисовали *N – 1*кругов и один квадрат (обозначающий снежный ком, как у пингвинов) в большом кругу. Переворачиватель с номером 1 встает в квадрат. Все остальные встают по порядку цифр, начиная со двойки, против часовой стрелки, лицом к центру.

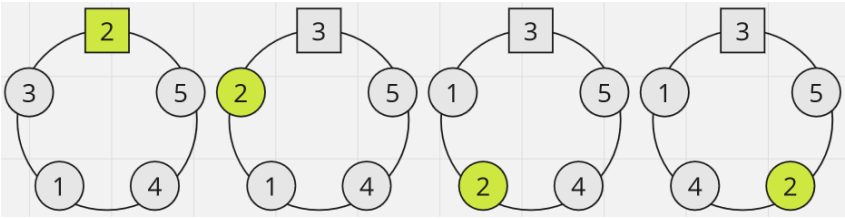
Они условились, что игра будет состоять из *M* раундов. В *i*-м раунде человек, который находиться в квадрате, подпрыгивает и кричит «Я пингвин» и затем меняется с человеком справа от него p_kраз, где p_k–простое число.

Например, для N = 5 и M = 3 происходят следующие три раунда:

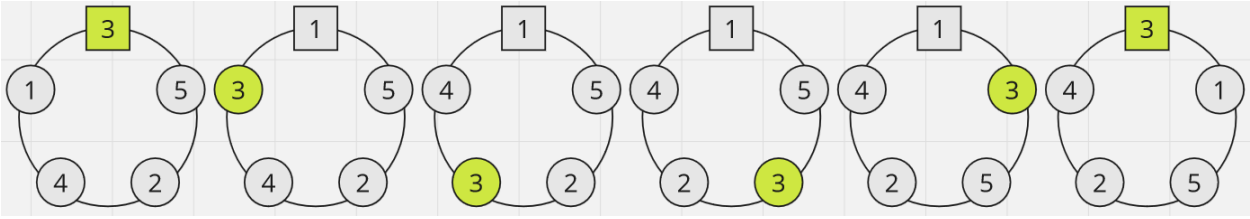
<u>1 - раунд</u>



2 - раунд



3 - раунд



Напишите программу, которая для заданных N, M и G определяет соседей человека под номером G в конце игры.

Входные данные: вам дается три числачерез пробел N, M, G, где N–количество участников, M–количество раундов и G– номер человека.

Результат должен содержатьномера двух соседей человека с номером G, сначала правого, потом левого.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод: 5 3 1	Результат: 3 5
Ввод: 5 3 2	Результат: 5 4
Ввод: 5 4 5	Результат: 3 2

Исходные данные:

9 5 3

Решение:

Мы используем решето Эратосфена, чтобы найти первые K простых чисел.

Теперь обозначим квадрат цифрой 0, а круги - числами от 1 до N - 1. Поскольку невозможно смоделировать всю игру за отведенное время, мы сначала определяем алгоритм, который поможет нам найти конечную позицию G. Предположим, что в данный момент G находится на позиции **X**, а текущеепростое число - p:

- Если **X = 0**, то **G** находится в квадрате, поэтому **X** становится **N**mod**p**.
- Иначе, если **X ≠ 0**, то мы можем определить, сколько раз человек в квадрате меняется местами с **X**:
  - Если X<= p mod (N-1), то X становится (X - 1) mod N
  - Иначе X становится (X - p div (N-1)) mod N

где mod - оператор деления по модулю, a div - целочисленное деление.

Теперь нам нужно решить другую задачу чтобы ответить на вопрос "Если G находится на X, то кто находится на X-1 (или X+1)?".

Предположим, что нам известна позиция X**после** того, как был сыгран простое число p.

- Если X = p mod N, то Xбыл 0 перед простым числом p
- Иначе, если x ≠ p mod N:
  - x становится (x + p div (N-1)) mod N
  - Если x <= p mod (N-1), то x становится (x + 1) mod N

РеализациянаC++

```
#define_CRT_SECURE_NO_WARNINGS

#include<cstdio>
```

```
#defineMAXSIEVE 7380000
#defineMAXPRIMES 501000
int p[MAXPRIMES], np;
charsieve[MAXSIEVE];

int N;
intmod(inta) { return (a % N + N) % N; }

intforward(intx, intp) {
if (x == 0) return mod(p);

if (x<= p % (N - 1)) x = mod(x - 1);
x = mod(x - p / (N - 1));

returnx;
}

intbackward(intx, intp) {
if (x == mod(p)) return 0;

x = mod(x + p / (N - 1));
if (x<= p % (N - 1)) x = mod(x + 1);

returnx;
}

int main(void) {
for (inti = 2; i<MAXSIEVE; ++i) {
if (sieve[i]) continue;
      p[np++] = i;
for (int j = i + i; j <MAXSIEVE; j += i) sieve[j] = 1;
  }
```

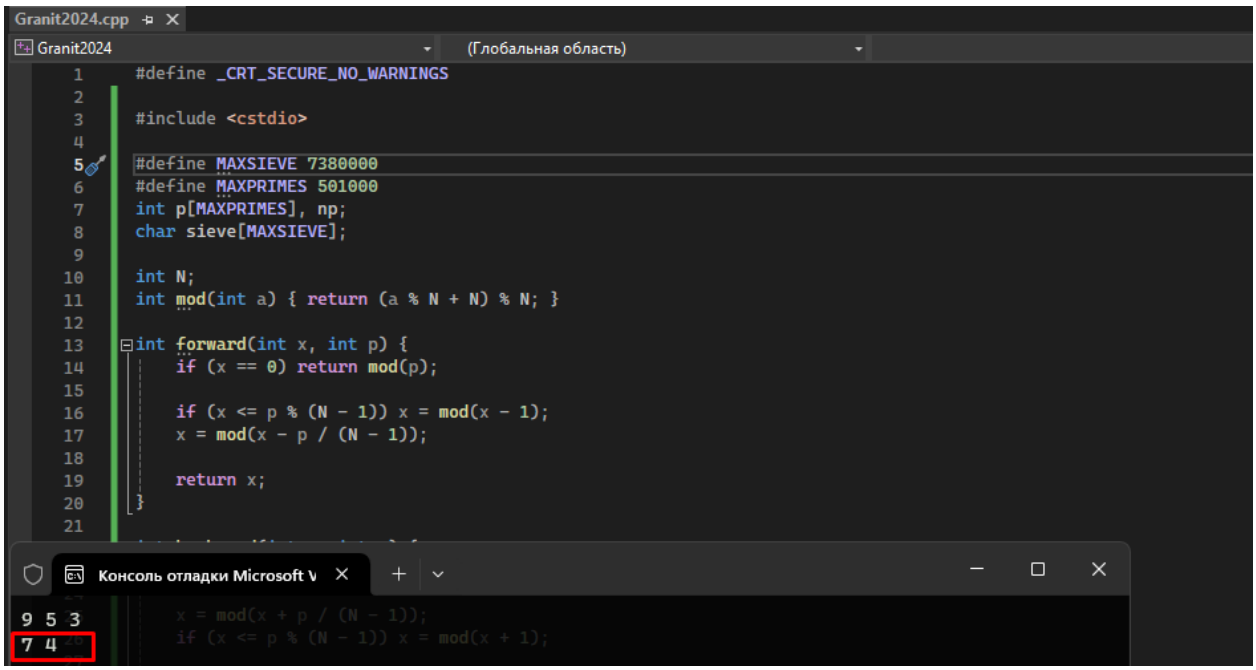
```
int A, K;
scanf("%d%d%d", &N, &K, &A); --A;
for (inti = 0; i< K; ++i) A = forward(A, p[i]);

int L = mod(A - 1), R = mod(A + 1);

for (int i = K - 1; i>= 0; --i) L = backward(L, p[i]);
for (int i = K - 1; i>= 0; --i) R = backward(R, p[i]);

printf("%d %d\n", R + 1, L + 1);
return 0;
}
```

Реализация на Python

```
MAXSIEVE = 7380000
MAXPRIMES = 501000
p = []
np = 0
sieve = [False] * MAXSIEVE

def mod(a):
  return (a % N + N) % N

defforward(x, p):
  if x == 0:
    return mod(p)
  if x <= p % (N - 1):
    x = mod(x - 1)
  x = mod(x - p // (N - 1))
  return x

defbackward(x, p):
  if x == mod(p):
```

```
        return0
    x = mod(x + p // (N - 1))
    if x <= p % (N - 1):
        x = mod(x + 1)
    return x

foriinrange(2, MAXSIEVE):
    if sieve[i]:
        continue
    p.append(i)
    np += 1
    for j inrange(i + i, MAXSIEVE, i):
        sieve[j] = True

N, K, A = map(int, input().split())
A -= 1
foriin range(K):
    A = forward(A, p[i])

L = mod(A - 1)
R = mod(A + 1)

foriinrange(K - 1, -1, -1):
    L = backward(L, p[i])
foriinrange(K - 1, -1, -1):
    R = backward(R, p[i])

print(R + 1, L + 1)
```

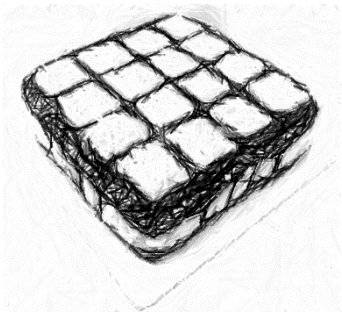
Демонстрация на C++



Ответ: 7 4

Задача 6 (25 баллов)

ОПЯТЬ ТОРТ

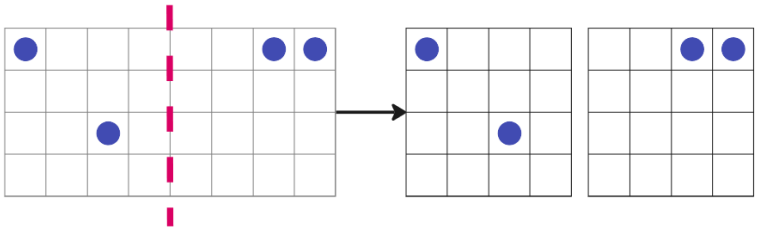


На день рождения Кати друзья испекли прямоугольный торт, украшенный К свечами – разумеется столько же сколько исполнилось лет Кате.Поскольку торт напечатан в клеточку, мы можем представить его как прямоугольник M × N.В некоторых ячейках расположена одна свеча, а в других нет.

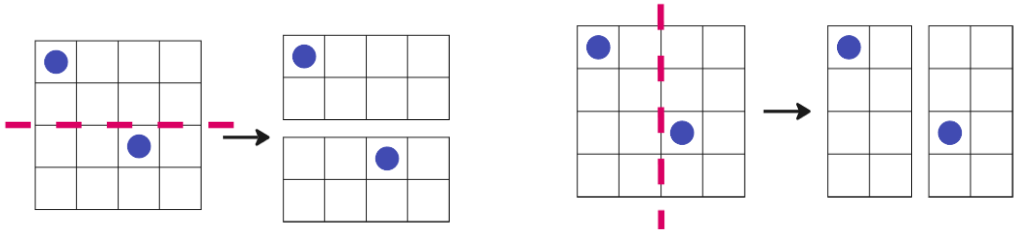
Лучшие друзья дали Кате задание отрезать кусок торта по определенным правилам:

- Горизонтальным или вертикальным разрезом, проходящим через края ячеек, торт разделяется на две прямоугольные части.Полученные кусочки должны быть одинакового размера и иметь одинаковое количество свечей.
- Далее Катя откладывает один из кусков в сторону, а остальные продолжает разрезать по тем же правилам.
- Когда остается кусок, который больше разрезать нельзя, т.е. в этом куске ровно одна свеча, он достается Кате.В противном случае Катя не получит торт.

Например, торт размером 4×8 можно разделить по вертикали на две части размером 4×4 по две свечи в каждой.Первый разрез не может быть горизонтальным, потому что, если вы разрежете середину, в верхней части будет три свечи, а в нижней - только одна.



По условию правый кусок не может быть разрезан дальше и имеет две свечи.Если бы Катяпродолжила разрезать его, она бы не получила торт.Левую часть можно разрезать как горизонтально, так и вертикально.



Разрезание в обе стороны дает по одной свече в каждой, поэтому любая из них может достаться к Кате. В этом примере Катя может получить одну из четырех разных частей. Напишите программу, которая считает, сколько разных фигур сможет получить Катя.Кусочки считаются разными, если они занимают в торте разное положение.

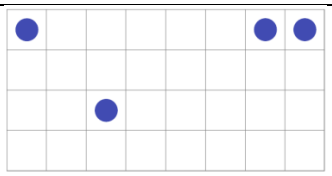
Входные данные:В первой строке записаны три целых числа: высота торта M, ширина N и количество свечей K. Следующие K строк содержат координаты кусочков со свечами.Первая координата вертикальная (нумеруется от 0 до M – 1 сверху вниз), а вторая горизонтальная (нумеруется от 0 до N – 1 слева направо).

Результат должен содержать одно число, обозначающее, сколько разных фигур может достатьсяКате.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример входных данных

Ввод:	Результат:	Пример описан в условии.
4 8 4 0 0 2 2	4	

0 6 0 7		

Исходные данные:

4 4 3

0 0

1 2

2 2

Решение:

Мы храним свечи в списке и делим торт рекурсивно. На каждом шаге мы также разбиваем список свечей на два списка — по одному на каждую из новых фигур.

Важная оптимизация — если мы доходим до части, которую уже разделили, то на ней мы останавливаем рекурсию. Таким образом, мы не только получаем правильный ответ (поскольку мы не пересчитываем одну и ту же часть несколько раз), но и значительно ускоряем работу программы: мы будем обращаться к каждой части не более двух раз, а не K раз.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>


using namespace std;

using Candle = pair<int, int>;
using Piece = set<tuple<int, int, int, int>>;
```

```
int cutCake(int v0, int v1, int h0, int h1, vector<Candle> candles,
  Piece& alreadySharedPieces)
{
  auto isNew = alreadySharedPieces.emplace(v0, v1, h0, h1).second;
  if (!isNew)
    return 0;
  if (candles.size() == 1)
    return 1;

  int result = 0;
  auto height = v1 - v0;
  auto width = h1 - h0;

  if (height % 2 == 0)
  {
    auto middle = (v0 + v1) / 2;
    vector<Candle> bottomPart;
    vector<Candle> topPart;
    for (auto z : candles)
    {
      if (z.first < middle)
        bottomPart.push_back(z);
      else
        topPart.push_back(z);
    }
    if (bottomPart.size() == topPart.size())
    {
      result += cutCake(v0, middle, h0, h1, bottomPart,
        alreadySharedPieces);
      result += cutCake(middle, v1, h0, h1, topPart,
        alreadySharedPieces);
    }
  }
```

```
    if (width % 2 == 0)
    {
        auto middle = (h0 + h1) / 2;
        vector<Candle> leftPart;
        vector<Candle> rightPart;
        for (auto z : candles)
        {
            if (z.second < middle)
                leftPart.push_back(z);
            else
                rightPart.push_back(z);
        }
        if (leftPart.size() == rightPart.size())
        {
            result += cutCake(v0, v1, h0, middle, leftPart,
                alreadySharedPieces);
            result += cutCake(v0, v1, middle, h1, rightPart,
                alreadySharedPieces);
        }
    }

    return result;
}

int main()
{
    int M, N, K;
    vector<Candle> candles;
    Piece pieces;

    cin >> M >> N >> K;
    for (int i = 0; i < K; ++i)
    {
        int v, h;
```

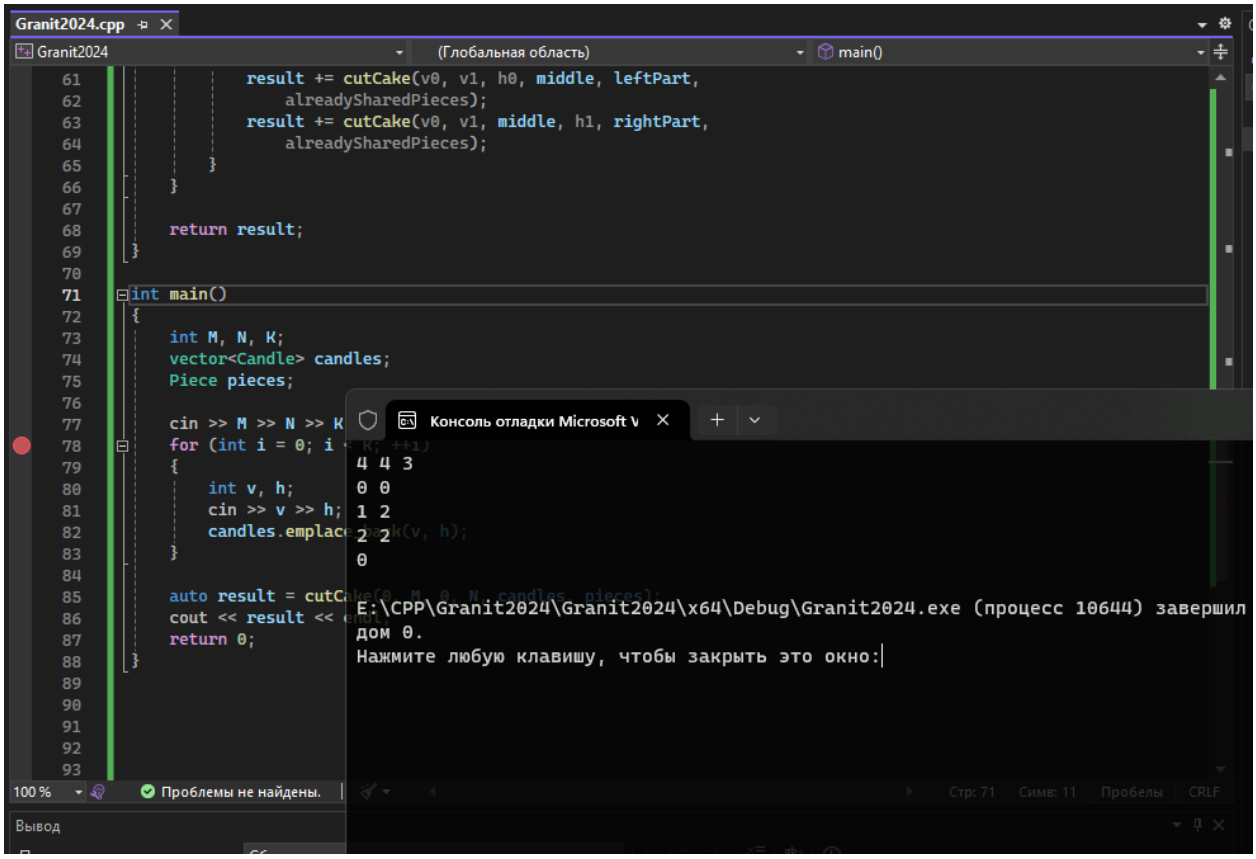
```
        cin >> v >> h;
        candles.emplace_back(v, h);
    }

    auto result = cutCake(0, M, 0, N, candles, pieces);
    cout << result << endl;
    return 0;
}
```

Реализация на Python

```
from typing import List, Tuple, Set

Candle = Tuple[int, int]
Piece = Set[Tuple[int, int, int, int]]

def cut_cake(v0: int, v1: int, h0: int, h1: int, candles: List[Candle],
             already_shared_pieces: Piece) -> int:
    if (v0, v1, h0, h1) in already_shared_pieces:
        return 0
    already_shared_pieces.add((v0, v1, h0, h1))
    if len(candles) == 1:
        return 1

    result = 0
    height = v1 - v0
    width = h1 - h0

    if height % 2 == 0:
        middle = (v0 + v1) // 2
        bottom_part = [z for z in candles if z[0] < middle]
        top_part = [z for z in candles if z[0] >= middle]
        if len(bottom_part) == len(top_part):
            result += cut_cake(v0, middle, h0, h1, bottom_part, already_shared_pieces)
```

```
            result += cut_cake(middle, v1, h0, h1, top_part, already_shared_pieces)

    if width % 2 == 0:
        middle = (h0 + h1) // 2
        left_part = [z for z in candles if z[1] < middle]
        right_part = [z for z in candles if z[1] >= middle]
        if len(left_part) == len(right_part):
            result += cut_cake(v0, v1, h0, middle, left_part, already_shared_pieces)
            result += cut_cake(v0, v1, middle, h1, right_part, already_shared_pieces)

    return result

def main():
    M, N, K = map(int, input().split())
    candles = [tuple(map(int, input().split())) for _ in range(K)]
    pieces = set()

    result = cut_cake(0, M, 0, N, candles, pieces)
    print(result)

if __name__ == "__main__":
    main()
```

Демонстрация работы



Ответ: 0

ИНФОРМАТИКА

ВАРИАНТ 2.

Задача 1 (10 баллов)

Определите названия ячеек в блоке A1:E1, используя подсказки.

	A	B	C	D	E
1					
2					
3	=ЗНАЧЕН(ПРАВСИМВ(A1;1))	=ЗНАЧЕН(ПРАВСИМВ(B1;1))	=ЗНАЧЕН(ПРАВСИМВ(C1;1))	=ЗНАЧЕН(ПРАВСИМВ(D1;1))	=ЗНАЧЕН(ПРАВСИМВ(E1;1))
4	=ЛЕВСИМВ(A1;1)	=ЛЕВСИМВ(B1;1)	=ЛЕВСИМВ(C1;1)	=ЛЕВСИМВ(D1;1)	=ЛЕВСИМВ(E1;1)
5					

	A	B	C
6	=A3-EXP(0)		=ПОИСК(A4;"bcadfe")
7	=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)		=A4=C4
8	=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3		=КОДСИМВ(B4)-КОДСИМВ(C4)
9	=C3-D3-E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))
10	=C3+D3*E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))

	A	B	C
6	1		2
7	7		ИСТИНА
8	4		-2
9	-2		4
10	17		3

Решение:

1. В соответствии с формулами «=ЗНАЧЕН(ПРАВСИМВ(A1;1))», «=ЗНАЧЕН(ПРАВСИМВ(B1;1))» и т.д. в ячейках A3:E3 находятся цифры обозначающие строки определяемых ячеек. В ячейках A4:E4 с формулами «=ЛЕВСИМВ(A1;1)», «=ЛЕВСИМВ(B1;1)» и т.д. соответственно, буквы латинского алфавита, обозначающие колонки определяемых ячеек.

2. Так как значение формулы «=A3-EXP(0)» в ячейке A6 равно 1, то A3 = 1 + EXP(0) = 2.

3. Значение формулы в A7 «=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)» равно 7, поэтому B3 = 7-A3-ОКРУГЛВВЕРХ(ПИ();0) = 7-2-4 = 1.

4. Значение формулы в A8 «=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3» равно 4, поэтому C3 = 4-B3+ ОКРУГЛВНИЗ(EXP(1);0) = 4-1+2 = 5.

5. Значение формулы в A9 «=C3-D3-E3» равно -2, а значение формулы в A10 «=C3+D3*E3» равно 17. Получаем систему уравнений: $\begin{cases} 5 - D3 - E3 = -2 \\ 5 + D3 * E3 = 17 \end{cases}$ => $\begin{cases} -D3 - E3 = -7 \\ D3 * E3 = 12 \end{cases}$, решая которую получим D3 = 4, E3 = 3.

6. Значение формулы в C6 «=ПОИСК(A4;"bcadfe")» равно 2, следовательно в A4 находится символ C.

7. Значение формулы в C7 «=A4=C4» равно «ИСТИНА», следовательно в C4 находится также символ C.

8. Значение формулы в C8 «=КОДСИМВ(B4)-КОДСИМВ(C4)» равно -2, следовательно в B4 находится символ, стоящий на 2 позиции ближе к началу алфавита от символа C, а это - A.

9. Значение формулы в C9 «=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 4, а значение формулы в C10 «=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 3. Получаем систему уравнений относительно смещения от начала алфавита символа в ячейке D4 и в ячейке E4. Если обозначим их, соответственно, как D и E, то получим: $\begin{cases} E + D = 4 \\ E * D = 3 \end{cases}$. Решая систему уравнений, определяем D = 3, E = 1. Следовательно в ячейке D4 находится символ D, а в ячейке E4 – символ B.

Ответ:

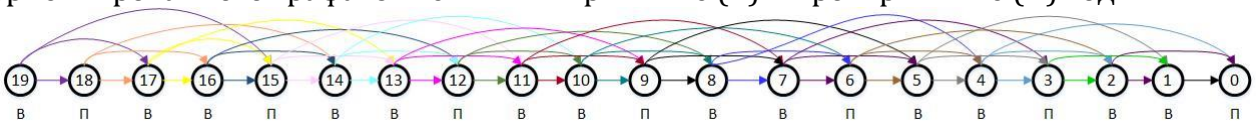
	A	B	C	D	E
1	C2	A1	C5	D4	B3

Задача 2 (10 баллов)

На столе лежит 19 драгоценных камня. Гномы, участвующие в игре, по очереди могут взять 1, 2 или 4 камня. Гном, который не может сделать ход (камней не осталось), – проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Какова должна быть стратегия, выигравшего гнома? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней стратегии выигрывающего гнома. С каким количеством камней этот алгоритм будет работать?

Решение:

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Делаем вывод, что с периодом 3: позиции, где число камней делится на 3 без остатка, будут проигрышными (для того, кто в них оказался), а где не делится — выигрышными. В нашем случае, когда в игре 19 камней, выигрывает первый игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось кратное трём количество.
Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней (в данном случае игрок 1 возьмёт 1 камень, чтобы осталось 18 или 4 камня, чтобы осталось 15).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет от 1,2 или 4 камня.

Ход_3: Игрок_1 берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней. И так далее до Победа Игрок_1. Проиграть невозможно.

Алгоритм для Игрок_1 (обобщенный):

Шаг 1 Игрок _1 взял К=1,2 или 4 камня, так чтобы (N-К) кратно 3
Если Остаток = 0 – значит СТОП_Выиграл.

Шаг 2 Игрок_2 взял К=1,2 или 4 камня
Алгоритм подходит в общем случае, для N камней, если N делится на 3 без остатка.

Задача 3 (10 баллов)

Дан фрагмент таблицы истинности и результирующий столбец G(x, y, z)?F(y, z). Укажите пропущенные значения, а также какие переменные в каком порядке указаны в столбцах, а также какой оператор должен стоять между функциями в результирующем столбце.

z	x	y	$x \wedge \bar{y}$	$z \vee \bar{x}$	$y \leftrightarrow z$	F(x, y) **?**F(x,z)	F(y,z) **?**G(x,yz)
0		0	0	1	1	0	1
0		1	0	1	0	0	0
	1	0	1	0	1	0	1
	1		0	0	0	1	1
	0	0	0	1	0	0	0

	0		0	1	1	0	1
1		0	1	1	0	1	1
1	1	1	0	1	1	0	1

Решение:

$x \wedge \bar{y}$ - это F(x,y)

$z \vee \bar{x}$ - это F(x,z)

$y \leftrightarrow z$ - это F (y,z)

G(x,y,z) – это F(x, y) $\leftrightarrow$F(x,z)

z	x	y	$x \wedge \bar{y}$	$z \vee \bar{x}$	$y \leftrightarrow z$	F(x, y) $\leftrightarrow$ F(x,z)	F(y,z) $\vee$ G(x,yz)
0	0	0	0	1	1	0	1
0	0	1	0	1	0	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	1	0	1	1
1	1	1	0	1	1	0	1

Задача 4 (20 баллов)

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

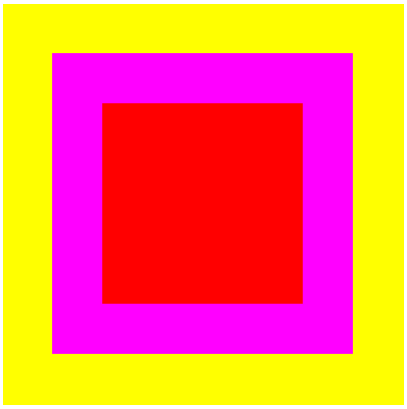


Рисунок 1

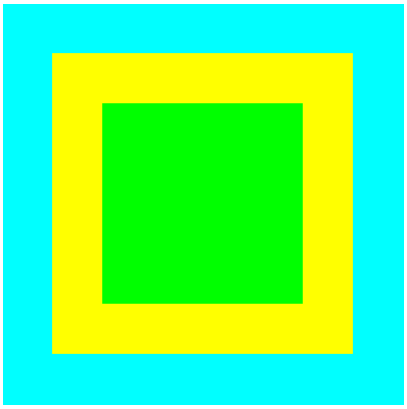


Рисунок 2

Решение:

Как мы видим, в обработанном изображении красный цвет стал зелёным. Делаем предположение, что произошла смена каналов. Значение яркости красной составляющей было записано в зелёный канал.

Жёлтый цвет стал голубым. Желтый цвет состоит из максимальных значений красной и зеленой составляющих. Так как мы уже знаем, что значение красной составляющей было записано в зелёный канал, делаем вывод, что значение зелёной составляющей было записано в синий канал. Таким образом был получен голубой цвет.

Значит, значение синей составляющей было записано в красный канал, что и подтверждает смена лилового цвета на жёлтый.

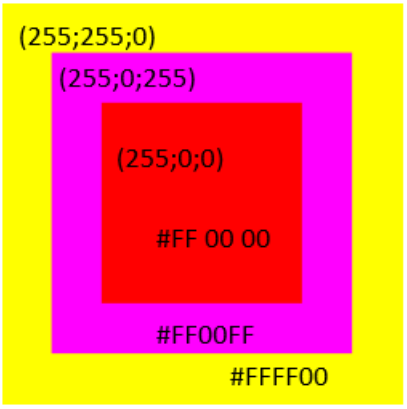
Алгоритм обработки выглядит следующим образом:

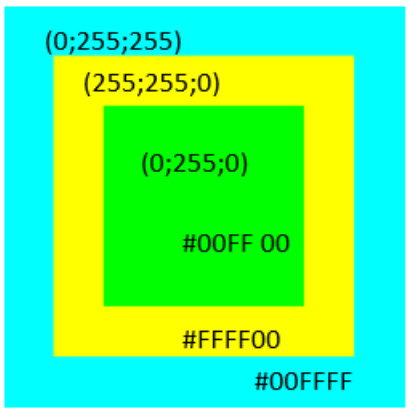
$$Y_R = Y_B$$

$$Y_G = Y_R$$

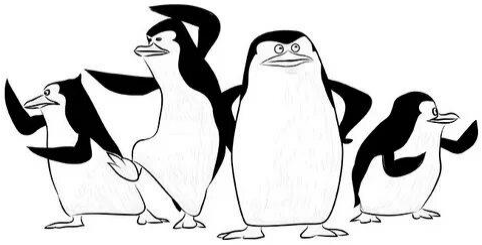
$$Y_B = Y_G$$

Далее укажем на рисунках значения яркости в RGB и HEX формате





Задача 5 (25 баллов)



Однажды гуляя по антарктическим просторам, группа переворачивателей пингвинов (Не удивляйтесь, есть и такая профессия) наткнулась на одно семейство. В течение нескольких дней они наблюдали по очереди за ними. И заметили, необычное поведение пингвинов. Было похоже, что они играют в какую-то игру чтобы не замерзнуть или просто весело провести время.

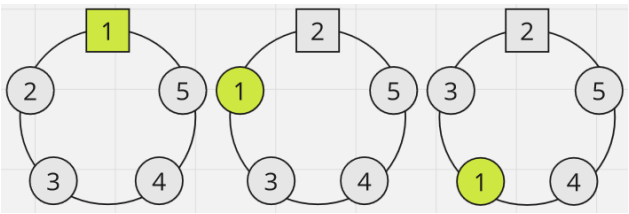
Пингвины вставали в круг лицом к центру рядом с одним из них всегда был небольшой снежный ком. Начиналась игра после того, как пингвин у камня подпрыгивал и похлопывал крыльями, после этого он менялся местами с соседом справа сколько-то раз. Когда первый пингвин перешел в новую позицию, другой пингвин, который оказался у снежного кома также подпрыгивал и тоже менялся с соседом справа, причем количество смен было больше, чем у предыдущего пингвина. Через некоторое количество таких передвижений, вероятно это был конец игры, пингвины разбегались и два которые были рядом с первым, приносили ему рыбку.

Переворачивателей заинтересовал этот необычный обряд у пингвинов, и продолжая наблюдать, они заметили, что количество перемещений совпадают с последовательностью простых чисел. Разумеется, они захотели попробовать эту игру. Для этого каждый из них присвоил себе цифру от 1 до *N*. Перед началом игры они нарисовали *N – 1*кругов и один квадрат (обозначающий снежный ком, как у пингвинов) в большом кругу. Переворачиватель с номером 1 встает в квадрат. Все остальные встают по порядку цифр, начиная со двойки, против часовой стрелки, лицом к центру.

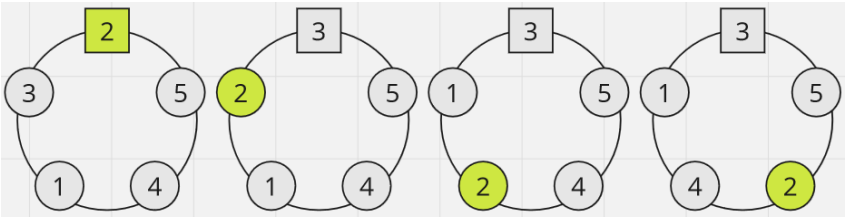
Они условились, что игра будет состоять из *M* раундов. В *i*-м раунде человек, который находиться в квадрате, подпрыгивает и кричит «Я пингвин» и затем меняется с человеком справа от него p_kраз, где p_k–простое число.

Например, для N = 5 и M = 3 происходят следующие три раунда:

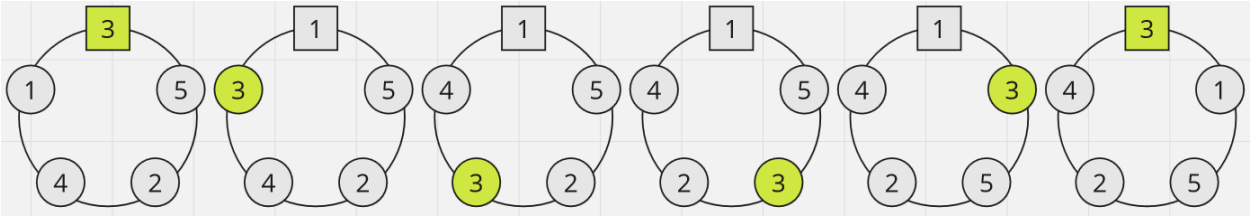
<u>1 - раунд</u>



<u>2 - раунд</u>



3 - раунд



Напишите программу, которая для заданных N, M и G определяет соседей человека под номером G в конце игры.

Входные данные: вам дается три числачерез пробел N, M, G, где N–количество участников, M–количество раундов и G– номер человека.

Результат должен содержатьномера двух соседей человека с номером G, сначала правого, потом левого.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод: 5 3 1	Результат: 3 5
Ввод: 5 3 2	Результат: 5 4
Ввод: 5 4 5	Результат: 3 2

Исходные данные:

9 5 8

Решение:

ПИНГВИГРА

Мы используем решето Эратосфена, чтобы найти первые K простых чисел.

Теперь обозначим квадрат цифрой 0, а круги - числами от 1 до N - 1. Поскольку невозможно смоделировать всю игру за отведенное время, мы сначала определяем алгоритм, который поможет нам найти конечную позицию G. Предположим, что в данный момент G находится на позиции **X**, а текущеепростое число - p:

- Если **X = 0**, то **G** находится в квадрате, поэтому **X** становится **N**mod**p**.
- Иначе, если **X ≠ 0**, то мы можем определить, сколько раз человек в квадрате меняется местами с **X**:
  - Если X<= p mod (N-1), то X становится (X - 1) mod N
  - Иначе X становится (X - p div (N-1)) mod N

где mod - оператор деления по модулю, a div - целочисленное деление.

Теперь нам нужно решить другую задачу чтобы ответить на вопрос "Если G находится на X, то кто находится на X-1 (или X+1)?".

Предположим, что нам известна позиция X**после** того, как был сыгран простое число p.

- Если X = p mod N, то Xбыл 0 перед простым числом p
- Иначе, если x ≠ p mod N:
  - x становится (x + p div (N-1)) mod N
  - Если x <= p mod (N-1), то x становится (x + 1) mod N

РеализациянаC++

```
#define_CRT_SECURE_NO_WARNINGS
```

```
#include<cstdio>

#defineMAXSIEVE 7380000
#defineMAXPRIMES 501000
int p[MAXPRIMES], np;
charsieve[MAXSIEVE];

int N;
intmod(inta) { return (a % N + N) % N; }

intforward(intx, intp) {
if (x == 0) return mod(p);

if (x<= p % (N - 1)) x = mod(x - 1);
x = mod(x - p / (N - 1));

returnx;
}

intbackward(intx, intp) {
if (x == mod(p)) return 0;

x = mod(x + p / (N - 1));
if (x<= p % (N - 1)) x = mod(x + 1);

returnx;
}

int main(void) {
for (inti = 2; i<MAXSIEVE; ++i) {
if (sieve[i]) continue;
      p[np++] = i;
for (int j = i + i; j <MAXSIEVE; j += i) sieve[j] = 1;
  }
```

```
int A, K;
scanf("%d%d%d", &N, &K, &A); --A;
for (inti = 0; i< K; ++i) A = forward(A, p[i]);

int L = mod(A - 1), R = mod(A + 1);

for (int i = K - 1; i>= 0; --i) L = backward(L, p[i]);
for (int i = K - 1; i>= 0; --i) R = backward(R, p[i]);

printf("%d %d\n", R + 1, L + 1);
return 0;
}
```

Реализация на Python

```
MAXSIEVE = 7380000
MAXPRIMES = 501000
p = []
np = 0
sieve = [False] * MAXSIEVE

def mod(a):
   return (a % N + N) % N

defforward(x, p):
   if x == 0:
      return mod(p)
   if x <= p % (N - 1):
      x = mod(x - 1)
   x = mod(x - p // (N - 1))
   return x

defbackward(x, p):
```

```
    if x == mod(p):
        return0
    x = mod(x + p // (N - 1))
    if x <= p % (N - 1):
        x = mod(x + 1)
    return x

foriinrange(2, MAXSIEVE):
    if sieve[i]:
        continue
    p.append(i)
    np += 1
    for j inrange(i + i, MAXSIEVE, i):
        sieve[j] = True

N, K, A = map(int, input().split())
A -= 1
foriin range(K):
    A = forward(A, p[i])

L = mod(A - 1)
R = mod(A + 1)

foriinrange(K - 1, -1, -1):
    L = backward(L, p[i])
foriinrange(K - 1, -1, -1):
    R = backward(R, p[i])

print(R + 1, L + 1)
```

Демонстрация на C++

main.cpp

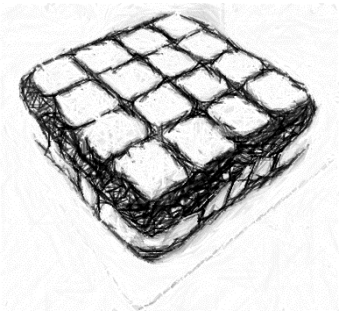
```
    x = mod(x + p/(N-1));
    if( x <= p%(N-1) ) x = mod(x+1);

    return x;
}

int main( void ) {
    for( int i = 2; i < MAXSIEVE; ++i ) {
        if( sieve[i] ) continue;
        p[np++] = i;
        for( int j = i+i; j < MAXSIEVE; j += i ) sieve[j] = 1;
    }

    int A, K;
    scanf( "%d%d%d", &N, &K, &A ); --A;
    for( int i = 0; i < K; ++i ) A = forward(A, p[i]);

    int L = mod(A-1), R = mod(A+1);

    for( int i = K-1; i >= 0; --i ) L = backward(L, p[i]);
    for( int i = K-1; i >= 0; --i ) R = backward(R, p[i]);

    printf( "%d %d\n", R+1, L+1 );
    return 0;
}
```

```
9 5 8
1 7
```

Ответ: 1 7

Задача 6 (25 баллов)

ОПЯТЬ ТОРТ

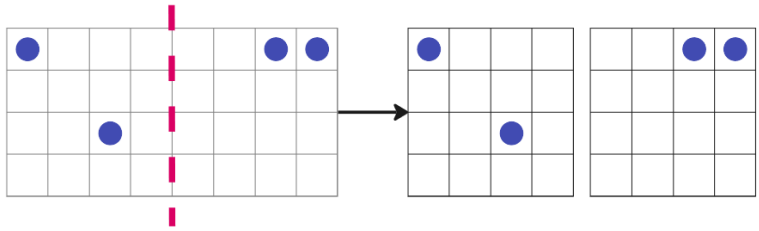


На день рождения Кати друзья испекли прямоугольный торт, украшенный K свечами – разумеется столько же сколько исполнилось лет Кате.Поскольку торт напечатан в клеточку, мы можем представить его как прямоугольник M × N.В некоторых ячейках расположена одна свеча, а в других нет.

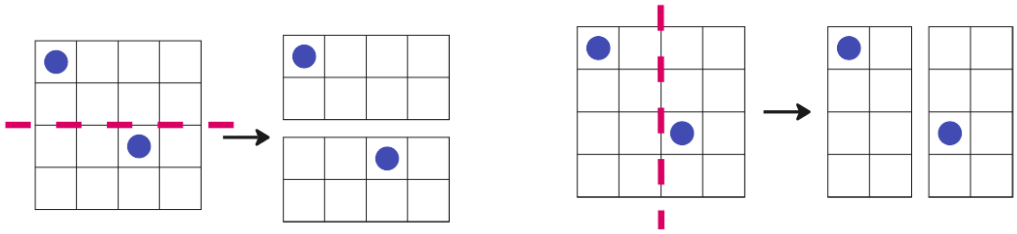
Лучшие друзья дали Кате задание отрезать кусок торта по определенным правилам:

- Горизонтальным или вертикальным разрезом, проходящим через края ячеек, торт разделяется на две прямоугольные части. Полученные кусочки должны быть одинакового размера и иметь одинаковое количество свечей.
- Далее Катя откладывает один из кусков в сторону, а остальные продолжает разрезать по тем же правилам.
- Когда остается кусок, который больше разрезать нельзя, т.е. в этом куске ровно одна свеча, он достается Кате. В противном случае Катя не получит торт.

Например, торт размером 4×8 можно разделить по вертикали на две части размером 4×4 по две свечи в каждой. Первый разрез не может быть горизонтальным, потому что, если вы разрежете середину, в верхней части будет три свечи, а в нижней - только одна.



По условию правый кусок не может быть разрезан дальше и имеет две свечи. Если бы Катяпродолжила разрезать его, она бы не получила торт. Левую часть можно разрезать как горизонтально, так и вертикально.



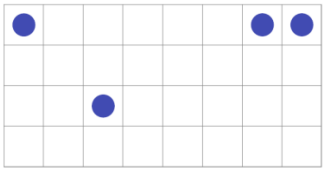
Разрезание в обе стороны дает по одной свече в каждой, поэтому любая из них может достаться к Кате. В этом примере Катя может получить одну из четырех разных частей. Напишите программу, которая считает, сколько разных фигур сможет получить Катя. Кусочки считаются разными, если они занимают в торте разное положение.

Входные данные: В первой строке записаны три целых числа: высота торта M, ширина N и количество свечей K. Следующие K строк содержат координаты кусочков со свечами. Первая координата вертикальная (нумеруется от 0 до M – 1 сверху вниз), а вторая горизонтальная (нумеруется от 0 до N – 1 слева направо).

Результат должен содержать одно число, обозначающее, сколько разных фигур может достаться Кате.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример входных данных

Ввод:	Результат:	Пример описан в условии.
4 8 4 0 0 2 2 0 6 0 7	4	

Исходные данные:

8 4 8

0 1

1 2

2 2

3 3

5 0

5 1

4 2

6 2

Решение:

ОПЯТЬ ТОРТ

Мы храним свечи в списке и делим торт рекурсивно. На каждом шаге мы также разбиваем список свечей на два списка — по одному на каждую из новых фигур.

Важная оптимизация — если мы доходим до части, которую уже разделили, то на ней мы останавливаем рекурсию. Таким образом, мы не только получаем правильный ответ (поскольку мы не пересчитываем одну и ту же часть несколько раз), но и

значительно ускоряем работу программы: мы будем обращаться к каждой части не более двух раз, а не K раз.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>


using namespace std;

using Candle = pair<int, int>;
using Piece = set<tuple<int, int, int, int>>;

int cutCake(int v0, int v1, int h0, int h1, vector<Candle> candles,
  Piece& alreadySharedPieces)
{
  auto isNew = alreadySharedPieces.emplace(v0, v1, h0, h1).second;
  if (!isNew)
    return 0;
  if (candles.size() == 1)
    return 1;

  int result = 0;
  auto height = v1 - v0;
  auto width = h1 - h0;

  if (height % 2 == 0)
  {
    auto middle = (v0 + v1) / 2;
    vector<Candle> bottomPart;
    vector<Candle> topPart;
```

```
    for (auto z : candles)
    {
        if (z.first < middle)
            bottomPart.push_back(z);
        else
            topPart.push_back(z);
    }
    if (bottomPart.size() == topPart.size())
    {
        result += cutCake(v0, middle, h0, h1, bottomPart,
            alreadySharedPieces);
        result += cutCake(middle, v1, h0, h1, topPart,
            alreadySharedPieces);
    }
}

if (width % 2 == 0)
{
    auto middle = (h0 + h1) / 2;
    vector<Candle> leftPart;
    vector<Candle> rightPart;
    for (auto z : candles)
    {
        if (z.second < middle)
            leftPart.push_back(z);
        else
            rightPart.push_back(z);
    }
    if (leftPart.size() == rightPart.size())
    {
        result += cutCake(v0, v1, h0, middle, leftPart,
            alreadySharedPieces);
        result += cutCake(v0, v1, middle, h1, rightPart,
            alreadySharedPieces);
```

```
        }
    }

    return result;
}

int main()
{
    int M, N, K;
    vector<Candle> candles;
    Piece pieces;

    cin >> M >> N >> K;
    for (int i = 0; i < K; ++i)
    {
        int v, h;
        cin >> v >> h;
        candles.emplace_back(v, h);
    }

    auto result = cutCake(0, M, 0, N, candles, pieces);
    cout << result << endl;
    return 0;
}
```

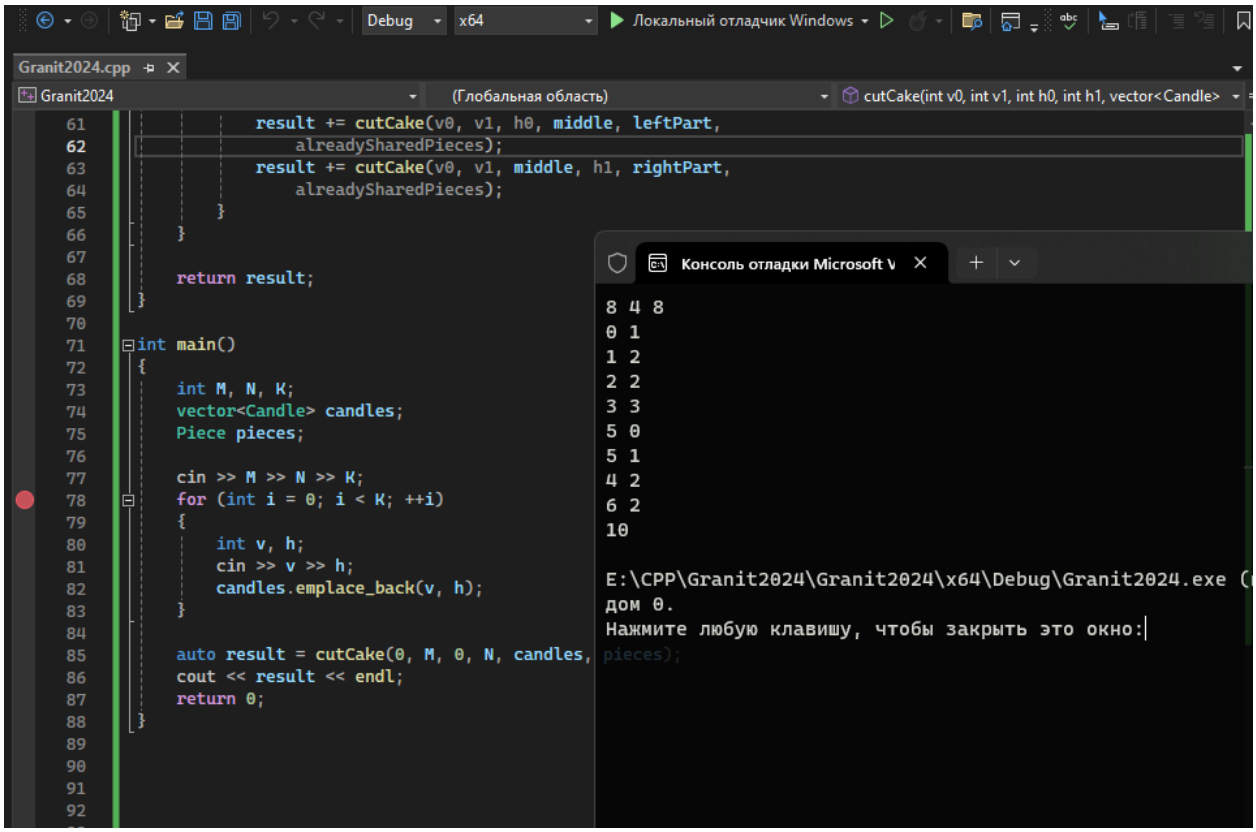
Реализация на Python

```
from typing import List, Tuple, Set

Candle = Tuple[int, int]
Piece = Set[Tuple[int, int, int, int]]

def cut_cake(v0: int, v1: int, h0: int, h1: int, candles: List[Candle],
             already_shared_pieces: Piece) -> int:
```

```
    if (v0, v1, h0, h1) in already_shared_pieces:
        return 0
    already_shared_pieces.add((v0, v1, h0, h1))
    if len(candles) == 1:
        return 1

    result = 0
    height = v1 - v0
    width = h1 - h0

    if height % 2 == 0:
        middle = (v0 + v1) // 2
        bottom_part = [z for z in candles if z[0] < middle]
        top_part = [z for z in candles if z[0] >= middle]
        if len(bottom_part) == len(top_part):
            result += cut_cake(v0, middle, h0, h1, bottom_part, already_shared_pieces)
            result += cut_cake(middle, v1, h0, h1, top_part, already_shared_pieces)

    if width % 2 == 0:
        middle = (h0 + h1) // 2
        left_part = [z for z in candles if z[1] < middle]
        right_part = [z for z in candles if z[1] >= middle]
        if len(left_part) == len(right_part):
            result += cut_cake(v0, v1, h0, middle, left_part, already_shared_pieces)
            result += cut_cake(v0, v1, middle, h1, right_part, already_shared_pieces)

    return result

def main():
    M, N, K = map(int, input().split())
    candles = [tuple(map(int, input().split())) for _ in range(K)]
    pieces = set()

    result = cut_cake(0, M, 0, N, candles, pieces)
```

```
    print(result)

if __name__ == "__main__":
    main()
```

Демонстрация работы



Ответ: 10

ИНФОРМАТИКА

ВАРИАНТ 3

Задача 1 (10 баллов)

Определите названия ячеек в блоке A1:E1, используя подсказки.

	A	B	C	D	E
1					
2					
3	=ЗНАЧЕН(ПРАВСИМВ(A1;1))	=ЗНАЧЕН(ПРАВСИМВ(B1;1))	=ЗНАЧЕН(ПРАВСИМВ(C1;1))	=ЗНАЧЕН(ПРАВСИМВ(D1;1))	=ЗНАЧЕН(ПРАВСИМВ(E1;1))
4	=ЛЕВСИМВ(A1;1)	=ЛЕВСИМВ(B1;1)	=ЛЕВСИМВ(C1;1)	=ЛЕВСИМВ(D1;1)	=ЛЕВСИМВ(E1;1)
5					

	A	B	C
6	=A3-EXP(0)		=ПОИСК(A4;"bcadfe")
7	=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)		=A4=C4
8	=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3		=КОДСИМВ(B4)-КОДСИМВ(C4)
9	=C3-D3-E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))
10	=C3+D3*E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))

	A	B	C
6	4		4
7	11		ИСТИНА
8	1		-2
9	-6		9
10	7		20

Решение:

1. В соответствии с формулами «=ЗНАЧЕН(ПРАВСИМВ(A1;1))», «=ЗНАЧЕН(ПРАВСИМВ(B1;1))» и т.д. в ячейках A3:E3 находятся цифры обозначающие строки определяемых ячеек. В ячейках A4:E4 с формулами «=ЛЕВСИМВ(A1;1)», «=ЛЕВСИМВ(B1;1)» и т.д. соответственно, буквы латинского алфавита, обозначающие колонки определяемых ячеек.

2. Так как значение формулы «=A3-EXP(0)» в ячейке A6 равно 4, то A3 = 4 +EXP(0) = 5.

3. Значение формулы в A7 «=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)» равно 11, поэтому B3 = 11-A3- ОКРУГЛВВЕРХ(ПИ();0) = 11-5-4 = 2.

4. Значение формулы в A8 «=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3» равно 1, поэтому C3 = 1-B3+ ОКРУГЛВНИЗ(EXP(1);0) = 1-2+2 = 1.

5. Значение формулы в A9 «=C3-D3-E3» равно -6, а значение формулы в A10 «=C3+D3*E3» равно 7. Получаем систему уравнений: $\begin{cases} 1 - D3 - E3 = -6 \\ 1 + D3 * E3 = 7 \end{cases}$ $\Rightarrow \begin{cases} -D3 - E3 = -7 \\ D3 * E3 = 8 \end{cases}$, решая которую получим D3 = 6, E3 = 1.

6. Значение формулы в C6 «=ПОИСК(A4;"bcadfe")» равно 4, следовательно в A4 находится символ D.

7. Значение формулы в C7 «=A4=C4» равно «ИСТИНА», следовательно в C4 находится также символ D.

8. Значение формулы в C8 «=КОДСИМВ(B4)-КОДСИМВ(C4)» равно -2, следовательно в B4 находится символ, стоящий на 2 позиции ближе к началу алфавита от символа D, а это - B.

9. Значение формулы в C9 «=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 9, а значение формулы в C10 «=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 20. Получаем систему уравнений относительно смещения от начала алфавита символа в ячейке D4 и в ячейке E4. Если обозначим их, соответственно, как Dи E, тополучим: $\begin{cases} E + D = 9 \\ E * D = 20 \end{cases}$. Решая систему уравнений, определяем D = 4, E = 5. Следовательно в ячейке D4 находится символ E, а в ячейке E4 – символ F.

Ответ:

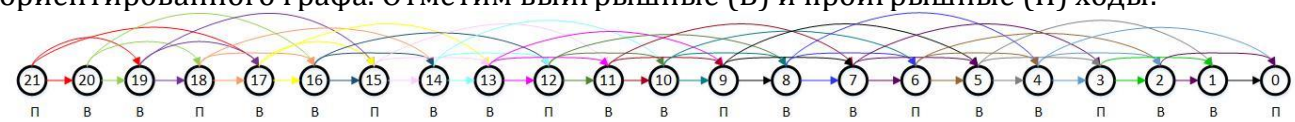
	A	B	C	D	E
1	D5	B2	D1	E6	F1

Задача 2 (10 баллов)

На столе лежит 21 драгоценных камень. Гномы, участвующие в игре, по очереди могут взять 1, 2 или 4 камня. Гном, который не может сделать ход (камней не осталось), – проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Какова должна быть стратегия, выигравшего гнома? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней стратегии выигрывающего гнома. С каким количеством камней этот алгоритм будет работать?

Решение:

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Делаем вывод, что с периодом 3: позиции, где число камней делится на 3 без остатка, будут проигрышными (для того, кто в них оказался), а где не делится — выигрышными. В нашем случае, когда в игре 21 камень, выигрывает второй игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось кратное трём количество.

Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет от 1,2 или 4 камня (в данном случае камней остаётся 20, 19 или 17).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней (в данном случае игрок 1 возьмёт 2 камня, если их осталось 20 или 17, 1 или 4 камня, если их осталось 19).

Ход_3: Игрок_1 берет от 1,2 или 4 камня.
Ход_4: Игрок_2 берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней. И так далее до Победа Игрок_2. Проиграть невозможно.

Алгоритм для Игрок_2 (обобщенный):

Шаг 1 Игрок _1 взял К=1,2 или 4 камня

Шаг 2 Игрок_2 взял К=1,2 или 4 камня, так чтобы (N-K) кратно 3
Если Остаток = 0 – значит СТОП_Выиграл.
Алгоритм подходит в общем случае, для N камней, если N делится на 3 без остатка.

Задача 3 (10 баллов)

Дан фрагмент таблицы истинности и результирующий столбец G(x, y, z)?F(y, z). Укажите пропущенные значения, а также какие переменные в каком порядке указаны в столбцах, а также какой оператор должен стоять между функциями в результирующем столбце.

?	?	?	$\bar{x} \wedge y$	$z \to x$	$\bar{y} \vee z$	F(x, y) ?F(x,z)	F(y,z) ?G(x,y,z)
	0	0	0	1	1	1	1
	0	1	1	1	0	1	0
0	1		0	0	1	1	1
0		1	1	0	1	0	0
1	0	0	0	1	1	1	1
	0	1	0	1	0	1	0
	1	0	0	1	1	1	1
1		1	0	1	1	1	1

Решение:

$\bar{x} \wedge y$ - это F(x,y)

$z \to x$ - это F(x,z)

$\bar{y} \vee z$ - это F(y,z)

G(x,y,z) – это F(x, y) →F(x,z)

x	z	y	$\bar{x} \wedge y$	$z \to x$	$\bar{y} \vee z$	F(x, y) → F(x,z)	F(y,z) ∧ G(x,y,z)
0	0	0	0	1	1	1	1
0	0	1	1	1	0	1	0
0	1	0	0	0	1	1	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	1	1
1	1	1	0	1	1	1	1

Задача 4 (20 баллов)

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

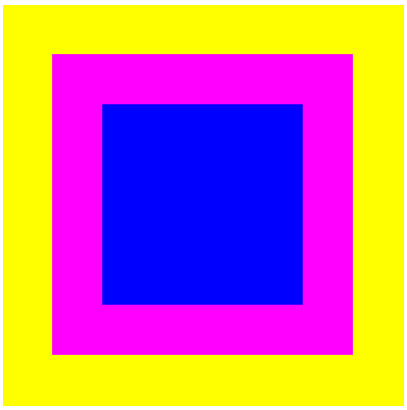


Рисунок 1

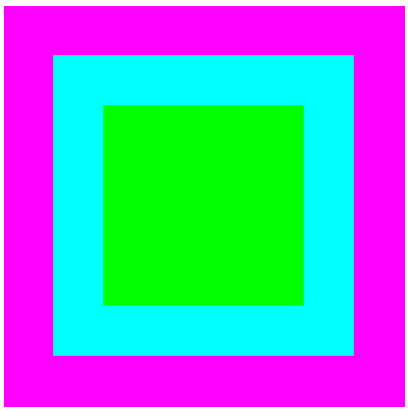


Рисунок 2

Решение:

Как мы видим, в обработанном изображении синий цвет стал зелёным. Делаем предположение, что произошла смена каналов. Значение яркости синей составляющей было записано в зелёный канал.

Лиловый цвет стал голубым. Лиловый цвет состоит из максимальных значений зелёной и синей составляющих. Так как мы уже знаем, что значение синей составляющей было записано в зелёный канал, делаем вывод, что значение зелёной составляющей было записано в красный канал. Таким образом был получен голубой цвет.

Значит, значение красной составляющей было записано в синий канал, что и подтверждает смена жёлтого цвета на лиловый.

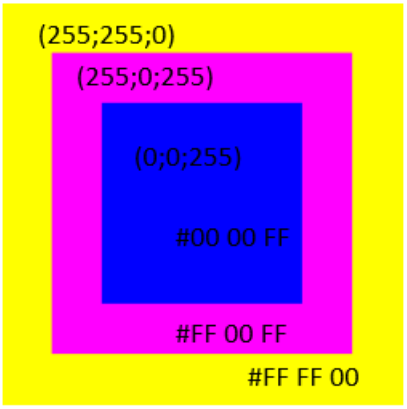
Алгоритм обработки выглядит следующим образом:

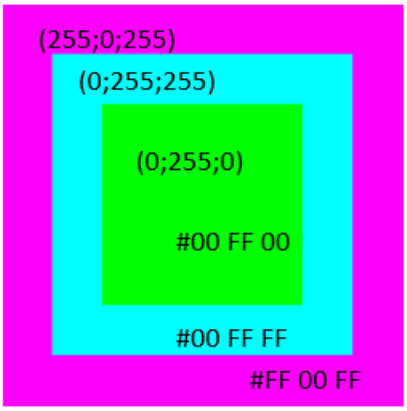
$$Y_R = Y_G$$

$$Y_G = Y_B$$

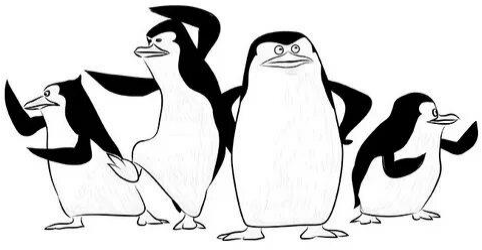
$$Y_B = Y_R$$

Далее укажем на рисунках значения яркости в RGB и HEX формате





Задача 5 (25 баллов)



Однажды гуляя по антарктическим просторам, группа переворачивателей пингвинов (Не удивляйтесь, есть и такая профессия) наткнулась на одно семейство. В течение нескольких дней они наблюдали по очереди за ними. И заметили, необычное поведение пингвинов. Было похоже, что они играют в какую-то игру чтобы не замерзнуть или просто весело провести время.

Пингвины вставали в круг лицом к центру рядом с одним из них всегда был небольшой снежный ком. Начиналась игра после того, как пингвин у камня подпрыгивал и похлопывал крыльями, после этого он менялся местами с соседом справа сколько-то раз. Когда первый пингвин перешел в новую позицию, другой пингвин, который оказался у снежного кома также подпрыгивал и тоже менялся с соседом справа, причем количество смен было больше, чем у предыдущего пингвина.

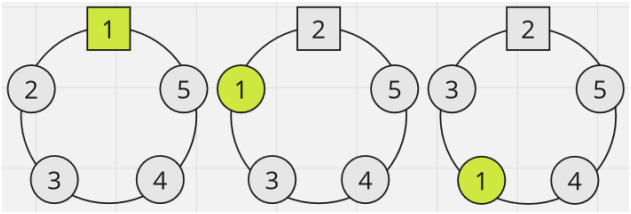
Через некоторое количество таких передвижений, вероятно это был конец игры, пингвины разбегались и два которые были рядом с первым, приносили ему рыбку.

Переворачивателей заинтересовал этот необычный обряд у пингвинов, и продолжая наблюдать, они заметили, что количество перемещений совпадают с последовательностью простых чисел. Разумеется, они захотели попробовать эту игру. Для этого каждый из них присвоил себе цифру от 1 до N. Перед началом игры они нарисовали N – 1 кругов и один квадрат (обозначающий снежный ком, как у пингвинов) в большом кругу. Переворачиватель с номером 1 встает в квадрат. Все остальные встают по порядку цифр, начиная со двойки, против часовой стрелки, лицом к центру.

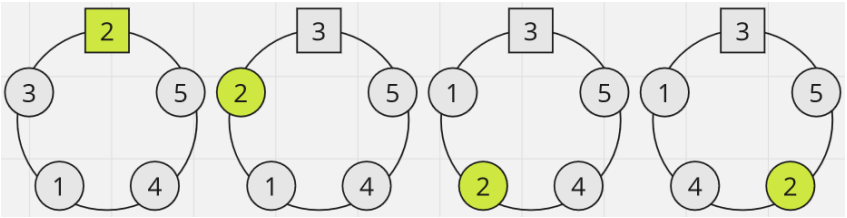
Они условились, что игра будет состоять из M раундов. В i-м раунде человек, который находиться в квадрате, подпрыгивает и кричит «Я пингвин» и затем меняется с человеком справа от него pk раз, где pk – простое число.

Например, для N = 5 и M = 3 происходят следующие три раунда:

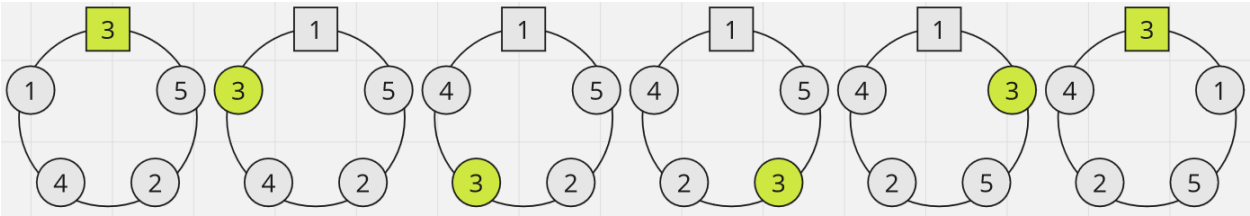
1 - раунд



2 - раунд



3 - раунд



Напишите программу, которая для заданных N, M и G определяет соседей человека под номером G в конце игры.

Входные данные: вам дается три числа через пробел N, M, G, где N – количество участников, M – количество раундов и G – номер человека.

Результат должен содержать номера двух соседей человека с номером G, сначала правого, потом левого.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод: 5 3 1	Результат: 3 5
Ввод: 5 3 2	Результат: 5 4
Ввод: 5 4 5	Результат: 3 2

Исходные данные:
8 7 5

Решение:

Мы используем решето Эратосфена, чтобы найти первые K простых чисел.

Теперь обозначим квадрат цифрой 0, а круги - числами от 1 до N - 1. Поскольку невозможно смоделировать всю игру за отведенное время, мы сначала определяем алгоритм, который поможет нам найти конечную позицию G. Предположим, что в данный момент G находится на позиции X, а текущее простое число - p:

- Если X = 0, то G находится в квадрате, поэтому X становится N mod p.
- Иначе, если X ≠ 0, то мы можем определить, сколько раз человек в квадрате меняется местами с X:
  - Если X <= p mod (N-1), то X становится (X - 1) mod N
  - Иначе X становится (X - p div (N-1)) mod N

где mod - оператор деления по модулю, a div - целочисленное деление.

Теперь нам нужно решить другую задачу чтобы ответить на вопрос "Если G находится на X, то кто находится на X-1 (или X+1)?".

Предположим, что нам известна позиция X после того, как был сыгран простое число p.

- Если X = p mod N, то X был 0 перед простым числом p
- Иначе, если x ≠ p mod N:
  - x становится (x + p div (N-1)) mod N
  - Если x <= p mod (N-1), то x становится (x + 1) mod N

Реализация на C++

```
#define _CRT_SECURE_NO_WARNINGS

#include <cstdio>
```

```
#define MAXSIEVE 7380000
#define MAXPRIMES 501000
int p[MAXPRIMES], np;
char sieve[MAXSIEVE];

int N;
int mod(int a) { return (a % N + N) % N; }

int forward(int x, int p) {
  if (x == 0) return mod(p);

  if (x <= p % (N - 1)) x = mod(x - 1);
  x = mod(x - p / (N - 1));

  return x;
}

int backward(int x, int p) {
  if (x == mod(p)) return 0;

  x = mod(x + p / (N - 1));
  if (x <= p % (N - 1)) x = mod(x + 1);

  return x;
}

int main(void) {
  for (int i = 2; i < MAXSIEVE; ++i) {
    if (sieve[i]) continue;
    p[np++] = i;
    for (int j = i + i; j < MAXSIEVE; j += i) sieve[j] = 1;
  }

  int A, K;
  scanf("%d%d%d", &N, &K, &A); --A;
  for (int i = 0; i < K; ++i) A = forward(A, p[i]);

  int L = mod(A - 1), R = mod(A + 1);

  for (int i = K - 1; i >= 0; --i) L = backward(L, p[i]);
  for (int i = K - 1; i >= 0; --i) R = backward(R, p[i]);

  printf("%d %d\n", R + 1, L + 1);
  return 0;
```

```
}
```

Реализация на Python

```
MAXSIEVE = 7380000
MAXPRIMES = 501000
p = []
np = 0
sieve = [False] * MAXSIEVE

def mod(a):
   return (a % N + N) % N

def forward(x, p):
   if x == 0:
      return mod(p)
   if x <= p % (N - 1):
      x = mod(x - 1)
   x = mod(x - p // (N - 1))
   return x

def backward(x, p):
   if x == mod(p):
      return 0
   x = mod(x + p // (N - 1))
   if x <= p % (N - 1):
      x = mod(x + 1)
   return x

for i in range(2, MAXSIEVE):
   if sieve[i]:
      continue
   p.append(i)
   np += 1
   for j in range(i + i, MAXSIEVE, i):
      sieve[j] = True

N, K, A = map(int, input().split())
A -= 1
for i in range(K):
   A = forward(A, p[i])

L = mod(A - 1)
R = mod(A + 1)

for i in range(K - 1, -1, -1):
   L = backward(L, p[i])
for i in range(K - 1, -1, -1):
   R = backward(R, p[i])
```

print(R + 1, L + 1)

Демонстрация на C++

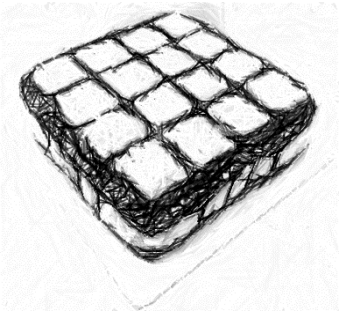
main.cpp

```
F8
    x = mod(x + p/(N-1));
    if( x <= p%(N-1) ) x = mod(x+1);

    return x;
}

int main( void ) {
    for( int i = 2; i < MAXSIEVE; ++i ) {
        if( sieve[i] ) continue;
        p[np++] = i;
        for( int j = i+i; j < MAXSIEVE; j += i ) sieve[j] = 1;
    }

    int A, K;
    scanf( "%d%d%d", &N, &K, &A ); --A;
    for( int i = 0; i < K; ++i ) A = forward(A, p[i]);

    int L = mod(A-1), R = mod(A+1);

    for( int i = K-1; i >= 0; --i ) L = backward(L, p[i]);
    for( int i = K-1; i >= 0; --i ) R = backward(R, p[i]);

    printf( "%d %d\n", R+1, L+1 );
    return 0;
}
```

```
8 7 5
2 1
```

Ответ: 2 1

Задача 6 (25 баллов)



ОПЯТЬ ТОРТ

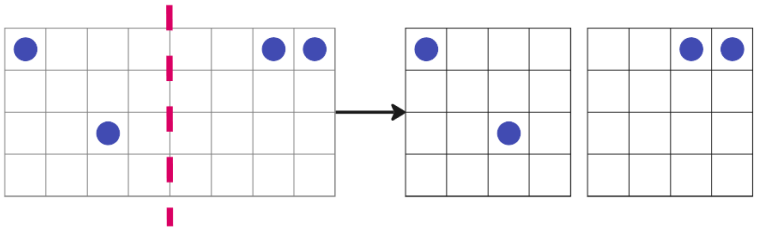
На день рождения Кати друзья испекли прямоугольный торт, украшенный K свечами – разумеется столько же сколько исполнилось лет Кате. Поскольку торт напечатан в клеточку, мы можем представить его как прямоугольник M × N. В некоторых ячейках расположена одна свеча, а в

других нет.

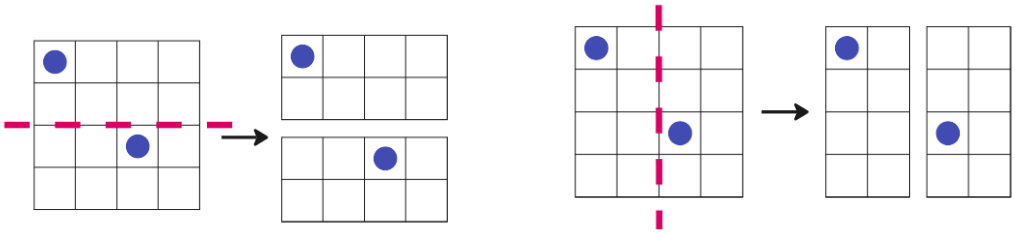
Лучшие друзья дали Кате задание отрезать кусок торта по определенным правилам:

- Горизонтальным или вертикальным разрезом, проходящим через края ячеек, торт разделяется на две прямоугольные части. Полученные кусочки должны быть одинакового размера и иметь одинаковое количество свечей.
- Далее Катя откладывает один из кусков в сторону, а остальные продолжает разрезать по тем же правилам.
- Когда остается кусок, который больше разрезать нельзя, т.е. в этом куске ровно одна свеча, он достается Кате. В противном случае Катя не получит торт.

Например, торт размером 4×8 можно разделить по вертикали на две части размером 4×4 по две свечи в каждой. Первый разрез не может быть горизонтальным, потому что, если вы разрежете середину, в верхней части будет три свечи, а в нижней - только одна.



По условию правый кусок не может быть разрезан дальше и имеет две свечи. Если бы Катя продолжила разрезать его, она бы не получила торт. Левую часть можно разрезать как горизонтально, так и вертикально.



Разрезание в обе стороны дает по одной свече в каждой, поэтому любая из них может достаться к Кате. В этом примере Катя может получить одну из четырех разных частей.

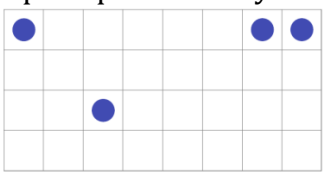
Напишите программу, которая считает, сколько разных фигур сможет получить Катя. Кусочки считаются разными, если они занимают в торте разное положение.

Входные данные: В первой строке записаны три целых числа: высота торта M, ширина N и количество свечей K. Следующие K строк содержат координаты кусочков со свечами. Первая координата вертикальная (нумеруется от 0 до M – 1 сверху вниз), а вторая горизонтальная (нумеруется от 0 до N – 1 слева направо).

Результат должен содержать одно число, обозначающее, сколько разных фигур может достаться Кате.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример входных данных

Ввод:	Результат:	Пример описан в условии.
4 8 4 0 0 2 2 0 6 0 7	4	

Исходные данные:

8 8 16
0 3
1 0
2 0
3 2
0 7
0 4
2 6
3 6
5 0
4 4
4 6
5 5
6 2
6 0
7 5
7 2

Решение:

Мы храним свечи в списке и делим торт рекурсивно. На каждом шаге мы также разбиваем список свечей на два списка — по одному на каждую из новых фигур.

Важная оптимизация — если мы доходим до части, которую уже разделили, то на ней мы останавливаем рекурсию. Таким образом, мы не только получаем правильный ответ (поскольку мы не пересчитываем одну и ту же часть несколько раз), но и значительно ускоряем работу программы: мы будем обращаться к каждой части не более двух раз, а не K раз.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>
```

```
using namespace std;

using Candle = pair<int, int>;
using Piece = set<tuple<int, int, int, int>>;

int cutCake(int v0, int v1, int h0, int h1, vector<Candle> candles,
  Piece& alreadySharedPieces)
{
  auto isNew = alreadySharedPieces.emplace(v0, v1, h0, h1).second;
  if (!isNew)
    return 0;
  if (candles.size() == 1)
    return 1;

  int result = 0;
  auto height = v1 - v0;
  auto width = h1 - h0;

  if (height % 2 == 0)
  {
    auto middle = (v0 + v1) / 2;
    vector<Candle> bottomPart;
    vector<Candle> topPart;
    for (auto z : candles)
    {
      if (z.first < middle)
        bottomPart.push_back(z);
      else
        topPart.push_back(z);
    }
    if (bottomPart.size() == topPart.size())
    {
      result += cutCake(v0, middle, h0, h1, bottomPart,
        alreadySharedPieces);
      result += cutCake(middle, v1, h0, h1, topPart,
        alreadySharedPieces);
    }
  }

  if (width % 2 == 0)
  {
    auto middle = (h0 + h1) / 2;
```

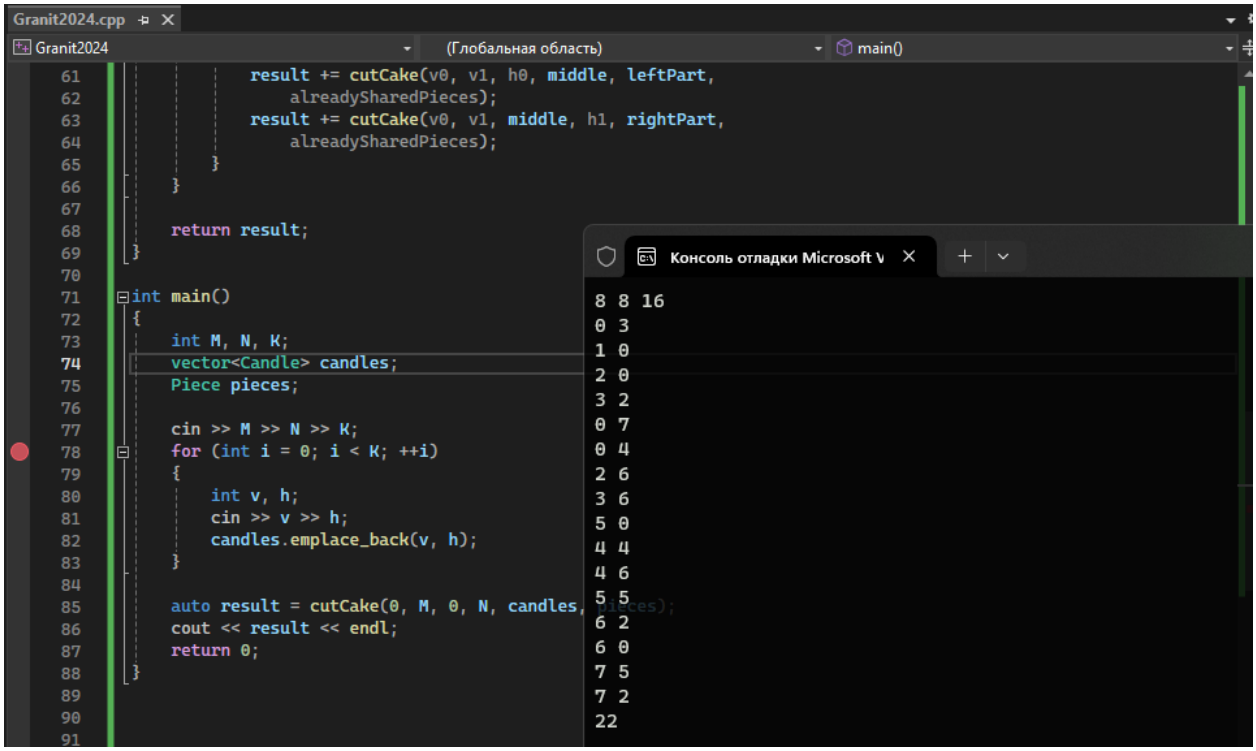
```
        vector<Candle> leftPart;
        vector<Candle> rightPart;
        for (auto z : candles)
        {
          if (z.second < middle)
            leftPart.push_back(z);
          else
            rightPart.push_back(z);
        }
        if (leftPart.size() == rightPart.size())
        {
          result += cutCake(v0, v1, h0, middle, leftPart,
            alreadySharedPieces);
          result += cutCake(v0, v1, middle, h1, rightPart,
            alreadySharedPieces);
        }
      }

      return result;
}

int main()
{
      int M, N, K;
      vector<Candle> candles;
      Piece pieces;

      cin >> M >> N >> K;
      for (int i = 0; i < K; ++i)
      {
        int v, h;
        cin >> v >> h;
        candles.emplace_back(v, h);
      }

      auto result = cutCake(0, M, 0, N, candles, pieces);
      cout << result << endl;
      return 0;
}
```

Реализация на Python

```
from typing import List, Tuple, Set
```

```
Candle = Tuple[int, int]
Piece = Set[Tuple[int, int, int, int]]

def cut_cake(v0: int, v1: int, h0: int, h1: int, candles: List[Candle],
             already_shared_pieces: Piece) -> int:
    if (v0, v1, h0, h1) in already_shared_pieces:
        return 0
    already_shared_pieces.add((v0, v1, h0, h1))
    if len(candles) == 1:
        return 1

    result = 0
    height = v1 - v0
    width = h1 - h0

    if height % 2 == 0:
        middle = (v0 + v1) // 2
        bottom_part = [z for z in candles if z[0] < middle]
        top_part = [z for z in candles if z[0] >= middle]
        if len(bottom_part) == len(top_part):
            result += cut_cake(v0, middle, h0, h1, bottom_part, already_shared_pieces)
            result += cut_cake(middle, v1, h0, h1, top_part, already_shared_pieces)

    if width % 2 == 0:
        middle = (h0 + h1) // 2
        left_part = [z for z in candles if z[1] < middle]
        right_part = [z for z in candles if z[1] >= middle]
        if len(left_part) == len(right_part):
            result += cut_cake(v0, v1, h0, middle, left_part, already_shared_pieces)
            result += cut_cake(v0, v1, middle, h1, right_part, already_shared_pieces)

    return result

def main():
    M, N, K = map(int, input().split())
    candles = [tuple(map(int, input().split())) for _ in range(K)]
    pieces = set()

    result = cut_cake(0, M, 0, N, candles, pieces)
    print(result)

if __name__ == "__main__":
    main()
```

Демонстрация работы


Granit2024.cpp
Granit2024
(Глобальная область)
main()
result += cutCake(v0, v1, h0, middle, leftPart,
alreadySharedPieces);
result += cutCake(v0, v1, middle, h1, rightPart,
alreadySharedPieces);
return result;
int main()
{
int M, N, K;
vector<Candle> candles;
Piece pieces;
cin >> M >> N >> K;
for (int i = 0; i < K; ++i)
{
int v, h;
cin >> v >> h;
candles.emplace_back(v, h);
auto result = cutCake(0, M, 0, N, candles, pieces);
cout << result << endl;
return 0;
Консоль отладки Microsoft V
8 8 16
0 3
1 0
2 0
3 2
0 7
0 4
2 6
3 6
5 0
4 4
4 6
5 5
6 2
6 0
7 5
7 2
22

Ответ: 22

ИНФОРМАТИКА.

ВАРИАНТ 4.

Задача 1 (10 баллов)

Определите названия ячеек в блоке A1:E1, используя подсказки.

	A	B	C	D	E
1					
2					
3	=ЗНАЧЕН(ПРАВСИМВ(A1;1))	=ЗНАЧЕН(ПРАВСИМВ(B1;1))	=ЗНАЧЕН(ПРАВСИМВ(C1;1))	=ЗНАЧЕН(ПРАВСИМВ(D1;1))	=ЗНАЧЕН(ПРАВСИМВ(E1;1))
4	=ЛЕВСИМВ(A1;1)	=ЛЕВСИМВ(B1;1)	=ЛЕВСИМВ(C1;1)	=ЛЕВСИМВ(D1;1)	=ЛЕВСИМВ(E1;1)
5					

	A	B	C
6	=A3-EXP(0)		=ПОИСК(A4;"bcadfe")
7	=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)		=A4=C4
8	=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3		=КОДСИМВ(B4)-КОДСИМВ(C4)
9	=C3-D3-E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))
10	=C3+D3*E3		=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))

	A	B	C
6	2		1
7	12		ИСТИНА
8	9		-1
9	-2		5
10	13		6

Решение:

1. В соответствии с формулами «=ЗНАЧЕН(ПРАВСИМВ(A1;1))», «=ЗНАЧЕН(ПРАВСИМВ(B1;1))» и т.д. в ячейках A3:E3 находятся цифры обозначающие строки определяемых ячеек. В ячейках A4:E4 с формулами «=ЛЕВСИМВ(A1;1)», «=ЛЕВСИМВ(B1;1)» и т.д. соответственно, буквы латинского алфавита, обозначающие колонки определяемых ячеек.

2. Так как значение формулы «=A3-EXP(0)» в ячейке A6 равно 2, то A3 = 2 +EXP(0) = 3.

3. Значение формулы в A7 «=A3+B3+ОКРУГЛВВЕРХ(ПИ();0)» равно 12, поэтому B3 = 12-A3- ОКРУГЛВВЕРХ(ПИ();0) = 12-3-4 = 5.

4. Значение формулы в A8 «=B3-ОКРУГЛВНИЗ(EXP(1);0)+C3» равно 9, поэтому C3 = 9-B3+ ОКРУГЛВНИЗ(EXP(1);0) = 9-5+2 = 6.

5. Значение формулы в A9 «=C3-D3-E3» равно -2, а значение формулы в A10 «=C3+D3*E3» равно 13. Получаем систему уравнений: $\begin{cases} 6 - D3 - E3 = -2 \\ 6 + D3 * E3 = 13 \end{cases}$ $\Rightarrow \begin{cases} -D3 - E3 = -8 \\ D3 * E3 = 7 \end{cases}$, решая которую получим D3 = 1, E3 = 7.

6. Значение формулы в C6 «=ПОИСК(A4;"bcadfe")» равно 1, следовательно в A4 находится символ B.

7. Значение формулы в C7 «=A4=C4» равно «ИСТИНА», следовательно в C4 находится также символ B.

8. Значение формулы в C8 «=КОДСИМВ(B4)-КОДСИМВ(C4)» равно -1, следовательно в B4 находится символ, стоящий на 1 позицию ближе к началу алфавита от символа B, а это - A.

9. Значение формулы в C9 «=(КОДСИМВ(E4)-КОДСИМВ("A"))+(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 5, а значение формулы в C10 «=(КОДСИМВ(E4)-КОДСИМВ("A"))*(КОДСИМВ(D4)-КОДСИМВ("A"))» равно 6. Получаем систему уравнений относительно смещения от начала алфавита символа в ячейке D4 и в ячейке E4. Если обозначим их, соответственно, как D и E, тополучим: $\begin{cases} E + D = 5 \\ E * D = 6 \end{cases}$. Решая систему уравнений, определяем D = 2, E = 3. Следовательно в ячейке D4 находится символ C, а в ячейке E4 – символ D.

Ответ:

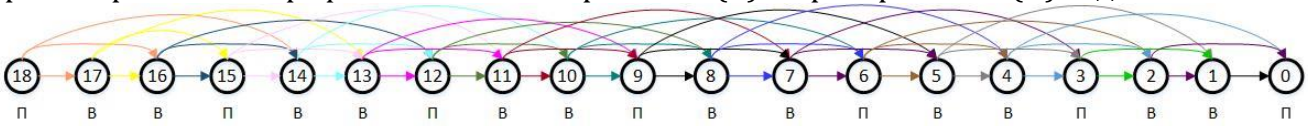
	A	B	C	D	E
1	B3	A5	B6	C1	D7

Задача 2 (10 баллов)

На столе лежит 18 драгоценных камней. Гномы, участвующие в игре, по очереди могут взять 1, 2 или 4 камня. Гном, который не может сделать ход (камней не осталось), – проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Какова должна быть стратегия, выигравшего гнома? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней стратегии выигрывающего гнома. С каким количеством камней этот алгоритм будет работать?

Решение:

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Делаем вывод, что с периодом 3: позиции, где число камней делится на 3 без остатка, будут проигрышными (для того, кто в них оказался), а где не делится — выигрышными. В нашем случае, когда в игре 18 камней, выигрывает второй игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось кратное трём количество.
Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет от 1,2 или 4 камня (в данном случае камней остаётся 17, 16 или 14).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней (в данном случае игрок 1 возьмёт 2 камня, если их осталось 17 или 14, 1 или 4 камня, если их осталось 16).

Ход_3: Игрок_1 берет от 1,2 или 4 камня.
Ход_4: Игрок_2 берет от 1,2 или 4 камней, так чтобы на ход противника осталось кратное 3 число камней. И так далее до Победа Игрок_2. Проиграть невозможно.

Алгоритм для Игрок_2 (обобщенный):

Шаг 1 Игрок _1 взял К=1,2 или 4 камня

Шаг 2 Игрок_2 взял К=1,2 или 4 камня, так чтобы (N-K) кратно 3
Если Остаток = 0 – значит СТОП_Выиграл.
Алгоритм подходит в общем случае, для N камней, если N делится на 3 без остатка.

Задача 3 (10 баллов)

Дан фрагмент таблицы истинности и результирующий столбец F(y,z) ?G(x,y, z). Укажите пропущенные значения, а также какие переменные в каком порядке указаны в столбцах, а также какой оператор должен стоять между функциями в результирующем столбце.

?	?	?	$x \leftrightarrow \bar{y}$	$y \to z$	$x \wedge \bar{z}$	F(x, y) ?F(x,z)	F(y,z) ?G(x,y, z)
0	0		0	1	0	0	0
			1	1	1	1	1
	1	0	1	0	0	0	1
	1	1	0	0	1	0	1
1	0	0	0	1	0	0	0
	0		1	1	0	0	0
		0	1	1	0	0	0
1	1	1	0	1	0	0	0

Решение:

$x \leftrightarrow \bar{y}$ - это F(x,y)

$y \to z$ - это F(y,z)

$x \wedge \bar{z}$ - это F(x,z)

G(x,y,z) – это F(x, y) ∧ F(x,z)

z	y	x	$x \leftrightarrow \bar{y}$	$y \rightarrow z$	$x \wedge \bar{z}$	F(x, y) ∧ F(x,z)	F(y,z) → G(x,y,z)
0	0	0	0	1	0	0	0
0	0	1	1	1	1	1	1
0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	1
1	0	0	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0

Задача 4.

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

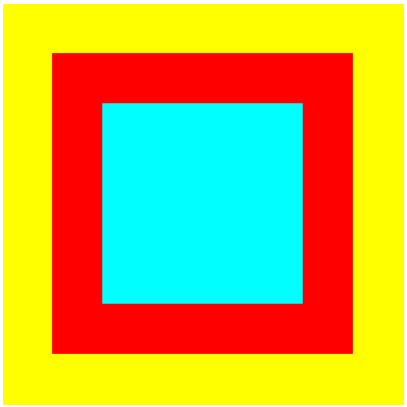


Рисунок 1

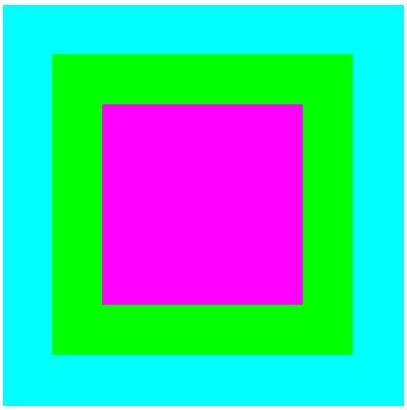


Рисунок 2

Решение:

Как мы видим, в обработанном изображении красный цвет стал зелёным. Делаем предположение, что произошла смена каналов. Значение яркости красной составляющей было записано в зелёный канал.

Жёлтый цвет стал голубым. Жёлтый цвет состоит из максимальных значений красной и зелёной составляющих. Так как мы уже знаем, что значение красной составляющей было записано в зелёный канал, делаем вывод, что значение зелёной составляющей было записано в синий канал. Таким образом был получен голубой цвет.

Значит, значение синей составляющей было записано в красный канал, что и подтверждает смена голубого цвета на лиловый.

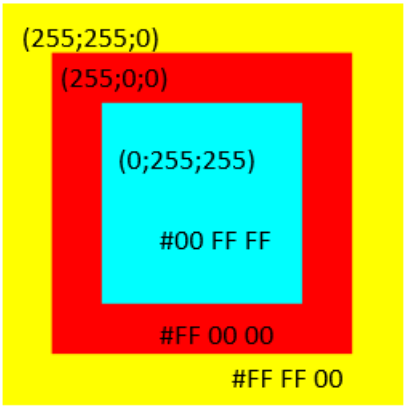
Алгоритм обработки выглядит следующим образом:

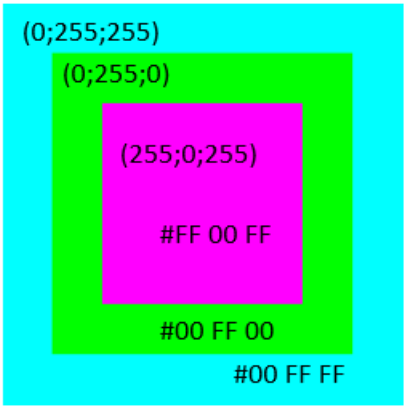
$$Y_R = Y_B$$

$$Y_G = Y_R$$

$$Y_B = Y_G$$

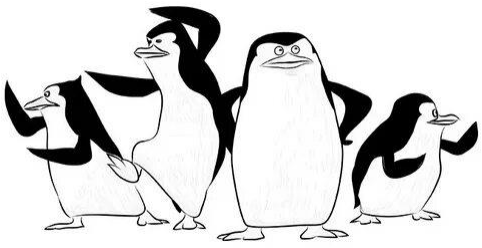
Далее укажем на рисунках значения яркости в RGB и HEX формате





Задача 5.

ПИНГВИГРА



Однажды гуляя по антарктическим просторам, группа переворачивателей пингвинов (Не удивляйтесь, есть и такая профессия) наткнулась на одно семейство. В течение нескольких дней они наблюдали по очереди за ними. И заметили, необычное поведение пингвинов. Было похоже, что они играют в какую-то игру чтобы не замерзнуть или просто весело провести время.

Пингвины вставали в круг лицом к центру рядом с одним из них всегда был небольшой снежный ком. Начиналась игра после того, как пингвин у камня подпрыгивал и похлопывал крыльями, после этого он менялся местами с соседом справа сколько-то раз. Когда первый пингвин перешел в новую позицию, другой пингвин, который оказался у снежного кома также подпрыгивал и тоже менялся с соседом справа, причем

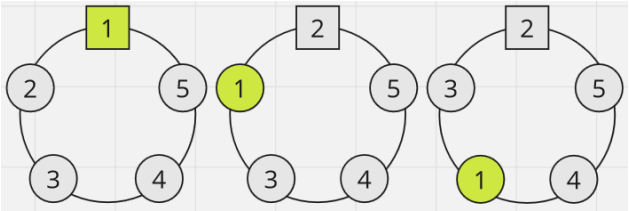
количество смен было больше, чем у предыдущего пингвина. Через некоторое количество таких передвижений, вероятно это был конец игры, пингвины разбегались и два которые были рядом с первым, приносили ему рыбку.

Переворачивателей заинтересовал этот необычный обряд у пингвинов, и продолжая наблюдать, они заметили, что количество перемещений совпадают с последовательностью простых чисел. Разумеется, они захотели попробовать эту игру. Для этого каждый из них присвоил себе цифру от 1 до *N*. Перед началом игры они нарисовали *N – 1* кругов и один квадрат (обозначающий снежный ком, как у пингвинов) в большом кругу. Переворачиватель с номером 1 встает в квадрат. Все остальные встают по порядку цифр, начиная со двойки, против часовой стрелки, лицом к центру.

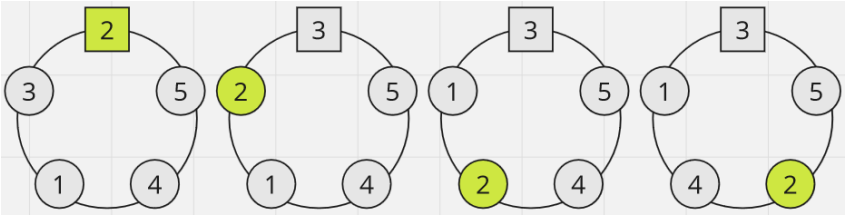
Они условились, что игра будет состоять из *M* раундов. В *i*-м раунде человек, который находиться в квадрате, подпрыгивает и кричит «Я пингвин» и затем меняется с человеком справа от него p_k раз, где p_k – простое число.

Например, для N = 5 и M = 3 происходят следующие три раунда:

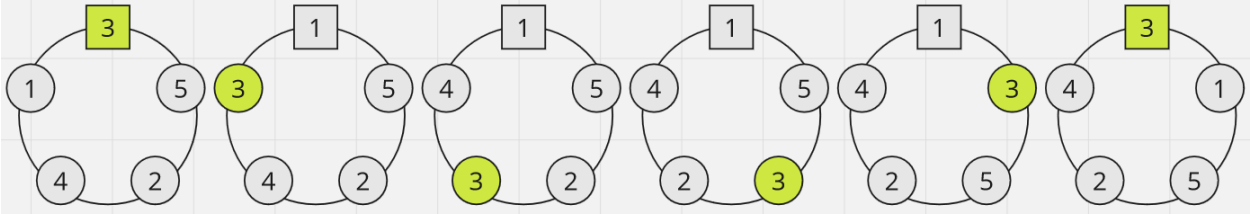
1 - раунд



2 - раунд



3 - раунд



Напишите программу, которая для заданных N, M и G определяет соседей человека под номером G в конце игры.

Входные данные: вам дается три числа через пробел N, M, G, где N – количество участников, M – количество раундов и G – номер человека.

Результат должен содержать номера двух соседей человека с номером G, сначала правого, потом левого.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод: 5 3 1	Результат: 3 5
Ввод: 5 3 2	Результат: 5 4
Ввод: 5 4 5	Результат: 3 2

Исходные данные:

8 7 6

Решение:

Мы используем решето Эратосфена, чтобы найти первые K простых чисел.

Теперь обозначим квадрат цифрой 0, а круги - числами от 1 до N - 1. Поскольку невозможно смоделировать всю игру за отведенное время, мы сначала определяем алгоритм, который поможет нам найти конечную позицию G. Предположим, что в данный момент G находится на позиции **X**, а текущее простое число - p:

- Если **X = 0**, то **G** находится в квадрате, поэтому **X** становится **N** mod **p**.
- Иначе, если **X ≠ 0**, то мы можем определить, сколько раз человек в квадрате меняется местами с **X**:
  - Если X <= p mod (N-1), то X становится (X - 1) mod N

- Иначе X становится (X - p div (N-1)) mod N

где mod - оператор деления по модулю, а div - целочисленное деление.

Теперь нам нужно решить другую задачу чтобы ответить на вопрос "Если G находится на X, то кто находится на X-1 (или X+1)?".

Предположим, что нам известна позиция X **после** того, как был сыгран простое число p.

- Если X = p mod N, то X был 0 перед простым числом p
- Иначе, если x ≠ p mod N:
  - x становится (x + p div (N-1)) mod N
  - Если x <= p mod (N-1), то x становится (x + 1) mod N

Реализация на C++

```
#define _CRT_SECURE_NO_WARNINGS

#include <cstdio>

#define MAXSIEVE 7380000
#define MAXPRIMES 501000
int p[MAXPRIMES], np;
char sieve[MAXSIEVE];

int N;
int mod(int a) { return (a % N + N) % N; }

int forward(int x, int p) {
  if (x == 0) return mod(p);

  if (x <= p % (N - 1)) x = mod(x - 1);
```

```
    x = mod(x - p / (N - 1));

    return x;
}

int backward(int x, int p) {
    if (x == mod(p)) return 0;

    x = mod(x + p / (N - 1));
    if (x <= p % (N - 1)) x = mod(x + 1);

    return x;
}

int main(void) {
    for (int i = 2; i < MAXSIEVE; ++i) {
        if (sieve[i]) continue;
        p[np++] = i;
        for (int j = i + i; j < MAXSIEVE; j += i) sieve[j] = 1;
    }

    int A, K;
    scanf("%d%d%d", &N, &K, &A); --A;
    for (int i = 0; i < K; ++i) A = forward(A, p[i]);

    int L = mod(A - 1), R = mod(A + 1);

    for (int i = K - 1; i >= 0; --i) L = backward(L, p[i]);
    for (int i = K - 1; i >= 0; --i) R = backward(R, p[i]);

    printf("%d %d\n", R + 1, L + 1);
    return 0;
}
```

Реализация на Python

```
MAXSIEVE = 7380000
MAXPRIMES = 501000
p = []
np = 0
sieve = [False] * MAXSIEVE

def mod(a):
    return (a % N + N) % N

def forward(x, p):
    if x == 0:
        return mod(p)
    if x <= p % (N - 1):
        x = mod(x - 1)
    x = mod(x - p // (N - 1))
    return x

def backward(x, p):
    if x == mod(p):
        return 0
    x = mod(x + p // (N - 1))
    if x <= p % (N - 1):
        x = mod(x + 1)
    return x

for i in range(2, MAXSIEVE):
    if sieve[i]:
        continue
    p.append(i)
    np += 1
    for j in range(i + i, MAXSIEVE, i):
        sieve[j] = True
```

```
N, K, A = map(int, input().split())
A -= 1
for i in range(K):
    A = forward(A, p[i])

L = mod(A - 1)
R = mod(A + 1)

for i in range(K - 1, -1, -1):
    L = backward(L, p[i])
for i in range(K - 1, -1, -1):
    R = backward(R, p[i])

print(R + 1, L + 1)
```

Демонстрация на C++

main.cpp

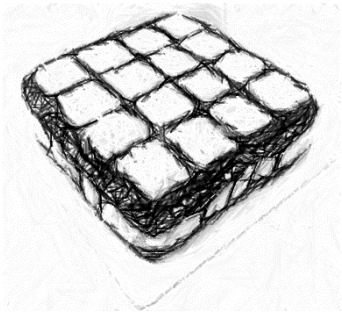
```
    x = mod(x + p/(N-1));
    if( x <= p%(N-1) ) x = mod(x+1);

    return x;
}

int main( void ) {
    for( int i = 2; i < MAXSIEVE; ++i ) {
        if( sieve[i] ) continue;
        p[np++] = i;
        for( int j = i+i; j < MAXSIEVE; j += i ) sieve[j] = 1;
    }

    int A, K;
    scanf( "%d%d%d", &N, &K, &A ); --A;
    for( int i = 0; i < K; ++i ) A = forward(A, p[i]);

    int L = mod(A-1), R = mod(A+1);

    for( int i = K-1; i >= 0; --i ) L = backward(L, p[i]);
    for( int i = K-1; i >= 0; --i ) R = backward(R, p[i]);

    printf( "%d %d\n", R+1, L+1 );
    return 0;
}
```

```
8 7 6
4 2
```

Ответ: 4 2

Задача 6.

ОПЯТЬ ТОРТ

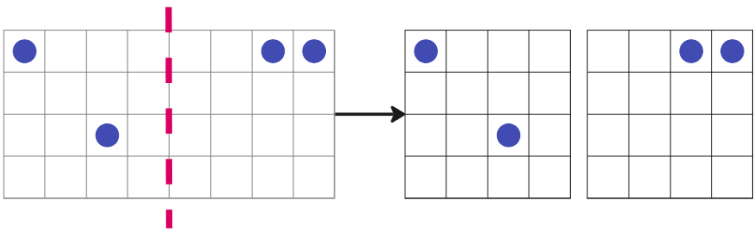


На день рождения Кати друзья испекли прямоугольный торт, украшенный К свечами – разумеется столько же сколько исполнилось лет Кате. Поскольку торт напечатан в клеточку, мы можем представить его как прямоугольник M × N. В некоторых ячейках расположена одна свеча, а в других нет.

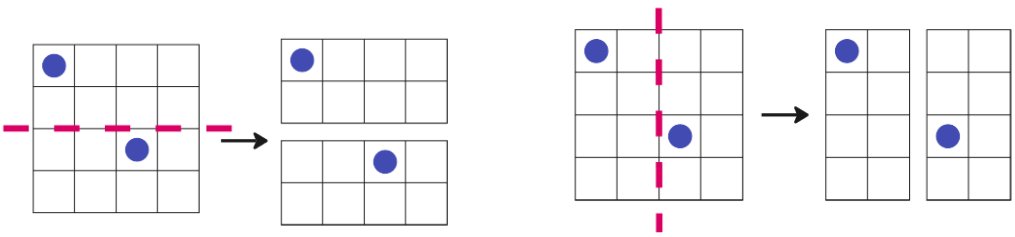
Лучшие друзья дали Кате задание отрезать кусок торта по определенным правилам:

- Горизонтальным или вертикальным разрезом, проходящим через края ячеек, торт разделяется на две прямоугольные части. Полученные кусочки должны быть одинакового размера и иметь одинаковое количество свечей.
- Далее Катя откладывает один из кусков в сторону, а остальные продолжает разрезать по тем же правилам.
- Когда остается кусок, который больше разрезать нельзя, т.е. в этом куске ровно одна свеча, он достается Кате. В противном случае Катя не получит торт.

Например, торт размером 4×8 можно разделить по вертикали на две части размером 4×4 по две свечи в каждой. Первый разрез не может быть горизонтальным, потому что, если вы разрежете середину, в верхней части будет три свечи, а в нижней - только одна.



По условию правый кусок не может быть разрезан дальше и имеет две свечи. Если бы Катя продолжила разрезать его, она бы не получила торт. Левую часть можно разрезать как горизонтально, так и вертикально.



Разрезание в обе стороны дает по одной свече в каждой, поэтому любая из них может достаться к Кате. В этом примере Катя может получить одну из четырех разных частей.

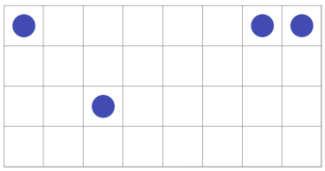
Напишите программу, которая считает, сколько разных фигур сможет получить Катя. Кусочки считаются разными, если они занимают в торте разное положение.

Входные данные: В первой строке записаны три целых числа: высота торта M, ширина N и количество свечей K. Следующие K строк содержат координаты кусочков со свечами. Первая координата вертикальная (нумеруется от 0 до M – 1 сверху вниз), а вторая горизонтальная (нумеруется от 0 до N – 1 слева направо).

Результат должен содержать одно число, обозначающее, сколько разных фигур может достаться Кате.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример входных данных

Ввод:	Результат:	Пример описан в условии.
4 8 4 0 0 2 2 0 6 0 7	4	

Исходные данные:

10 10 4
4 2
2 2
5 5
2 7

Решение:

ОПЯТЬ ТОРТ

Мы храним свечи в списке и делим торт рекурсивно. На каждом шаге мы также разбиваем список свечей на два списка — по одному на каждую из новых фигур.

Важная оптимизация — если мы доходим до части, которую уже разделили, то на ней мы останавливаем рекурсию. Таким образом, мы не только получаем правильный ответ (поскольку мы не пересчитываем одну и ту же часть несколько раз), но и значительно ускоряем работу программы: мы будем обращаться к каждой части не более двух раз, а не K раз.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>


using namespace std;

using Candle = pair<int, int>;
using Piece = set<tuple<int, int, int, int>>;

int cutCake(int v0, int v1, int h0, int h1, vector<Candle> candles,
  Piece& alreadySharedPieces)
{
  auto isNew = alreadySharedPieces.emplace(v0, v1, h0, h1).second;
  if (!isNew)
    return 0;
  if (candles.size() == 1)
    return 1;

  int result = 0;
  auto height = v1 - v0;
  auto width = h1 - h0;
```

```
if (height % 2 == 0)
{
   auto middle = (v0 + v1) / 2;
   vector<Candle> bottomPart;
   vector<Candle> topPart;
   for (auto z : candles)
   {
      if (z.first < middle)
         bottomPart.push_back(z);
      else
         topPart.push_back(z);
   }
   if (bottomPart.size() == topPart.size())
   {
      result += cutCake(v0, middle, h0, h1, bottomPart,
         alreadySharedPieces);
      result += cutCake(middle, v1, h0, h1, topPart,
         alreadySharedPieces);
   }
}

if (width % 2 == 0)
{
   auto middle = (h0 + h1) / 2;
   vector<Candle> leftPart;
   vector<Candle> rightPart;
   for (auto z : candles)
   {
      if (z.second < middle)
         leftPart.push_back(z);
      else
         rightPart.push_back(z);
   }
```

```
        if (leftPart.size() == rightPart.size())
        {
            result += cutCake(v0, v1, h0, middle, leftPart,
                alreadySharedPieces);
            result += cutCake(v0, v1, middle, h1, rightPart,
                alreadySharedPieces);
        }
    }

    return result;
}

int main()
{
    int M, N, K;
    vector<Candle> candles;
    Piece pieces;

    cin >> M >> N >> K;
    for (int i = 0; i < K; ++i)
    {
        int v, h;
        cin >> v >> h;
        candles.emplace_back(v, h);
    }

    auto result = cutCake(0, M, 0, N, candles, pieces);
    cout << result << endl;
    return 0;
}
```

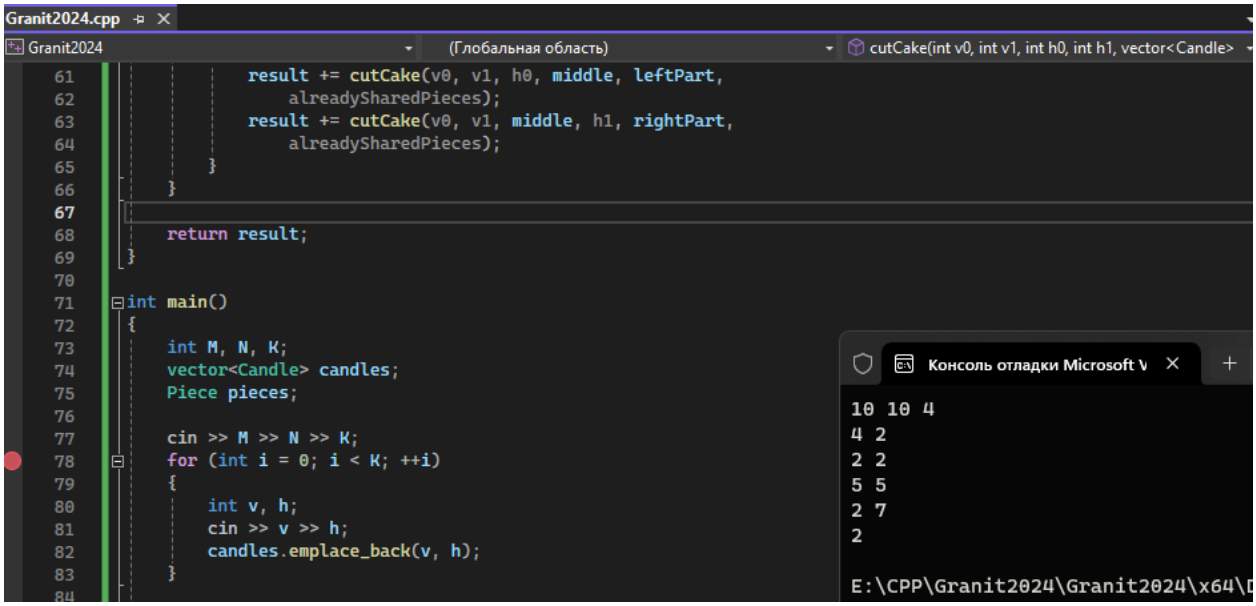
Реализация на Python

```
from typing import List, Tuple, Set
```

```
Candle = Tuple[int, int]
Piece = Set[Tuple[int, int, int, int]]

def cut_cake(v0: int, v1: int, h0: int, h1: int, candles: List[Candle],
             already_shared_pieces: Piece) -> int:
    if (v0, v1, h0, h1) in already_shared_pieces:
        return 0
    already_shared_pieces.add((v0, v1, h0, h1))
    if len(candles) == 1:
        return 1

    result = 0
    height = v1 - v0
    width = h1 - h0

    if height % 2 == 0:
        middle = (v0 + v1) // 2
        bottom_part = [z for z in candles if z[0] < middle]
        top_part = [z for z in candles if z[0] >= middle]
        if len(bottom_part) == len(top_part):
            result += cut_cake(v0, middle, h0, h1, bottom_part, already_shared_pieces)
            result += cut_cake(middle, v1, h0, h1, top_part, already_shared_pieces)

    if width % 2 == 0:
        middle = (h0 + h1) // 2
        left_part = [z for z in candles if z[1] < middle]
        right_part = [z for z in candles if z[1] >= middle]
        if len(left_part) == len(right_part):
            result += cut_cake(v0, v1, h0, middle, left_part, already_shared_pieces)
            result += cut_cake(v0, v1, middle, h1, right_part, already_shared_pieces)

    return result
```

```
def main():
    M, N, K = map(int, input().split())
    candles = [tuple(map(int, input().split())) for _ in range(K)]
    pieces = set()

    result = cut_cake(0, M, 0, N, candles, pieces)
    print(result)

if __name__ == "__main__":
    main()
```

Демонстрация работы



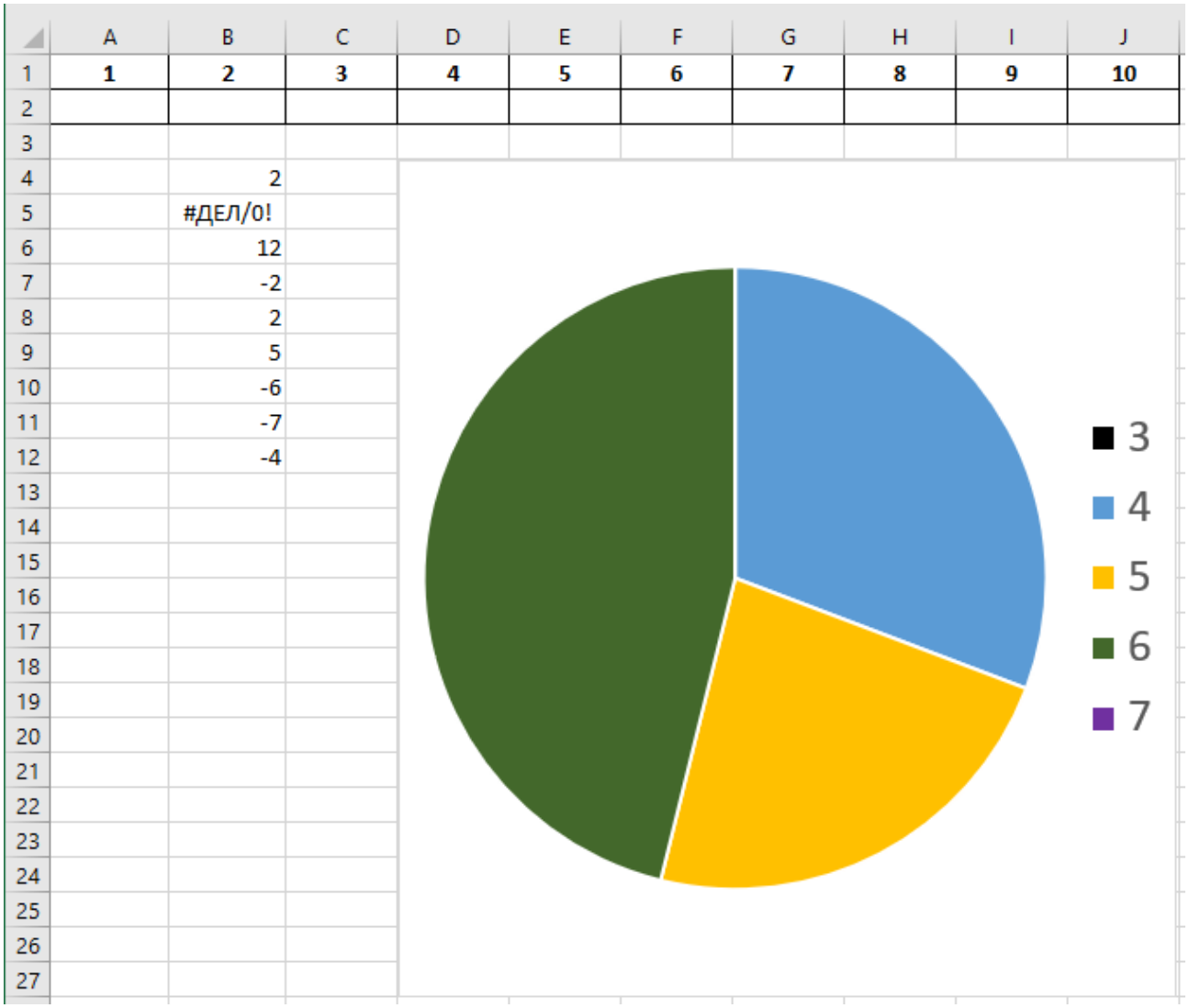
Ответ: 2

ИНФОРМАТИКА

ВАРИАНТ 5

Задача 1 (10 баллов)

В каждой из ячеек A2:J2 находится целое число. Определите их значения, используя круговую диаграмму и значения формул в блоке B4:B12. В формулах используются только ссылки на ячейки из диапазона A2:J2.



Задача усложняется тем, что отображение самих формул изменилось после копирования их в блок A5:A13.

	A
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")
6	=A3*D$2/($F3-G3)
7	=$F$2+F$2+$H3
8	=#ССЫЛКА!-E$2+#ССЫЛКА!
9	=ЕСЛИ(A3+D3;1;2)
10	=$A3+A3
11	=D3+C3+H3
12	=D3+C3*H3
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3

Решение:

1. Восстановим формулы в блоке B4:B12 исходя из формул блока A5:A13.Для получения формулы из ячейки A8 «=#ССЫЛКА!-E$2+#ССЫЛКА!» вспомним, что #ССЫЛКА! – это ссылка на ячейку, вышедшую за диапазон листа. Так как при копировании произошло смещение ссылок на 1 позицию вниз и 1 позицию влево, а также учитывая, что по условию задачи, в формулах используются только ссылки на ячейки из диапазона A2:J2, единственной ссылкой на ячейку способной выйти за диапазон листа является A2.

	A	B
4		=СЧЁТЕСЛИ($A$2:$J2;"<0")
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")	=B2*E$2/($F2-H2)
6	=A3*D$2/($F3-G3)	=$F$2+G$2+$H2
7	=$F$2+F$2+$H3	=A2-F$2+A2
8	=#ССЫЛКА!-E$2+#ССЫЛКА!	=ЕСЛИ(B2+E2;1;2)
9	=ЕСЛИ(A3+D3;1;2)	=$A2+B2
10	=$A3+A3	=E2+D2+I2
11	=D3+C3+H3	=E2+D2*I2
12	=D3+C3*H3	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3	
14		

2. По круговой диаграмме, которая, в соответствии с легендой, отображает значения категорий 3,4,5,6 и 7, видно, что значения 3 и 7 равны 0, т.е. C2=0и G2=0. Замечание: на круговой диаграмме отрицательные значения отображаются как положительные.

3. Значение формулы «=СЧЁТЕСЛИ($A$2:$J2;"<0")» в ячейке B4 равно 2, следовательно в искомом диапазоне имеется 2 отрицательных числа.

4. Значение формулы «=B2*E$2/($F2-H2)» в ячейке B5 равно значению «#ДЕЛ/0!», это означает, что ячейки F2 и H2 равны.

5. Значение формулы «=$F$2+G$2+$H2» в ячейке B6 равно 12, следовательноF2+H2 = 12-G2 = 12–0 = 12, а так как они равны, то F2 = 6, H2 = 6.

6. Значение формулы «=A2-F$2+A$2» в ячейке B7 равно -2, следовательно -2 = A2-6+A2. Откуда A2 = 4/2 = 2.

7. Значение формулы «=ЕСЛИ(B2+E2;1;2» в ячейке B8 равно 2, следовательно условие в виде суммы «B2+E2» не выполнено, т.е. B2+E2 = 0. Делаем вывод, что значения в ячейках B2 и E2 одинаковые по модулю и различные по знаку.

8. Значение формулы «=$A2+B2» в ячейке B9 равно 5, следовательно B2 = 5-A2 = 5-2 = 3,а в соответствии с 7 пунктом E2 = -3.

9. Значение формулы в B10 «=E2+D2+I2» равно -6, а значение формулы в B11 «=E2+D2*I2» равно -7. Получаем систему уравнений $\begin{cases} E2 + D2 + I2 = -6 \\ E2 + D2 * I2 = -7 \end{cases} => \begin{cases} -3 + D2 + I2 = -6 \\ -3 + D2 * I2 = -7 \end{cases} => \begin{cases} D2 + I2 = -3 \\ D2 * I2 = -4 \end{cases}$. Решая систему уравнений, находимD2 = -4 иI2 = 1.

10. Значение формулы «=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2» в ячейке B12 равно -4, следовательно J2 = ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0) + 4 = ОКРУГЛВВЕРХ (1,22222;0) + 4 = 2 + 4 = 6

Ответ:

	A	B	C	D	E	F	G	H	I	J
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	2	3	0	-4	-3	6	0	6	1	6

Задача 2 (10 баллов)

На столе лежит две кучки драгоценных камней: в одной 11, в другой 8. Гномы ходят по очереди. За один ход можно взять один камень из одной из кучек (по выбору гнома) или взять по одному камню из двух сразу. Кто не может сделать ход (камней не осталось), проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Какова должна быть стратегия, выигравшего гнома? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней.

Решение:

Необходимо использовать стратегию, беря такое количество камней, чтобы в каждой кучке оставалось чётное количество камней. Позиции, где число камней в каждой кучке чётное, будут проигрышными (для того, кто в них оказался), а где хотя бы одна

кучка с нечётным количеством камней — выигрышными. В нашем случае, когда в игре в одной 11, в другой 8 камней, выигрывает первый игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось чётное количество в каждой кучке.

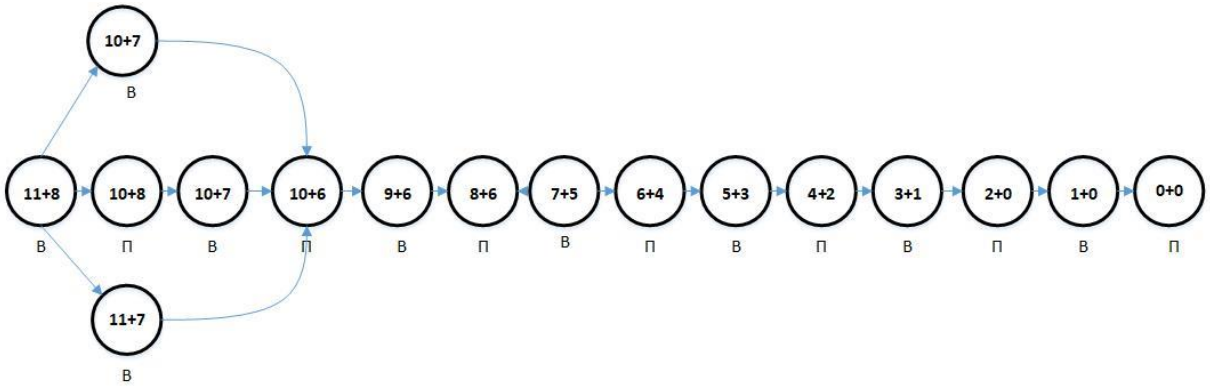
Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней (в данном случае игрок 1 возьмёт 1 камень из первой кучки).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет 1 камень из любой кучки или по 1 камню из каждой кучки (в данном случае камней может остаться 9 и 7, 10 и 7, 9 и 8).

Ход_3: Игрок_1 берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней. И так далее до Победа Игрок_1. Проиграть невозможно.

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Алгоритм для Игрок_1 (обобщенный):
Если M нечётное N чётное
Шаг 1 Игрок _1 взял m=1 камень, так чтобы (M-m) кратно 2
Если Остаток = 0 – значит СТОП_Выиграл.

Если N нечётное M чётное

Шаг 1 Игрок _1 взял n=1 камень, так чтобы (N-n) кратно 2

Если N нечётное M нечётное

Шаг 1 Игрок_1 взял m=1 и n=1 камни, так чтобы (M-m) и (N-n) кратно 2

Если Остаток = 0 – значит СТОП_Выиграл.

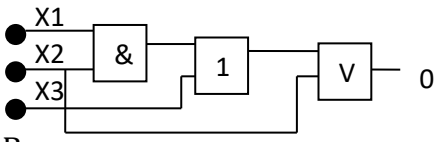
Шаг 2 Игрок _2 взял m=1 или n=1 или (m=1 и n=1) камней

Алгоритм подходит в общем случае, для M и N камней, если нечётно или M, или N, или оба вместе.

Задача 3 (10 баллов)

Дана логическая схема. Найти все наборы, при которых выражение, записанное в схеме, принимает значение ложь. Подставить i+1 наборы значений из таблицы истинности в логическое выражение для определения, чему равно результирующее выражение. В ответе указать, чему будет равен результирующий столбец.

Логическая схема:



Выражение:

$x1 \to \bar{x}2 \oplus x3 \vee x1$

Где  - это конъюнкция

 - это исключающее или

 - это дизъюнкция

Решение:

Каждый i-ый набор -это тот, при котором выражение, записанное с помощью логической схемы дает значение, указанное по условию задачи (ложь). В условии сказано подставить i+1 наборы, это значит, что необходимо подставить 2, 4 и 6 наборы, т.к. 1, 3 и 5 дали в результирующем столбце «ложь».

x1	x2	x3	x1&x2	x1&x2 $\oplus$ x3	x1&x2 $\oplus$ x3 $\vee$ x2
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	1

0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	1	0	1

$x1 \to \bar{x}2 \oplus x3 \vee x1$

x1	x2	x3	$\bar{x}\,2$	$x3 \vee x1$	$x1 \to \bar{x}2$	$x1 \to \bar{x}2 \oplus x3 \vee x1$
0	0	1	1	1	1	0
1	0	1	1	1	1	0

Ответ. 00

Задача 4 (20 баллов)

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

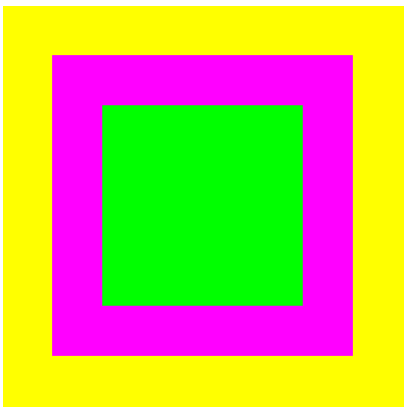


Рисунок 1

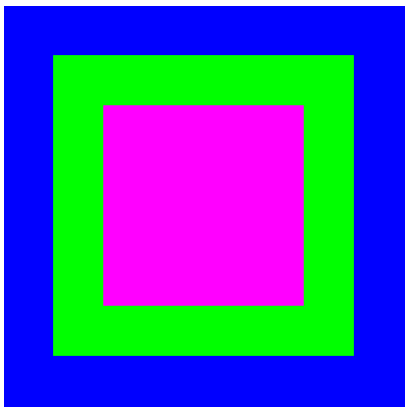


Рисунок 2

Решение:

Как мы видим, в обработанном изображении зелёный цвет стал лиловым. Получается, что после обработки, яркости зелёной составляющей приняли нулевые значения, а яркости красной и синей приняли значение 255. Делаем предположение, что была

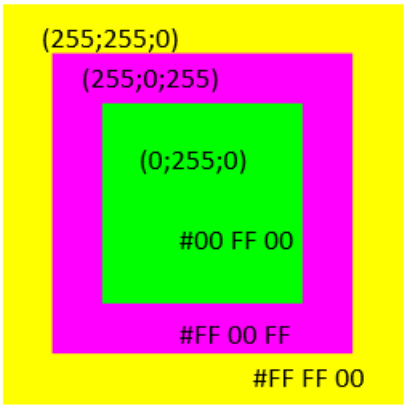
произведена инверсия изображения по трем каналам RGB, то есть алгоритм обработки выглядит следующим образом:

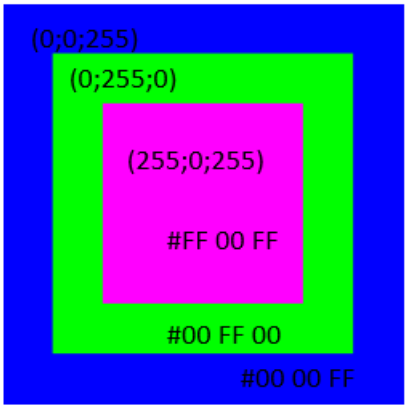
$$Y_R = 255 - Y_R$$
$$Y_G = 255 - Y_G$$
$$Y_B = 255 - Y_B$$

Жёлтый цвет стал синим, то есть яркости красной и зелёной составляющих приняли нулевые значения, а яркости синей составляющей приняли значение 255.

Лиловый цвет стал зелёным, то есть яркости красной и синей составляющих приняли нулевые значения, а яркости зелёной составляющей приняли значение 255.

Далее укажем на рисунках значения яркости в RGB и HEX формате

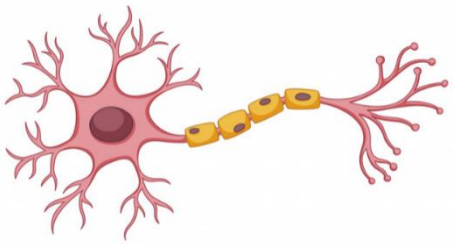




Задача 5 (25 баллов)

ОТКРЫТИЯ АРКТИКИ

Однажды после очередной экспедиции в Арктику ученые университета привезли образец льдины, добытый из глубин арктических пластов. При исследовании его в лаборатории химии они обнаружили новые клетки красного цвета.



Клетка сама по себе ничего не делала, и ученые уже расстроились, но однажды очень жарким летом, сломались кондиционеры в лаборатории, и клетка проснулась. За ночь клетка размножилась и стала похожа на дерево, состоящее из соединительных каналов и N прикрепленных к ним таких же клеток, из которых R красные, а остальные зеленые. Позже, выяснилось, что новые клетки тоже размножаются, и каждую последующую ночь все красные клетки вырастают в другое дерево, подобное тому, которое выросло из первой клетки в первую ночь.

Помогите, ученым написать программу, определяющую, сколько красных и сколько зеленых клеток будет в получившемся дереве через D дней.

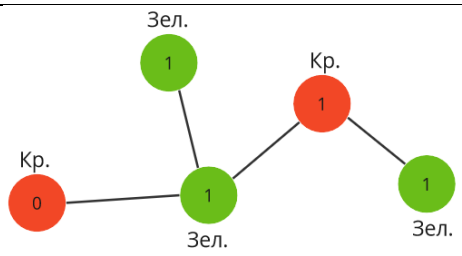
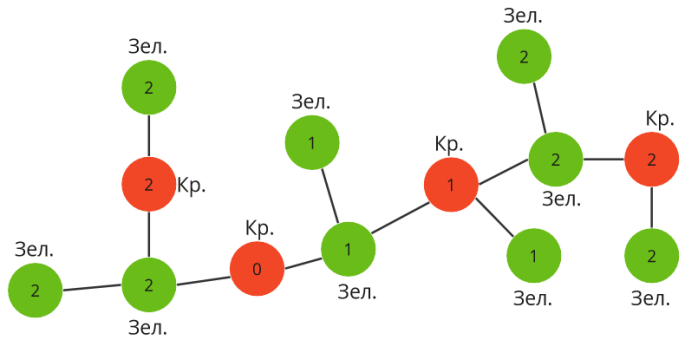
Получите ответ по модулю 1 000 000 007.

Входные данные. Три целых числа разделенных пробелом N, R и D, где N – количество клеток, которые появляются после первого дня. R – Количество красных клеток и D – количество дней.

Результат должен содержать два числа через пробел — количество красных и зеленых яиц за D дней, по модулю 1 000 000 007.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод: 5 2 2	Результат: 4 9
Пояснение	
После первого дня будет 2 красные клетки и 3 зеленых. Дерево может выглядеть так.	
Через два дня дерево клеток будет выглядеть уже так. Числа на клетке обозначают: через сколько дней она появилась.	

Исходные данные:

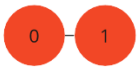
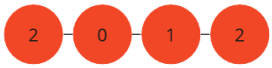
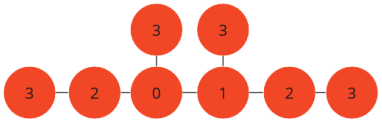
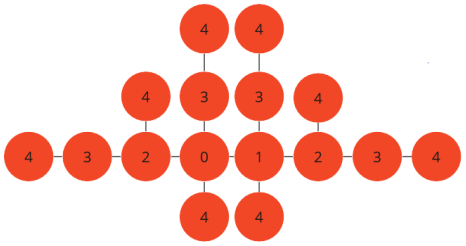
7 3 7

Решение:

Итак, у нас есть дерево с клетками, из которых R красных, а остальные (N - R) зеленые. Каждую ночь все красные клетки размножаются в новое дерево такого же размера, как и первое, а затем все красные клетки в этом новом дереве также размножаются в новое дерево и так далее.

Мы хотим найти количество красных и зеленых клеток через D дней.

Рассмотрим процесс размножения для красных клеток, если у нас каждую ночь рождается новая красная клетка.

После 1 дня		$2 = 2^1$
После 2 дня		$4 = 2^2$
После 3 дня		$8 = 2^3$
После 4 дня		$16 = 2^4$

Как можно заметит, что количество красных клеток будет равно их заданному количеству в степени равной заданному дню. Для зеленых клеток ситуация подобная. Проблема заключается в возрастании чисел и вычислении степени.

Для нахождения красных клеток через D дней, определим функцию pow(base, exp) которая используется для быстрого возведения числа base в степень exp по модулю MOD. Она также использует рекурсию, чтобы разбить вычисление степени на более мелкие шаги. В алгоритме используется взятие остатка от деления (mod), чтобы эффективно работать с большими числами и избегать переполнения.

Для нахождения зеленых клеток используется подобный рекурсивный алгоритм.

От участников требуется показать навыки реализации алгоритма быстрого возведения в степень.

Реализация на C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <cstdio>
#include <iostream>
#define MOD 1000000007


using namespace std;

typedef long long ll;


ll count(ll base, int d) {
    if (d == 0)
        return 1;
    if (d % 2 == 0)
        return (1 + base * count(base, d - 1)) % MOD;
    return (count((base * base) % MOD, d / 2) * (1 + base)) % MOD;
}

ll pow(ll base, int exp) {
    if (exp == 0)
        return 1;
    if (exp & 1)
        return (base * pow(base, exp - 1)) % MOD;
    return pow((base * base) % MOD, exp / 2);
}

int main() {
    ll N, R, D;
    cin >> N >> R >> D;
    ll totalR = pow(R, D);
    ll totalZ = (count(R, D - 1) * (N - R)) % MOD;
    printf("%lld %lld\n", totalR, totalZ);
```

```
    return 0;
}
```

Реализация на Python

```
MOD = 1000000007

def count(base, d):
    if d == 0:
        return 1
    if d % 2 == 0:
        return (1 + base * count(base, d - 1)) % MOD
    return (count((base * base) % MOD, d // 2) * (1 + base)) % MOD

def pow(base, exp):
    if exp == 0:
        return 1
    if exp & 1:
        return (base * pow(base, exp - 1)) % MOD
    return pow((base * base) % MOD, exp // 2)

N, R, D = map(int, input().split())
totalR = pow(R, D)
totalZ = (count(R, D - 1) * (N - R)) % MOD
print(totalR, totalZ)
```

Демонстрация C++

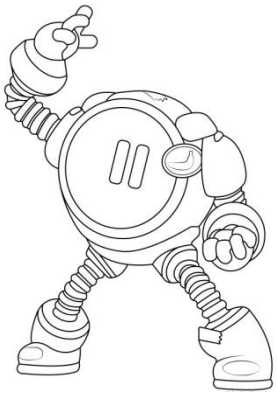
```cpp
#define MOD 1000000007


using namespace std;

typedef long long ll;


ll count(ll base, int d) {
    if (d == 0)
        return 1;
    if (d % 2 == 0)
        return (1 + base * count(base, d - 1)) % MOD;
    return (count((base * base) % MOD, d / 2) * (1 + base)) % MOD;
}

ll pow(ll base, int exp) {
    if (exp == 0)
        return 1;
    if (exp & 1)
        return (base * pow(base, exp - 1)) % MOD;
    return pow((base * base) % MOD, exp / 2);
}

int main() {
```

```
7 3 7
2187 4372
```

Ответ: 2187 4372

Задача 6.

СОРЕВНОВАНИЯ ПО РОБОТОТЕХНИКЕ



В соревнование участвуют роботы квадратной формы, которые могут двигаться только в четырех направлениях параллельно сторонам трассы. На одном из этапов соревнований робот располагается на стартовой дорожке слева. Он должен заехать на трассу с левого края, пересечь лабиринт (не обязательно по кратчайшему пути) и выйти через правый край, чтобы попасть на финишную дорожку. Победителем становится участник с самым большим роботом (т.е. роботом с самой большой квадратной формой), который выполнит задание.

Организаторы хотят проверить трассу непосредственно перед соревнованием и выяснить размер роботов, которых нужно будет построить для соревнований. Напишите программу, которая, зная планировку трассы, вычисляла, какой должна быть максимальная длина стороны робота.

Входные данные: в первой строке, указаны ширина и длина трассы. Следующие строки n_i содержат m_i символов, описывающих *i-ю* дорожку:

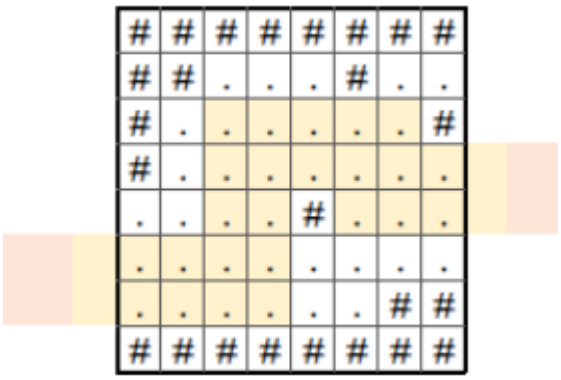
- « . » обозначает пустую ячейку, по которой может двигаться робот,
- « # » обозначает занятую ячейку - стену.

Верхний и нижний ряды всех дорожек состоят только из занятых ячеек.

Результат должен содержать одно целое число – максимальную длину стороны робота, такую, что робот такого размера может преодолеть трассу.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод:	Результат:	
8 8 ######## ##...#.. #......# #....... ....#... ....## ########	2	 Пример движения робота 2×2: Более крупные роботы не преодолеют этот путь.

Исходные данные:

```
14 23
######################
.#.....#........#....#.
.#.............#.......
....#......#.....#.....
.....#......##....#....
#.##...#..........#....
..#.....#....#......#..
#.##....##...#.........
......................
..#.............#.....
..#...................
........#............##
#.......#..#..###....#.
######################
```

Решение:

Поиск максимального размера робота для прохода по трассе использует подход основанный на бинарном поиске и поиске в глубину (DFS).

В начале программы определяются несколько глобальных переменных, таких как размеры карты **n** и **m**, размер квадрата a, массивы для отслеживания пути **track**, массив для хранения значений префиксных сумм **take_dp** и массив для отслеживания посещенных клеток **visited**.

Функция **free_square** проверяет, можно ли разместить квадрат с верхним левым углом в клетке (i, j) и размером a. Она также проверяет, что внутри этого квадрата нет препятствий.

Функция **dfs** выполняет поиск в глубину для обхода клеток карты. Она проверяет соседние клетки и вызывает себя рекурсивно, если они доступны.

Функция **binary_search** используется для выполнения бинарного поиска максимального размера квадрата. Она вызывает функцию **dfs** для обхода клеток карты и проверяет, можно ли достичь правой границы с использованием найденного размера квадрата.

В функции **main** происходит чтение входных данных, инициализация массивов **track** и таблицы префиксных сумм **take_dp**, которая показывает, сколько занятых клеток в прямоугольнике с углами (0,0) и (i,j).

После заполнения таблицы **take_dp** для каждой клетки трассы вычисляется значение, на основе которого определяется, является ли клетка доступной. Когда все доступные клетки найдены, выполняется поиск в глубину по доступным клеткам начиная с клеток всего левого столбца. Если поиск в глубину достигает клеток правого столбца, то размер стороны квадрата подходит для работа.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>

using namespace std;

int n, m, a;
char Track[502][502];
int TakeDp[502][502];
bool Visited[502][502];

// В прямоугольнике с углами (I, J) и (I+a-1, J+a-1) нет занятой ячейки '#'
bool freeSquare(int I, int J) {
  if (I + (a - 1) > n) {
    return false;
  }
  int I2 = I + a - 1;
  int J2 = min(J + a - 1, m);
  if (TakeDp[I2][J2] - TakeDp[I - 1][J2] - TakeDp[I2][J - 1] + TakeDp[I - 1][J - 1] > 0) {
    return false;
  }
```

```
    return true;
}

void dfs(int I, int J) {
    if (Visited[I][J + 1] == 0 && freeSquare(I, J + 1) == true) {
        Visited[I][J + 1] = 1;
        dfs(I, J + 1);
    }
    if (Visited[I][J - 1] == 0 && freeSquare(I, J - 1) == true) {
        Visited[I][J - 1] = 1;
        dfs(I, J - 1);
    }
    if (Visited[I - 1][J] == 0 && freeSquare(I - 1, J) == true) {
        Visited[I - 1][J] = 1;
        dfs(I - 1, J);
    }
    if (Visited[I + 1][J] == 0 && freeSquare(I + 1, J) == true) {
        Visited[I + 1][J] = 1;
        dfs(I + 1, J);
    }
}

bool binarySearch() {
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, 1) == true && Visited[i][1] == 0) {
            Visited[i][1] = 1;
            dfs(i, 1);
        }
    }
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, m) == true && Visited[i][m] == 1) {
            return true;
        }
    }
```

```
    return false;
}

int main() {
    int T;
    cin >> T;
    for (int t = 0; t < T; t++) {
        cin >> n >> m;

        for (int i = 0; i <= n + 1; i++) {
            for (int j = 0; j <= m + 1; j++) {
                Track[i][j] = '.';  // Вокруг дорожки добавляется рамка из свободных ячеек («.»)
                TakeDp[i][j] = 0;
                Visited[i][j] = 1;
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                cin >> Track[i][j];
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                TakeDp[i][j] = TakeDp[i - 1][j] + TakeDp[i][j - 1] - TakeDp[i - 1][j - 1];
                if (Track[i][j] == '#') {
                    TakeDp[i][j]++;
                }
            }
        }

        int left = 0, right = n, middle;
        while (right - left > 1) {
```

```
        middle = (left + right) / 2;
        a = middle;

        for (int i = 1; i <= n; i++) {
          for (int j = 1; j <= m; j++) {
            Visited[i][j] = 0;
          }
        }

        if (binarySearch()) {
          left = middle;
        }
        else {
          right = middle;
        }
      }

      cout << left << endl;
    }
    return 0;
}
```

Реализация на Python

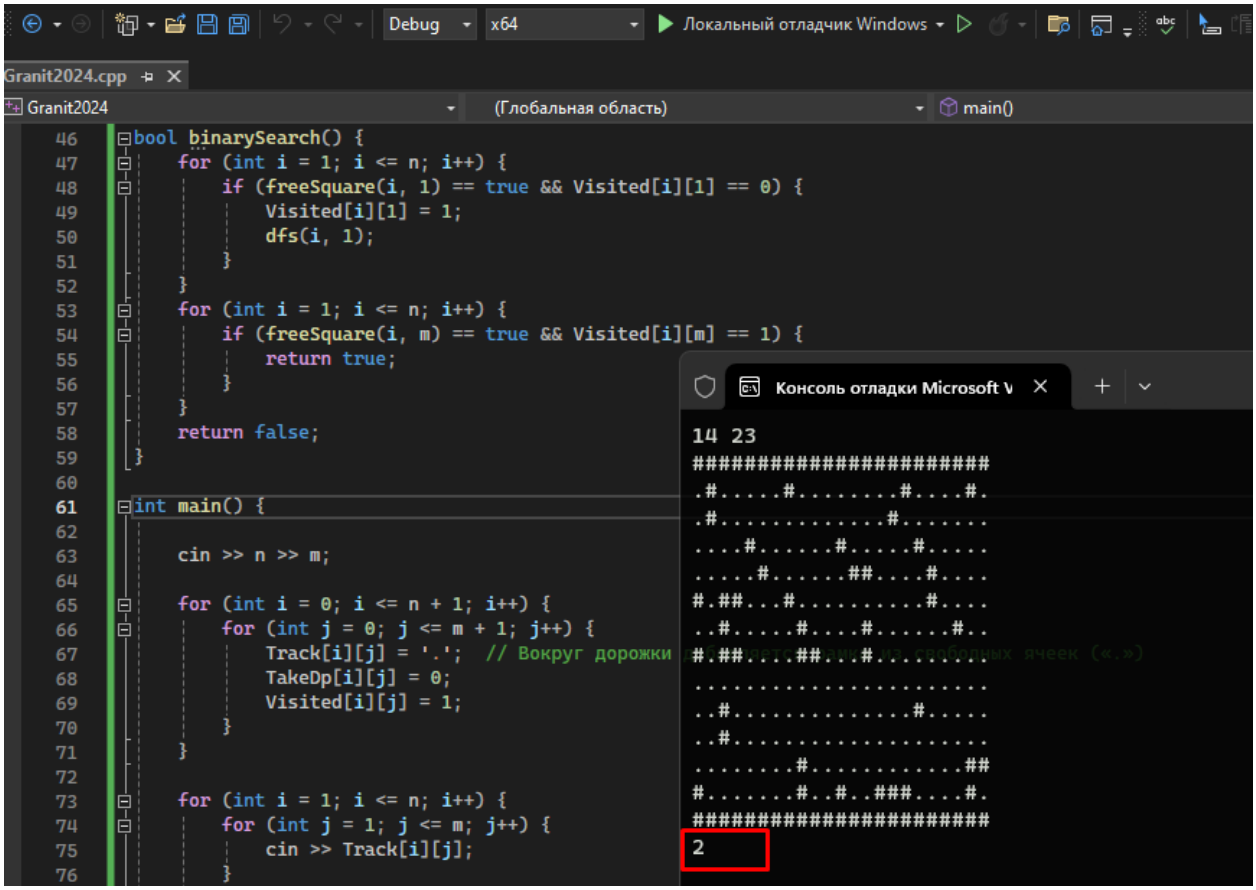
```
n, m, a = 0, 0, 0
track = [['' for _ in range(502)] for _ in range(502)]
take_dp = [[0 for _ in range(502)] for _ in range(502)]
visited = [[False for _ in range(502)] for _ in range(502)]

def free_square(i: int, j: int) -> bool:
    global n, m, a, take_dp
    if i + (a - 1) > n:
        return False
    i2 = i + a - 1
```

```
    j2 = min(j + a - 1, m)
    if take_dp[i2][j2] - take_dp[i - 1][j2] - take_dp[i2][j - 1] + take_dp[i - 1][j - 1] > 0:
        return False
    return True

def dfs(i: int, j: int) -> None:
    global visited
    if j + 1 <= m and not visited[i][j + 1] and free_square(i, j + 1):
        visited[i][j + 1] = True
        dfs(i, j + 1)
    if j - 1 >= 1 and not visited[i][j - 1] and free_square(i, j - 1):
        visited[i][j - 1] = True
        dfs(i, j - 1)
    if i - 1 >= 1 and not visited[i - 1][j] and free_square(i - 1, j):
        visited[i - 1][j] = True
        dfs(i - 1, j)
    if i + 1 <= n and not visited[i + 1][j] and free_square(i + 1, j):
        visited[i + 1][j] = True
        dfs(i + 1, j)

def binary_search() -> bool:
    global n, m, a, visited
    for i in range(1, n + 1):
        if free_square(i, 1) and not visited[i][1]:
            visited[i][1] = True
            dfs(i, 1)
    for i in range(1, n + 1):
        if free_square(i, m) and visited[i][m]:
            return True
    return False

def main():
    global n, m, a, track, take_dp, visited
    n, m = map(int, input().split())
```

```
for i in range(n + 2):
    for j in range(m + 2):
        track[i][j] = '.'
        take_dp[i][j] = 0
        visited[i][j] = False

for i in range(1, n + 1):
    row = input()
    for j in range(1, m + 1):
        track[i][j] = row[j - 1]

for i in range(1, n + 1):
    for j in range(1, m + 1):
        take_dp[i][j] = take_dp[i - 1][j] + take_dp[i][j - 1] - take_dp[i - 1][j - 1]
        if track[i][j] == '#':
            take_dp[i][j] += 1

left, right = 0, n
while right - left > 1:
    middle = (left + right) // 2
    a = middle

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            visited[i][j] = False

    if binary_search():
        left = middle
    else:
        right = middle

print(left)
```

```
if __name__ == "__main__":
    main()
```

Демонстрация на C++



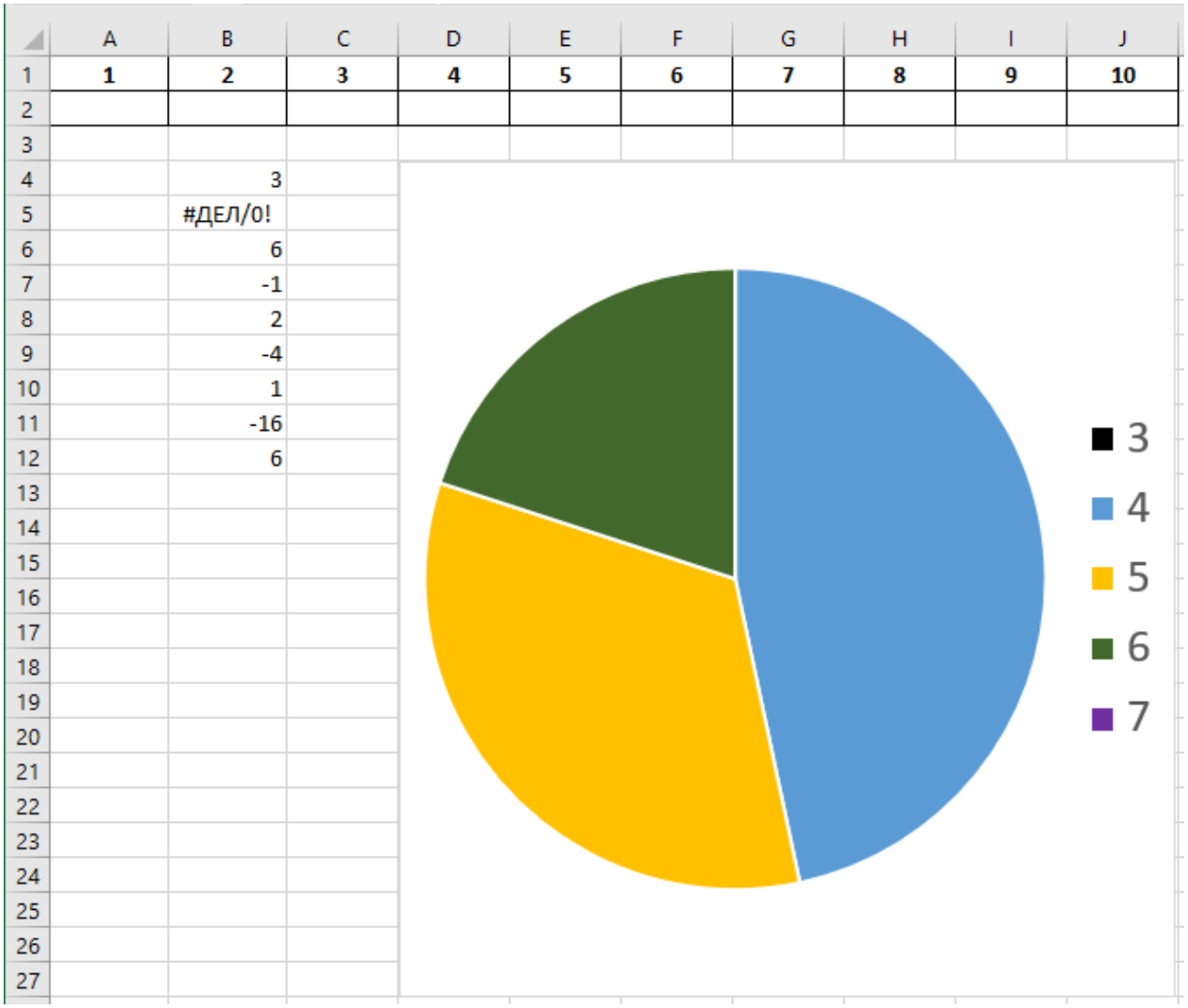
Ответ: 2

ИФНОРМАТИКА

ВАРИАНТ 6

Задача 1 (10 баллов)

В каждой из ячеек A2:J2 находится целое число. Определите их значения, используя круговую диаграмму и значения формул в блоке B4:B12. В формулах используются только ссылки на ячейки из диапазона A2:J2.



Задача усложняется тем, что отображение самих формул изменилось после копирования их в блок A5:A13.

	A
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")
6	=A3*D$2/($F3-G3)
7	=$F$2+F$2+$H3
8	=#ССЫЛКА!-E$2+#ССЫЛКА!
9	=ЕСЛИ(A3+D3;1;2)
10	=$A3+A3
11	=D3+C3+H3
12	=D3+C3*H3
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3

Решение:

1. Восстановим формулы в блоке B4:B12 исходя из формул блока A5:A13. Для получения формулы из ячейки A8 «=#ССЫЛКА!-E$2+#ССЫЛКА!» вспомним, что #ССЫЛКА! – это ссылка на ячейку, вышедшую за диапазон листа. Так как при копировании произошло смещение ссылок на 1 позицию вниз и 1 позицию влево, а также учитывая, что по условию задачи, в формулах используются только ссылки на ячейки из диапазона A2:J2, единственной ссылкой на ячейку способной выйти за диапазон листа является A2.

	A	B
4		=СЧЁТЕСЛИ($A$2:$J2;"<0")
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")	=B2*E$2/($F2-H2)
6	=A3*D$2/($F3-G3)	=$F$2+G$2+$H2
7	=$F$2+F$2+$H3	=A2-F$2+A2
8	=#ССЫЛКА!-E$2+#ССЫЛКА!	=ЕСЛИ(B2+E2;1;2)
9	=ЕСЛИ(A3+D3;1;2)	=$A2+B2
10	=$A3+A3	=E2+D2+I2
11	=D3+C3+H3	=E2+D2*I2
12	=D3+C3*H3	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3	
14		

2. По круговой диаграмме, которая, в соответствии с легендой, отображает значения категорий 3,4,5,6 и 7, видно, что значения 3 и 7 равны 0, т.е. C2=0и G2=0. Замечание: на круговой диаграмме отрицательные значения отображаются как положительные.

3. Значение формулы «=СЧЁТЕСЛИ($A$2:$J2;"<0")» в ячейке B4 равно 3, следовательно в искомом диапазоне имеется 3 отрицательных числа.

4. Значение формулы «=B2*E$2/($F2-H2)» в ячейке B5 равно значению «#ДЕЛ/0!», это означает, что ячейки F2 и H2 равны.

5. Значение формулы «=$F$2+G$2+$H2» в ячейке B6 равно 6, следовательно F2+H2 = 6-G2 = 6–0 = 6, а так как они равны, то F2 = 3, H2 = 3.

6. Значение формулы «=A2-F$2+A$2» в ячейке B7 равно -1, следовательно -1 = A2-3+A2. Откуда A2 = 2/2 = 1.

7. Значение формулы «=ЕСЛИ(B2+E2;1;2» в ячейке B8 равно 2, следовательно условие в виде суммы «B2+E2» не выполнено, т.е. B2+E2 = 0. Делаем вывод, что значения в ячейках B2 и E2 одинаковые по модулю и различные по знаку.

8. Значение формулы «=$A2+B2» в ячейке B9 равно -4, следовательно B2 = -4-A2 = -4-1 = -5, а в соответствии с 7 пунктом E2 = 5.

9. Значение формулы в B10 «=E2+D2+I2» равно 1, а значение формулы в B11 «=E2+D2*I2» равно -16. Получаем систему уравнений $\begin{cases} E2 + D2 + I2 = 1 \\ E2 + D2 * I2 = -16 \end{cases} => \begin{cases} 5 + D2 + I2 = 1 \\ 5 + D2 * I2 = -16 \end{cases} => \begin{cases} D2 + I2 = -4 \\ D2 * I2 = -21 \end{cases}$. Решая систему уравнений, находим D2 = -7 и I2 = 3.

10. Значение формулы «=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2» в ячейке B12 равно 6, следовательно J2 = ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0) -6 = ОКРУГЛВВЕРХ (0,333333;0) -6 = 1-6 = -5

Ответ:

	A	B	C	D	E	F	G	H	I	J
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	1	-5	0	-7	5	3	0	3	3	-5

Задача 2 (10 баллов)

На столе лежит две кучки драгоценных камней: в одной 13, в другой 9. Гномы ходят по очереди. За один ход можно взять один камень из одной из кучек (по выбору гнома) или взять по одному камню из двух сразу. Кто не может сделать ход (камней не осталось), проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней.

Решение:

Необходимо использовать стратегию, беря такое количество камней, чтобы в каждой кучке оставалось чётное количество камней. Позиции, где число камней в каждой кучке чётное, будут проигрышными (для того, кто в них оказался), а где хотя бы одна

кучка с нечётным количеством камней — выигрышными. В нашем случае, когда в игре в одной 13, в другой 9 камней, выигрывает первый игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось чётное количество в каждой кучке.

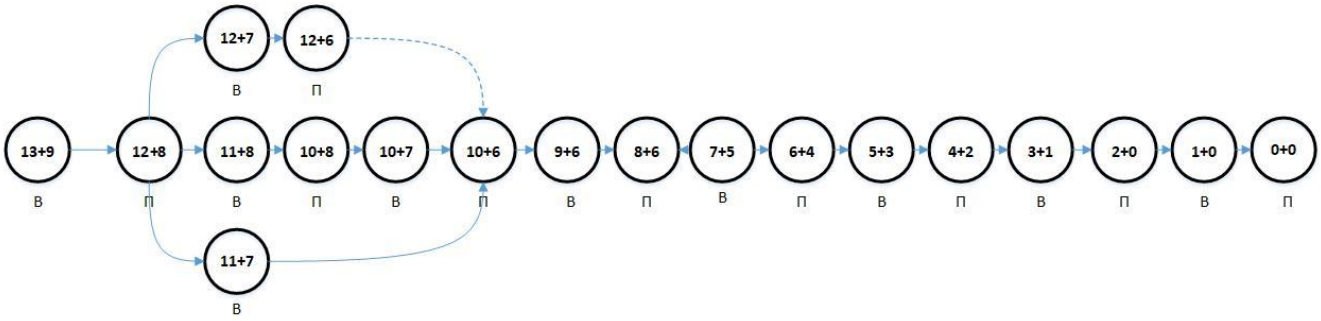
Приведём стратегию выигрывающего гнома:

Ход_1: Игрок_1 (игрок делающий ход первым) берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней (в данном случае игрок 1 возьмёт по 1 камню из каждой кучки).

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет 1 камень из любой кучки или по 1 камню из каждой кучки (в данном случае камней может остаться 11 и 7, 11 и 8, 12 и 7).

Ход_3: Игрок_1 берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней. И так далее до Победа Игрок_1. Проиграть невозможно.

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Алгоритм для Игрок_1 (обобщенный):

Если M нечётное N чётное

Шаг 1 Игрок _1 взял m=1 камень, так чтобы (M-m) кратно 2

Если Остаток = 0 – значит СТОП_Выиграл.

Если N нечётное M чётное

Шаг 1 Игрок _1 взял n=1 камень, так чтобы (N-n) кратно 2

Если N нечётное M нечётное

Шаг 1 Игрок_1 взял m=1 и n=1 камни, так чтобы (M-m) и (N-n) кратно 2

Если Остаток = 0 – значит СТОП_Выиграл.

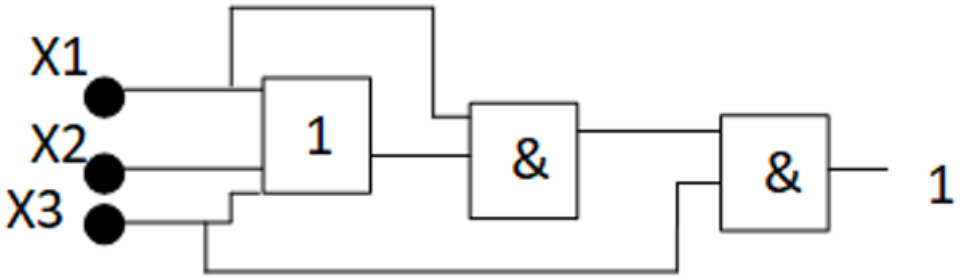
Шаг 2 Игрок _2 взял m=1 или n=1 или (m=1 и n=1) камней

Алгоритм подходит в общем случае, для M и N камней, если нечётно или M, или N, или оба вместе.

Задача 3 (10 баллов)

Дана логическая схема. Найти все наборы, при которых выражение, записанное в схеме, принимает значение истина. Подставить i-1 наборы значений из таблицы истинности в логическое выражение для определения, чему равно результирующее выражение. В ответе указать, чему будет равен результирующий столбец.

Логическая схема:



Выражение:

$x1 \to \bar{x}2 \,\&\, x3 \oplus x1$

Где [&] - это конъюнкция

[1] - это исключающее или

[V] - это дизъюнкция

Решение:

Каждый i-ый набор -это тот, при котором выражение, записанное с помощью логической схемы дает значение, указанное по условию задачи (истину). В условии сказано подставить i-1 наборы, это значит, что необходимо подставить 1 и 5 наборы, т.к. 2 и 6 дали в результирующем столбце «истину». Первое действие задумано, как X1 $\oplus$ x2 $\oplus$ x3 последовательно: X1 $\oplus$ x2, а следующее, то результат этого действия и XOR x3

x1	x2	x3	X1⊕x2	X1 ⊕ x2 ⊕x3	x1& X1⊕ x2⊕x3	x1& X1⊕x2⊕x3&x3
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	1	0	0
0	1	1	1	0	0	0
1	0	0	1	1	1	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	1	0	1	1	1

$x1 \to \bar{x}2 \,\&\, x3 \oplus x1$

x1	x2	x3	$\bar{x}\,2$	$\bar{x}2 \,\&\, x3$	$x1 \to \bar{x}2 \,\&\, x3$	$x1 \to \bar{x}2 \,\&\, x3 \oplus x1$
1	1	0	0	0	0	1

Ответ. 1

Задача 4.

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

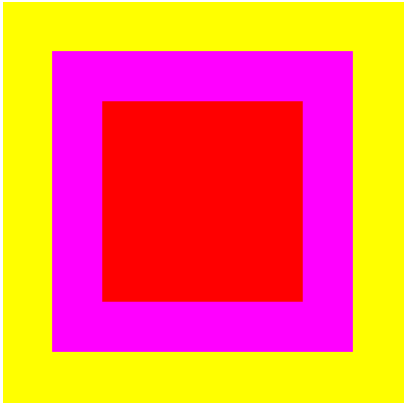


Рисунок 1

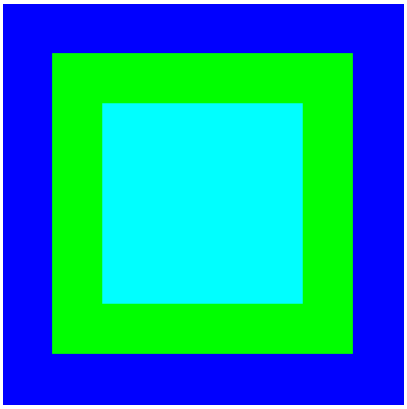


Рисунок 2

Решение:

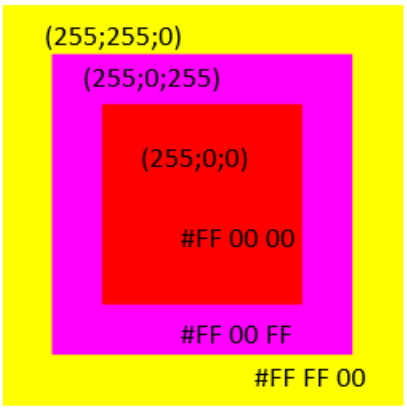
Как мы видим, в обработанном изображении красный цвет стал голубым. Получается, что после обработки, яркости красной составляющей приняли нулевые значения, а яркости зелёной и синей приняли значение 255. Делаем предположение, что была произведена инверсия изображения по трем каналам RGB, то есть алгоритм обработки выглядит следующим образом:

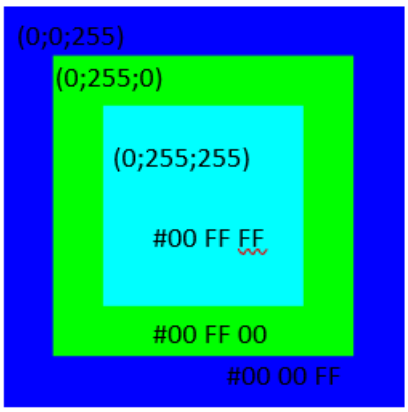
$$Y_R = 255 - Y_R$$
$$Y_G = 255 - Y_G$$
$$Y_B = 255 - Y_B$$

Жёлтый цвет стал синим, то есть яркости красной и зелёной составляющих приняли нулевые значения, а яркости синей составляющей приняли значение 255.

Лиловый цвет стал зелёным, то есть яркости красной и синей составляющих приняли нулевые значения, а яркости зелёной составляющей приняли значение 255.

Далее укажем на рисунках значения яркости в RGB и HEX формате

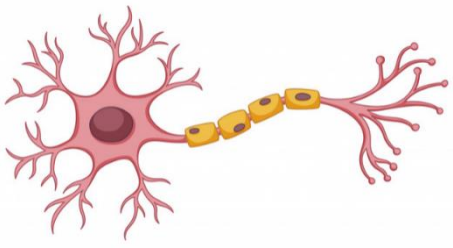




Задача 5.

ОТКРЫТИЯ АРКТИКИ

Однажды после очередной экспедиции в Арктику ученые университета привезли образец льдины, добытый из глубин арктических пластов. При исследовании его в лаборатории химии они обнаружили новые клетки красного цвета.



Клетка сама по себе ничего не делала, и ученые уже расстроились, но однажды очень жарким летом, сломались кондиционеры в лаборатории, и клетка проснулась. За ночь клетка размножилась и стала похожа на дерево, состоящее из соединительных каналов и N прикрепленных к ним таких же клеток, из которых R красные, а остальные зеленые. Позже, выяснилось, что новые клетки тоже размножаются, и каждую

последующую ночь все красные клетки вырастают в другое дерево, подобное тому, которое выросло из первой клетки в первую ночь.

Помогите, ученым написать программу, определяющую, сколько красных и сколько зеленых клеток будет в получившемся дереве через D дней.

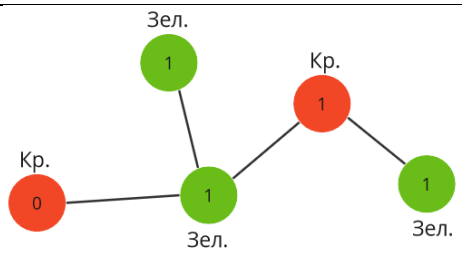
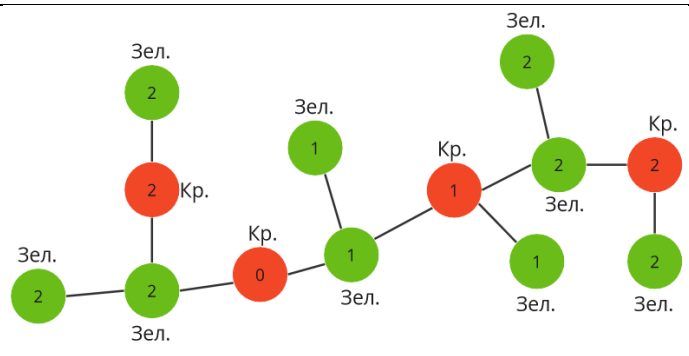
Получите ответ по модулю 1 000 000 007.

Входные данные. Три целых числа разделенных пробелом N, R и D, где N – количество клеток, которые появляются после первого дня. R – Количество красных клеток и D – количество дней.

Результат должен содержать два числа через пробел — количество красных и зеленых яиц за D дней, по модулю 1 000 000 007.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

<table>
<tr><td>Ввод:
5 2 2</td><td>Результат:
4 9</td></tr>
<tr><td colspan="2">Пояснение</td></tr>
<tr><td>После первого дня будет 2 красные клетки и 3 зеленых. Дерево может выглядеть так.</td><td> 
</td></tr>
<tr><td>Через два дня дерево клеток будет выглядеть уже так. Числа на клетке обозначают: через сколько дней она появилась.</td><td> 
</td></tr>
</table>

Исходные данные:

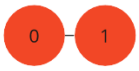
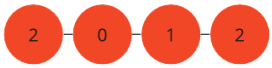
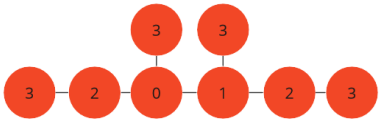
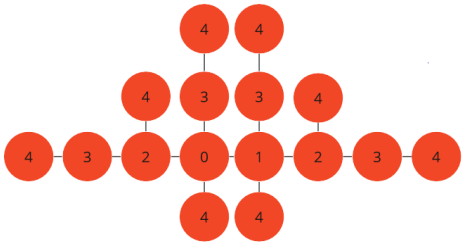
8 4 5

Решение:

Итак, у нас есть дерево с клетками, из которых R красных, а остальные (N - R) зеленые. Каждую ночь все красные клетки размножаются в новое дерево такого же размера, как и первое, а затем все красные клетки в этом новом дереве также размножаются в новое дерево и так далее.

Мы хотим найти количество красных и зеленых клеток через D дней.

Рассмотрим процесс размножения для красных клеток, если у нас каждую ночь рождается новая красная клетка.

После 1 дня		$2 = 2^1$
После 2 дня		$4 = 2^2$
После 3 дня		$8 = 2^3$
После 4 дня		$16 = 2^4$

Как можно заметит, что количество красных клеток будет равно их заданному количеству в степени равной заданному дню. Для зеленых клеток ситуация подобная. Проблема заключается в возрастании чисел и вычислении степени.

Для нахождения красных клеток через D дней, определим функцию pow(base, exp) которая используется для быстрого возведения числа base в степень exp по модулю MOD. Она также использует рекурсию, чтобы разбить вычисление степени на более мелкие шаги. В алгоритме используется взятие остатка от деления (mod), чтобы эффективно работать с большими числами и избегать переполнения.

Для нахождения зеленых клеток используется подобный рекурсивный алгоритм.

От участников требуется показать навыки реализации алгоритма быстрого возведения в степень.

Реализация на C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <cstdio>
#include <iostream>
#define MOD 1000000007


using namespace std;

typedef long long ll;


ll count(ll base, int d) {
    if (d == 0)
        return 1;
    if (d % 2 == 0)
        return (1 + base * count(base, d - 1)) % MOD;
    return (count((base * base) % MOD, d / 2) * (1 + base)) % MOD;
}

ll pow(ll base, int exp) {
    if (exp == 0)
        return 1;
    if (exp & 1)
        return (base * pow(base, exp - 1)) % MOD;
    return pow((base * base) % MOD, exp / 2);
}

int main() {
    ll N, R, D;
    cin >> N >> R >> D;
    ll totalR = pow(R, D);
    ll totalZ = (count(R, D - 1) * (N - R)) % MOD;
    printf("%lld %lld\n", totalR, totalZ);
```

```
    return 0;
}
```

Реализация на Python

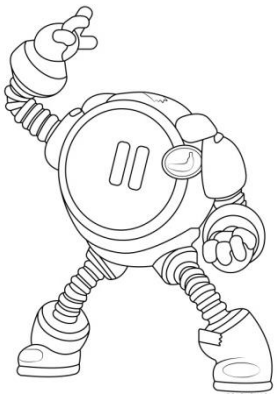
```
MOD = 1000000007

def count(base, d):
    if d == 0:
        return 1
    if d % 2 == 0:
        return (1 + base * count(base, d - 1)) % MOD
    return (count((base * base) % MOD, d // 2) * (1 + base)) % MOD

def pow(base, exp):
    if exp == 0:
        return 1
    if exp & 1:
        return (base * pow(base, exp - 1)) % MOD
    return pow((base * base) % MOD, exp // 2)

N, R, D = map(int, input().split())
totalR = pow(R, D)
totalZ = (count(R, D - 1) * (N - R)) % MOD
print(totalR, totalZ)
```

Демонстрация C++

Ответ: 1024 1364

Задача 6.

СОРЕВНОВАНИЯ ПО РОБОТОТЕХНИКЕ



В соревнование участвуют роботы квадратной формы, которые могут двигаться только в четырех направлениях параллельно сторонам трассы. На одном из этапов соревнований робот располагается на стартовой дорожке слева. Он должен заехать на трассу с левого края, пересечь лабиринт (не обязательно по кратчайшему пути) и выйти через правый край, чтобы попасть на финишную дорожку. Победителем становится участник с самым большим роботом (т.е. роботом с самой большой квадратной формой), который выполнит задание.

Организаторы хотят проверить трассу непосредственно перед соревнованием и выяснить размер роботов, которых нужно будет построить для соревнований. Напишите программу, которая, зная планировку трассы, вычисляла, какой должна быть максимальная длина стороны робота.

Входные данные: в первой строке, указаны ширина и длина трассы. Следующие строки n_i содержат m_i символов, описывающих *i-ю* дорожку:

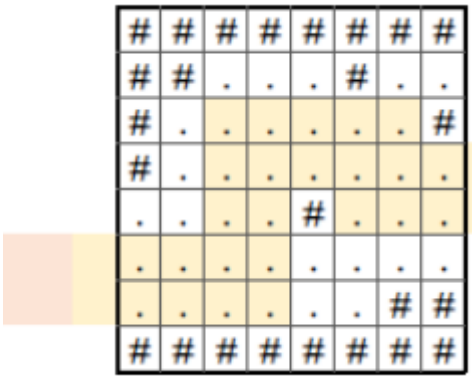
- « . » обозначает пустую ячейку, по которой может двигаться робот,
- « # » обозначает занятую ячейку - стену.

Верхний и нижний ряды всех дорожек состоят только из занятых ячеек.

Результат должен содержать одно целое число – максимальную длину стороны робота, такую, что робот такого размера может преодолеть трассу.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод:	Результат:	
8 8 ######## ##...#.. #......# #....... ....#... ....## ########	2	 Пример движения робота 2×2: Более крупные роботы не преодолеют этот путь.

Исходные данные:

```
13 23
######################
###.#.......#.....##.#.
.............#......#..
...#.....#.#.#.##......
..........#.#.#....#...
##....#.#..............
...#.#......#...##.#...
....#.......#....#..#..
..#.....#....#.........
........##...###....###
.......#.#...........#.
........#..##....####..
######################
```

Решение:

Поиск максимального размера робота для прохода по трассе использует подход основанный на бинарном поиске и поиске в глубину (DFS).

В начале программы определяются несколько глобальных переменных, таких как размеры карты **n** и **m**, размер квадрата a, массивы для отслеживания пути **track**, массив для хранения значений префиксных сумм **take_dp** и массив для отслеживания посещенных клеток **visited**.

Функция **free_square** проверяет, можно ли разместить квадрат с верхним левым углом в клетке (i, j) и размером a. Она также проверяет, что внутри этого квадрата нет препятствий.

Функция **dfs** выполняет поиск в глубину для обхода клеток карты. Она проверяет соседние клетки и вызывает себя рекурсивно, если они доступны.

Функция **binary_search** используется для выполнения бинарного поиска максимального размера квадрата. Она вызывает функцию **dfs** для обхода клеток карты и проверяет, можно ли достичь правой границы с использованием найденного размера квадрата.

В функции **main** происходит чтение входных данных, инициализация массивов **track** и таблицы префиксных сумм **take_dp**, которая показывает, сколько занятых клеток в прямоугольнике с углами (0,0) и (i,j).

После заполнения таблицы **take_dp** для каждой клетки трассы вычисляется значение, на основе которого определяется, является ли клетка доступной. Когда все доступные клетки найдены, выполняется поиск в глубину по доступным клеткам начиная с клеток всего левого столбца. Если поиск в глубину достигает клеток правого столбца, то размер стороны квадрата подходит для работа.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>

using namespace std;

int n, m, a;
char Track[502][502];
int TakeDp[502][502];
bool Visited[502][502];

// В прямоугольнике с углами (I, J) и (I+a-1, J+a-1) нет занятой ячейки '#'
bool freeSquare(int I, int J) {
  if (I + (a - 1) > n) {
    return false;
  }
  int I2 = I + a - 1;
  int J2 = min(J + a - 1, m);
  if (TakeDp[I2][J2] - TakeDp[I - 1][J2] - TakeDp[I2][J - 1] + TakeDp[I - 1][J - 1] > 0) {
    return false;
  }
  return true;
}
```

```
void dfs(int I, int J) {
  if (Visited[I][J + 1] == 0 && freeSquare(I, J + 1) == true) {
    Visited[I][J + 1] = 1;
    dfs(I, J + 1);
  }
  if (Visited[I][J - 1] == 0 && freeSquare(I, J - 1) == true) {
    Visited[I][J - 1] = 1;
    dfs(I, J - 1);
  }
  if (Visited[I - 1][J] == 0 && freeSquare(I - 1, J) == true) {
    Visited[I - 1][J] = 1;
    dfs(I - 1, J);
  }
  if (Visited[I + 1][J] == 0 && freeSquare(I + 1, J) == true) {
    Visited[I + 1][J] = 1;
    dfs(I + 1, J);
  }
}

bool binarySearch() {
  for (int i = 1; i <= n; i++) {
    if (freeSquare(i, 1) == true && Visited[i][1] == 0) {
      Visited[i][1] = 1;
      dfs(i, 1);
    }
  }
  for (int i = 1; i <= n; i++) {
    if (freeSquare(i, m) == true && Visited[i][m] == 1) {
      return true;
    }
  }
  return false;
}
```

```
int main() {
    int T;
    cin >> T;
    for (int t = 0; t < T; t++) {
        cin >> n >> m;

        for (int i = 0; i <= n + 1; i++) {
            for (int j = 0; j <= m + 1; j++) {
                Track[i][j] = '.';  // Вокруг дорожки добавляется рамка из свободных ячеек («.»)
                TakeDp[i][j] = 0;
                Visited[i][j] = 1;
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                cin >> Track[i][j];
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                TakeDp[i][j] = TakeDp[i - 1][j] + TakeDp[i][j - 1] - TakeDp[i - 1][j - 1];
                if (Track[i][j] == '#') {
                    TakeDp[i][j]++;
                }
            }
        }

        int left = 0, right = n, middle;
        while (right - left > 1) {
            middle = (left + right) / 2;
            a = middle;
```

```
            for (int i = 1; i <= n; i++) {
                for (int j = 1; j <= m; j++) {
                    Visited[i][j] = 0;
                }
            }

            if (binarySearch()) {
                left = middle;
            }
            else {
                right = middle;
            }
        }

        cout << left << endl;
    }
    return 0;
}
```

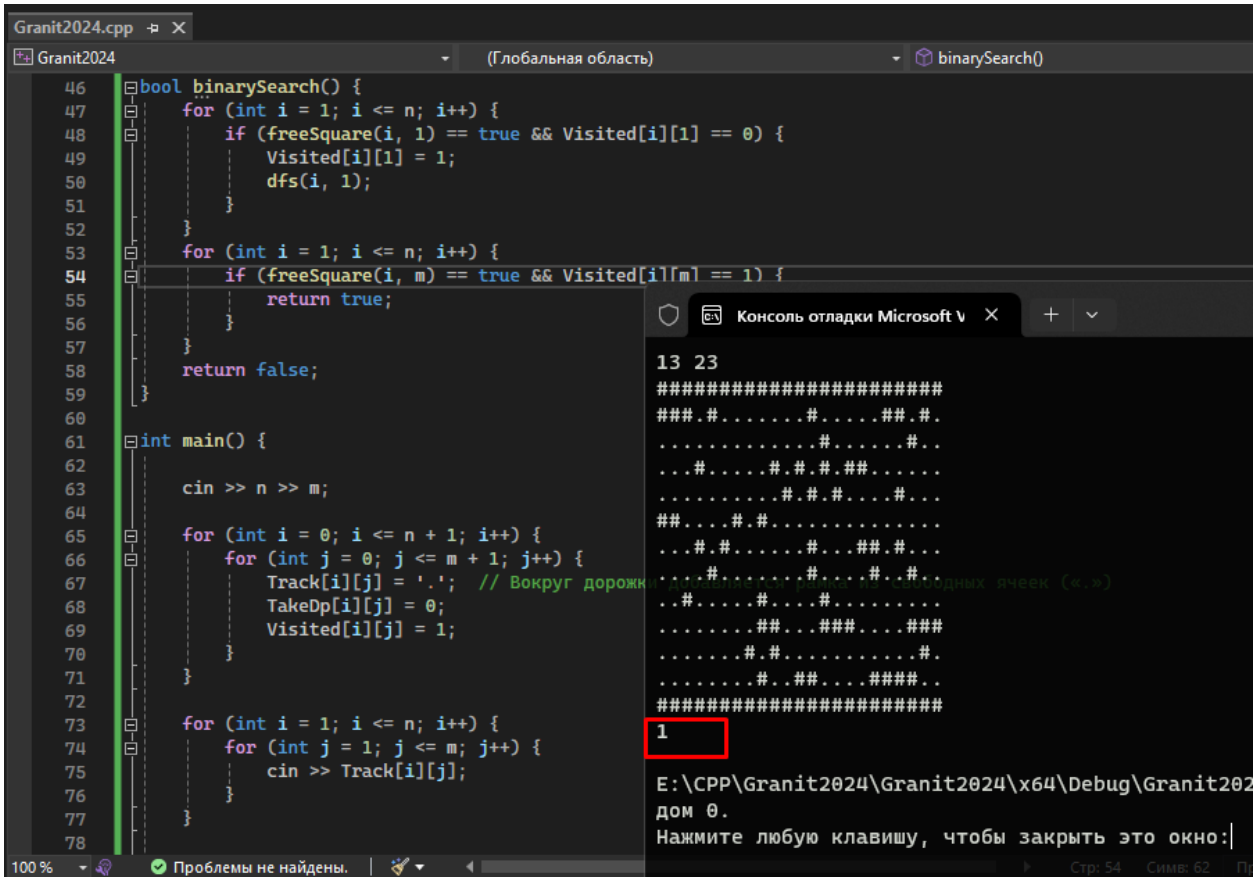
Реализация на Python

```
n, m, a = 0, 0, 0
track = [['' for _ in range(502)] for _ in range(502)]
take_dp = [[0 for _ in range(502)] for _ in range(502)]
visited = [[False for _ in range(502)] for _ in range(502)]

def free_square(i: int, j: int) -> bool:
    global n, m, a, take_dp
    if i + (a - 1) > n:
        return False
    i2 = i + a - 1
    j2 = min(j + a - 1, m)
    if take_dp[i2][j2] - take_dp[i - 1][j2] - take_dp[i2][j - 1] + take_dp[i - 1][j - 1] > 0:
        return False
```

```
    return True

def dfs(i: int, j: int) -> None:
    global visited
    if j + 1 <= m and not visited[i][j + 1] and free_square(i, j + 1):
        visited[i][j + 1] = True
        dfs(i, j + 1)
    if j - 1 >= 1 and not visited[i][j - 1] and free_square(i, j - 1):
        visited[i][j - 1] = True
        dfs(i, j - 1)
    if i - 1 >= 1 and not visited[i - 1][j] and free_square(i - 1, j):
        visited[i - 1][j] = True
        dfs(i - 1, j)
    if i + 1 <= n and not visited[i + 1][j] and free_square(i + 1, j):
        visited[i + 1][j] = True
        dfs(i + 1, j)

def binary_search() -> bool:
    global n, m, a, visited
    for i in range(1, n + 1):
        if free_square(i, 1) and not visited[i][1]:
            visited[i][1] = True
            dfs(i, 1)
    for i in range(1, n + 1):
        if free_square(i, m) and visited[i][m]:
            return True
    return False

def main():
    global n, m, a, track, take_dp, visited
    n, m = map(int, input().split())

    for i in range(n + 2):
        for j in range(m + 2):
```

```
            track[i][j] = '.'
            take_dp[i][j] = 0
            visited[i][j] = False

    for i in range(1, n + 1):
        row = input()
        for j in range(1, m + 1):
            track[i][j] = row[j - 1]

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            take_dp[i][j] = take_dp[i - 1][j] + take_dp[i][j - 1] - take_dp[i - 1][j - 1]
            if track[i][j] == '#':
                take_dp[i][j] += 1

    left, right = 0, n
    while right - left > 1:
        middle = (left + right) // 2
        a = middle

        for i in range(1, n + 1):
            for j in range(1, m + 1):
                visited[i][j] = False

        if binary_search():
            left = middle
        else:
            right = middle

    print(left)

if __name__ == "__main__":
    main()
```

Демонстрация на C++



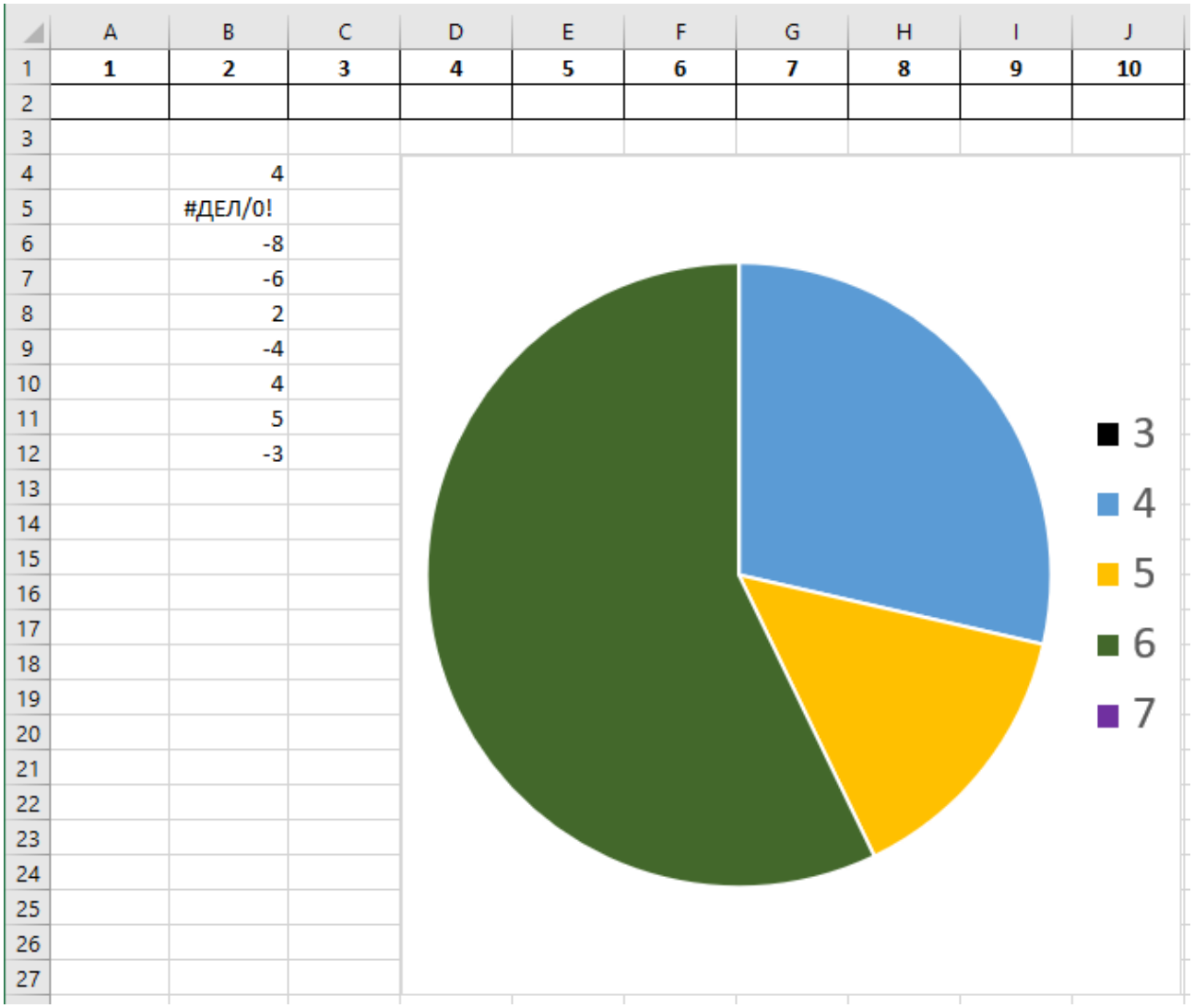
Ответ: 1

ИНФОРМАТИКА

ВАРИАНТ 7

Задача 1 (10 баллов)

В каждой из ячеек A2:J2 находится целое число. Определите их значения, используя круговую диаграмму и значения формул в блоке B4:B12. В формулах используются только ссылки на ячейки из диапазона A2:J2.



Задача усложняется тем, что отображение самих формул изменилось после копирования их в блок A5:A13.

	A
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")
6	=A3*D$2/($F3-G3)
7	=$F$2+F$2+$H3
8	=#ССЫЛКА!-E$2+#ССЫЛКА!
9	=ЕСЛИ(A3+D3;1;2)
10	=$A3+A3
11	=D3+C3+H3
12	=D3+C3*H3
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3

Решение:

1. Восстановим формулы в блоке B4:B12 исходя из формул блока A5:A13. Для получения формулы из ячейки A8 «=#ССЫЛКА!-E$2+#ССЫЛКА!» вспомним, что #ССЫЛКА! – это ссылка на ячейку, вышедшую за диапазон листа. Так как при копировании произошло смещение ссылок на 1 позицию вниз и 1 позицию влево, а также учитывая, что по условию задачи, в формулах используются только ссылки на ячейки из диапазона A2:J2, единственной ссылкой на ячейку способной выйти за диапазон листа является A2.

	A	B
4		=СЧЁТЕСЛИ($A$2:$J2;"<0")
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")	=B2*E$2/($F2-H2)
6	=A3*D$2/($F3-G3)	=$F$2+G$2+$H2
7	=$F$2+F$2+$H3	=A2-F$2+A2
8	=#ССЫЛКА!-E$2+#ССЫЛКА!	=ЕСЛИ(B2+E2;1;2)
9	=ЕСЛИ(A3+D3;1;2)	=$A2+B2
10	=$A3+A3	=E2+D2+I2
11	=D3+C3+H3	=E2+D2*I2
12	=D3+C3*H3	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3	
14		

2. По круговой диаграмме, которая, в соответствии с легендой, отображает значения категорий 3,4,5,6 и 7, видно, что значения 3 и 7 равны 0, т.е. C2=0и G2=0. Замечание: на круговой диаграмме отрицательные значения отображаются как положительные.

3. Значение формулы «=СЧЁТЕСЛИ($A$2:$J2;"<0")» в ячейке B4 равно 4, следовательно в искомом диапазоне имеется 4 отрицательных числа.

4. Значение формулы «=B2*E$2/($F2-H2)» в ячейке B5 равно значению «#ДЕЛ/0!», это означает, что ячейки F2 и H2 равны.

5. Значение формулы «=$F$2+G$2+$H2» в ячейке B6 равно -8, следовательно F2+H2 = -8-G2 = -8–0 = -8, а так как они равны, то F2 = -4, H2 = -4.

6. Значение формулы «=A2-F$2+A$2» в ячейке B7 равно -6, следовательно -6 = A2+4+A2. Откуда A2 = -10/2 = -5.

7. Значение формулы «=ЕСЛИ(B2+E2;1;2» в ячейке B8 равно 2, следовательно условие в виде суммы «B2+E2» не выполнено, т.е. B2+E2 = 0. Делаем вывод, что значения в ячейках B2 и E2 одинаковые по модулю и различные по знаку.

8. Значение формулы «=$A2+B2» в ячейке B9 равно -4, следовательно B2 = -4-A2 = -4+5 = 1, а в соответствии с 7 пунктом E2 = -1.

9. Значение формулы в B10 «=E2+D2+I2» равно 4, а значение формулы в B11 «=E2+D2*I2» равно 5. Получаем систему уравнений $\begin{cases} E2 + D2 + I2 = 4 \\ E2 + D2 * I2 = 5 \end{cases} => \begin{cases} -1 + D2 + I2 = 4 \\ -1 + D2 * I2 = 5 \end{cases} => \begin{cases} D2 + I2 = 5 \\ D2 * I2 = 6 \end{cases}$. Решая систему уравнений, находим D2 = 2 и I2 = 3.

10. Значение формулы «=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2» в ячейке B12 равно -3, следовательно J2 = ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0) +3 = ОКРУГЛВВЕРХ (-0,88889;0) +3 = -1+3 = 2.

Ответ:

	A	B	C	D	E	F	G	H	I	J
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	-5	1	0	2	-1	-4	0	-4	3	2

Задача 2 (10 баллов)

На столе лежит две кучки драгоценных камней: в одной 10, в другой 8. Гномы ходят по очереди. За один ход можно взять один камень из одной из кучек (по выбору гнома) или взять по одному камню из двух сразу. Кто не может сделать ход (камней не осталось), проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней.

Решение:

Необходимо использовать стратегию, беря такое количество камней, чтобы в каждой кучке оставалось чётное количество камней. Позиции, где число камней в каждой кучке чётное, будут проигрышными (для того, кто в них оказался), а где хотя бы одна

кучка с нечётным количеством камней — выигрышными. В нашем случае, когда в игре в одной 10, в другой 8 камней, выигрывает второй игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось чётное количество в каждой кучке.

Приведём стратегию выигрывающего гнома:

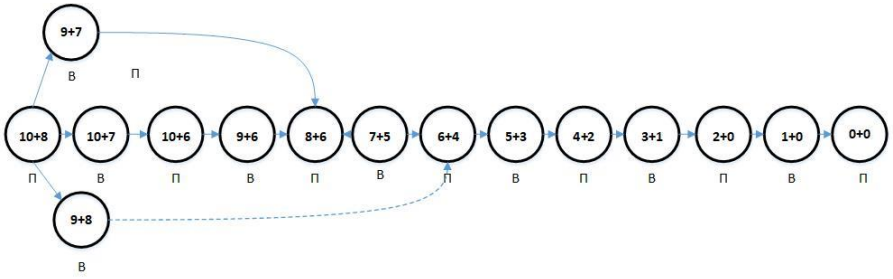
Ход_1: Игрок_1 (игрок делающий ход первым) берет 1 камень из любой кучки или по 1 камню из каждой кучки (в данном случае камней может остаться 9 и 7, 9 и 8, 10 и 7)

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней (в данном случае игрок 2 может свезти все к позиции 8 и 6, 8 и 8, 10 и 6).

Ход_3: Игрок_1 берет 1 камень из любой кучки или по 1 камню из каждой кучки.

Ход_4: Игрок_2 берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней. И так далее до Победа Игрок_2. Проиграть невозможно.

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Алгоритм для Игрок_2 (обобщенный):

Шаг 1 Игрок _1 взял m=1 или n=1 или (m=1 и n=1) камней

Если (M-m) нечётное (N-n) чётное

Шаг 2 Игрок _2 взял m=1 камень, так чтобы (M-m) кратно 2

Если Остаток = 0 – значит СТОП_Выиграл.

Если (N-n) нечётное (M-m) чётное

Шаг 2 Игрок _2 взял n=1 камень, так чтобы (N-n) кратно 2

Если (N-n) нечётное (M-m) нечётное

Шаг 2 Игрок_2 взял m=1 и n=1 камни, так чтобы (M-m) и (N-n) кратно 2

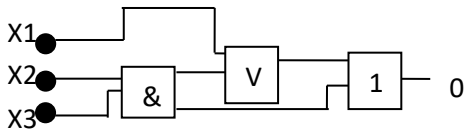
Если Остаток = 0 – значит СТОП_Выиграл.

Алгоритм подходит в общем случае, когда и M и N четны.

Задача 3

Дана логическая схема. Найти все наборы, при которых выражение, записанное в схеме, принимает значение истина. Подставить і наборы значений из таблицы истинности в логическое выражение для определения, чему равно результирующее выражение. В ответе указать, чему будет равен результирующий столбец.

Логическая схема:



Выражение:

$x2 \to x1 \leftrightarrow \bar{x}3 \oplus x1$

Где & - это конъюнкция

1 - это исключающее или

V - это дизъюнкция

Решение:

Каждый і набор -это тот, при котором выражение, записанное с помощью логической схемы дает значение, указанное по условию задачи (ложь). В условии сказано подставить і наборы, это значит, что необходимо подставить 1, 2, 3, 4, 8, т.к. дали в результирующем столбце «ложь»

x1	x2	x3	x2&x3	x1V x2&x3	x1V x2&x3 ⊕ x2&x3
0	0	0	0	0	0
0	0	1	0	0	0

0	1	0	0	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	0

$x2 \to x1 \leftrightarrow \bar{x}3 \oplus x1$

x1	x2	x3	$\bar{x}\,3$	$x2 \to x1$	$x2 \to x1 \leftrightarrow \bar{x}3$	$x2 \to x1 \leftrightarrow \bar{x}3 \oplus x1$
0	0	0	1	1	1	1
0	0	1	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
1	1	1	0	1	0	1

Ответ. 10011

Задача 4 (20 баллов)

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

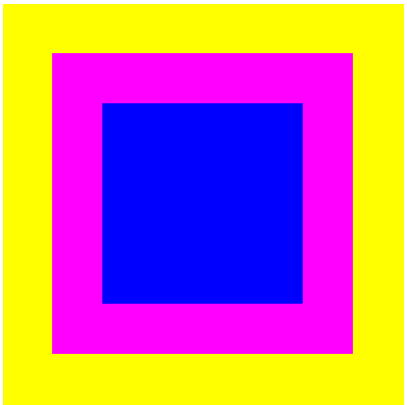


Рисунок 1

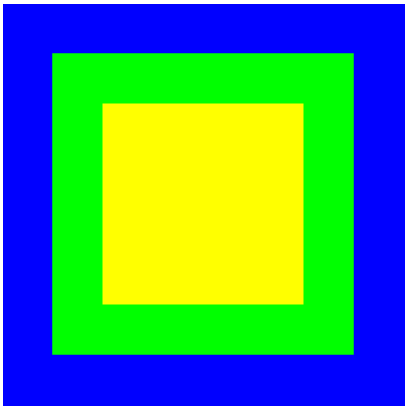


Рисунок 2

Решение:

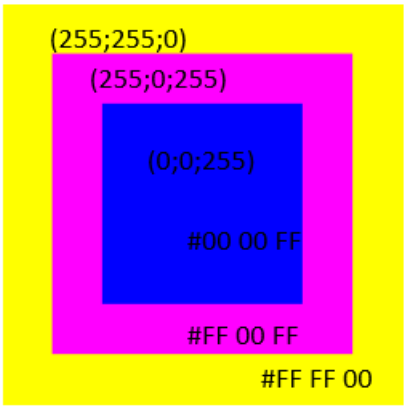
Как мы видим, в обработанном изображении синий цвет стал жёлтым. Получается, что после обработки, яркости синей составляющей приняли нулевые значения, а яркости красной и зелёной приняли значение 255. Делаем предположение, что была произведена инверсия изображения по трем каналам RGB, то есть алгоритм обработки выглядит следующим образом:

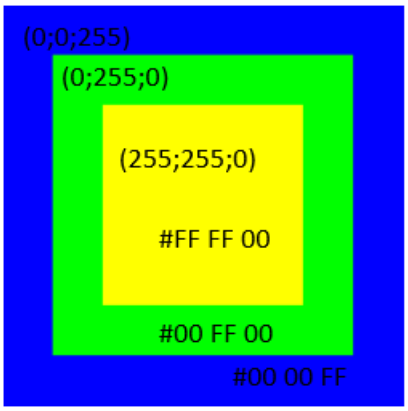
$$Y_R = 255 - Y_R$$
$$Y_G = 255 - Y_G$$
$$Y_B = 255 - Y_B$$

Жёлтый цвет стал синим, то есть яркости красной и зелёной составляющих приняли нулевые значения, а яркости синей составляющей приняли значение 255.

Лиловый цвет стал зелёным, то есть яркости красной и синей составляющих приняли нулевые значения, а яркости зелёной составляющей приняли значение 255.

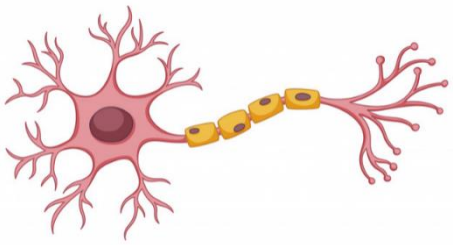
Далее укажем на рисунках значения яркости в RGB и HEX формате





Задача 5 (25 баллов)

ОТКРЫТИЯ АРКТИКИ



Однажды после очередной экспедиции в Арктику ученые университета привезли образец льдины, добытый из глубин арктических пластов. При исследовании его в лаборатории химии они обнаружили новые клетки красного цвета.

Клетка сама по себе ничего не делала, и ученые уже расстроились, но однажды очень жарким летом, сломались кондиционеры в лаборатории, и клетка проснулась. За ночь клетка размножилась и стала похожа на дерево, состоящее из соединительных каналов и N прикрепленных к ним таких же клеток, из которых R красные, а остальные зеленые. Позже, выяснилось, что новые клетки тоже размножаются, и каждую

последующую ночь все красные клетки вырастают в другое дерево, подобное тому, которое выросло из первой клетки в первую ночь.

Помогите, ученым написать программу, определяющую, сколько красных и сколько зеленых клеток будет в получившемся дереве через D дней.

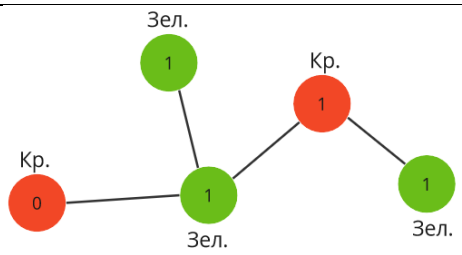
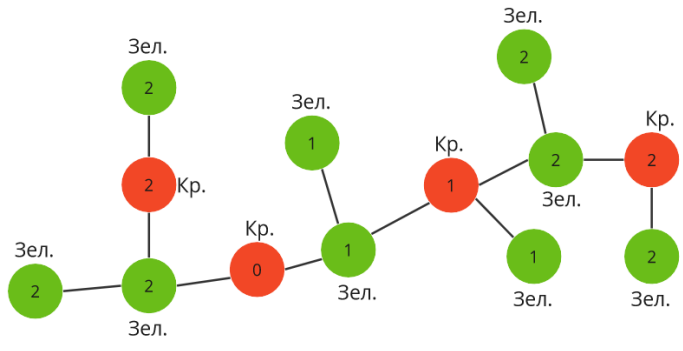
Получите ответ по модулю 1 000 000 007.

Входные данные. Три целых числа разделенных пробелом N, R и D, где N – количество клеток, которые появляются после первого дня. R – Количество красных клеток и D – количество дней.

Результат должен содержать два числа через пробел — количество красных и зеленых яиц за D дней, по модулю 1 000 000 007.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

<table>
<tr><td>Ввод:
5 2 2</td><td>Результат:
4 9</td></tr>
<tr><td colspan="2">Пояснение</td></tr>
<tr><td>После первого дня будет 2 красные клетки и 3 зеленых. Дерево может выглядеть так.</td><td> 
</td></tr>
<tr><td>Через два дня дерево клеток будет выглядеть уже так. Числа на клетке обозначают: через сколько дней она появилась.</td><td> 
</td></tr>
</table>

Исходные данные:

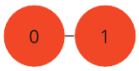
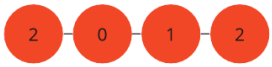
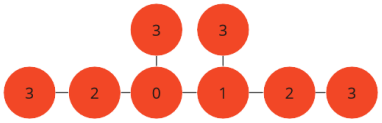
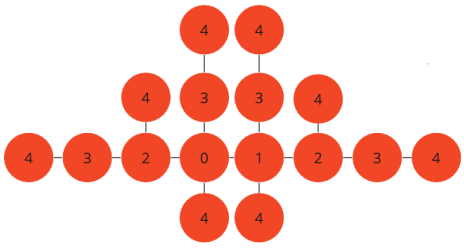
9 4 6

Решение:

Итак, у нас есть дерево с клетками, из которых R красных, а остальные (N - R) зеленые. Каждую ночь все красные клетки размножаются в новое дерево такого же размера, как и первое, а затем все красные клетки в этом новом дереве также размножаются в новое дерево и так далее.

Мы хотим найти количество красных и зеленых клеток через D дней.

Рассмотрим процесс размножения для красных клеток, если у нас каждую ночь рождается новая красная клетка.

После 1 дня		$2 = 2^1$
После 2 дня		$4 = 2^2$
После 3 дня		$8 = 2^3$
После 4 дня		$16 = 2^4$

Как можно заметит, что количество красных клеток будет равно их заданному количеству в степени равной заданному дню. Для зеленых клеток ситуация подобная. Проблема заключается в возрастании чисел и вычислении степени.

Для нахождения красных клеток через D дней, определим функцию pow(base, exp) которая используется для быстрого возведения числа base в степень exp по модулю MOD. Она также использует рекурсию, чтобы разбить вычисление степени на более мелкие шаги. В алгоритме используется взятие остатка от деления (mod), чтобы эффективно работать с большими числами и избегать переполнения.

Для нахождения зеленых клеток используется подобный рекурсивный алгоритм.

От участников требуется показать навыки реализации алгоритма быстрого возведения в степень.

Реализация на C++

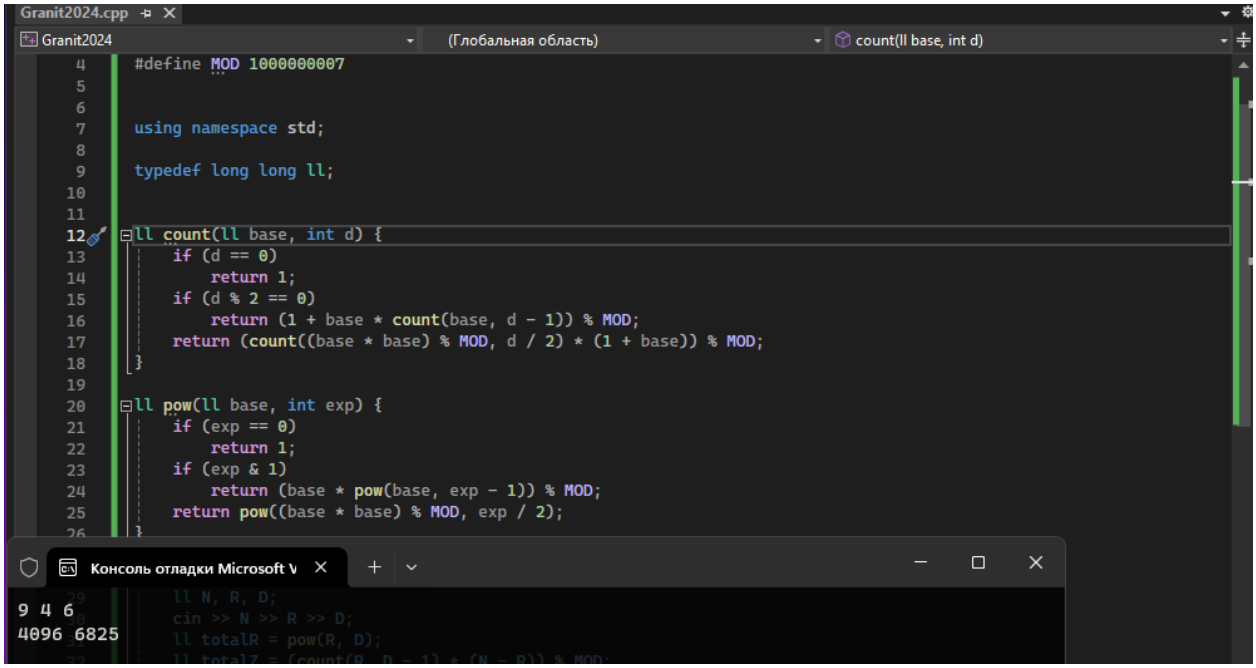
```
#define _CRT_SECURE_NO_WARNINGS
#include <cstdio>
#include <iostream>
#define MOD 1000000007


using namespace std;

typedef long long ll;


ll count(ll base, int d) {
    if (d == 0)
        return 1;
    if (d % 2 == 0)
        return (1 + base * count(base, d - 1)) % MOD;
    return (count((base * base) % MOD, d / 2) * (1 + base)) % MOD;
}

ll pow(ll base, int exp) {
    if (exp == 0)
        return 1;
    if (exp & 1)
        return (base * pow(base, exp - 1)) % MOD;
    return pow((base * base) % MOD, exp / 2);
}

int main() {
    ll N, R, D;
```

```
    cin >> N >> R >> D;
    ll totalR = pow(R, D);
    ll totalZ = (count(R, D - 1) * (N - R)) % MOD;
    printf("%lld %lld\n", totalR, totalZ);
    return 0;
}
```

Реализация на Python

```
MOD = 1000000007

def count(base, d):
    if d == 0:
        return 1
    if d % 2 == 0:
        return (1 + base * count(base, d - 1)) % MOD
    return (count((base * base) % MOD, d // 2) * (1 + base)) % MOD

def pow(base, exp):
    if exp == 0:
        return 1
    if exp & 1:
        return (base * pow(base, exp - 1)) % MOD
    return pow((base * base) % MOD, exp // 2)

N, R, D = map(int, input().split())
totalR = pow(R, D)
totalZ = (count(R, D - 1) * (N - R)) % MOD
print(totalR, totalZ)
```

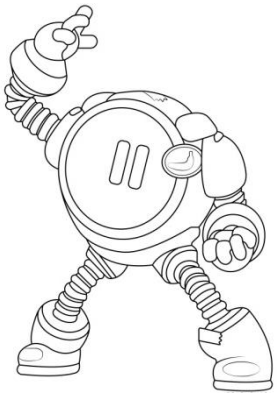
Демонстрация C++



Ответ: 4096 6825

Задача 6.

СОРЕВНОВАНИЯ ПО РОБОТОТЕХНИКЕ



В соревнование участвуют роботы квадратной формы, которые могут двигаться только в четырех направлениях параллельно сторонам трассы. На одном из этапов соревнований робот располагается на стартовой дорожке слева. Он должен заехать на трассу с левого края, пересечь лабиринт (не обязательно по кратчайшему пути) и выйти через правый край, чтобы попасть на финишную дорожку. Победителем становится участник с самым большим роботом (т.е. роботом с самой большой квадратной формой), который выполнит задание.

Организаторы хотят проверить трассу непосредственно перед соревнованием и выяснить размер роботов, которых нужно будет построить для соревнований. Напишите программу, которая, зная планировку трассы, вычисляла, какой должна быть максимальная длина стороны робота.

Входные данные: в первой строке, указаны ширина и длина трассы. Следующие строки n_i содержат m_i символов, описывающих *i-ю* дорожку:

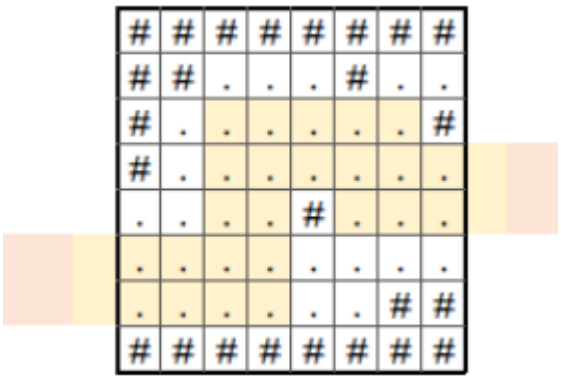
- « **.** » обозначает пустую ячейку, по которой может двигаться робот,
- « # » обозначает занятую ячейку - стену.

Верхний и нижний ряды всех дорожек состоят только из занятых ячеек.

Результат должен содержать одно целое число – максимальную длину стороны робота, такую, что робот такого размера может преодолеть трассу.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод:	Результат:	
8 8 ######## ##...#.. #......# #....... ....#... ....## ########	2	 Пример движения робота 2×2: Более крупные роботы не преодолеют этот путь.

Исходные данные:

```
13 23
#######################
..#..#.................
#.......#......#...##..
........#..#........##.
..........#............
....#.......#..#.......
....#..................
#..#..............#.#..
..#......#..#.....#....
.....#.....#.#....#....
...........#...#.......
..........#.......#....
#######################
```

Решение:

Поиск максимального размера робота для прохода по трассе использует подход основанный на бинарном поиске и поиске в глубину (DFS).

В начале программы определяются несколько глобальных переменных, таких как размеры карты **n** и **m**, размер квадрата a, массивы для отслеживания пути **track**, массив для хранения значений префиксных сумм **take_dp** и массив для отслеживания посещенных клеток **visited**.

Функция **free_square** проверяет, можно ли разместить квадрат с верхним левым углом в клетке (i, j) и размером a. Она также проверяет, что внутри этого квадрата нет препятствий.

Функция **dfs** выполняет поиск в глубину для обхода клеток карты. Она проверяет соседние клетки и вызывает себя рекурсивно, если они доступны.

Функция **binary_search** используется для выполнения бинарного поиска максимального размера квадрата. Она вызывает функцию **dfs** для обхода клеток карты и проверяет, можно ли достичь правой границы с использованием найденного размера квадрата.

В функции **main** происходит чтение входных данных, инициализация массивов **track** и таблицы префиксных сумм **take_dp**, которая показывает, сколько занятых клеток в прямоугольнике с углами (0,0) и (i,j).

После заполнения таблицы **take_dp** для каждой клетки трассы вычисляется значение, на основе которого определяется, является ли клетка доступной. Когда все доступные клетки найдены, выполняется поиск в глубину по доступным клеткам начиная с клеток всего левого столбца. Если поиск в глубину достигает клеток правого столбца, то размер стороны квадрата подходит для работа.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>

using namespace std;

int n, m, a;
char Track[502][502];
int TakeDp[502][502];
bool Visited[502][502];

// В прямоугольнике с углами (I, J) и (I+a-1, J+a-1) нет занятой ячейки '#'
bool freeSquare(int I, int J) {
  if (I + (a - 1) > n) {
    return false;
  }
  int I2 = I + a - 1;
  int J2 = min(J + a - 1, m);
  if (TakeDp[I2][J2] - TakeDp[I - 1][J2] - TakeDp[I2][J - 1] + TakeDp[I - 1][J - 1] > 0) {
    return false;
  }
```

```
    return true;
}

void dfs(int I, int J) {
    if (Visited[I][J + 1] == 0 && freeSquare(I, J + 1) == true) {
        Visited[I][J + 1] = 1;
        dfs(I, J + 1);
    }
    if (Visited[I][J - 1] == 0 && freeSquare(I, J - 1) == true) {
        Visited[I][J - 1] = 1;
        dfs(I, J - 1);
    }
    if (Visited[I - 1][J] == 0 && freeSquare(I - 1, J) == true) {
        Visited[I - 1][J] = 1;
        dfs(I - 1, J);
    }
    if (Visited[I + 1][J] == 0 && freeSquare(I + 1, J) == true) {
        Visited[I + 1][J] = 1;
        dfs(I + 1, J);
    }
}

bool binarySearch() {
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, 1) == true && Visited[i][1] == 0) {
            Visited[i][1] = 1;
            dfs(i, 1);
        }
    }
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, m) == true && Visited[i][m] == 1) {
            return true;
        }
    }
```

```
    return false;
}

int main() {
    int T;
    cin >> T;
    for (int t = 0; t < T; t++) {
        cin >> n >> m;

        for (int i = 0; i <= n + 1; i++) {
            for (int j = 0; j <= m + 1; j++) {
                Track[i][j] = '.';  // Вокруг дорожки добавляется рамка из свободных ячеек («.»)
                TakeDp[i][j] = 0;
                Visited[i][j] = 1;
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                cin >> Track[i][j];
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                TakeDp[i][j] = TakeDp[i - 1][j] + TakeDp[i][j - 1] - TakeDp[i - 1][j - 1];
                if (Track[i][j] == '#') {
                    TakeDp[i][j]++;
                }
            }
        }

        int left = 0, right = n, middle;
        while (right - left > 1) {
```

```
        middle = (left + right) / 2;
        a = middle;

        for (int i = 1; i <= n; i++) {
          for (int j = 1; j <= m; j++) {
            Visited[i][j] = 0;
          }
        }

        if (binarySearch()) {
          left = middle;
        }
        else {
          right = middle;
        }
      }

      cout << left << endl;
    }
    return 0;
}
```

Реализация на Python

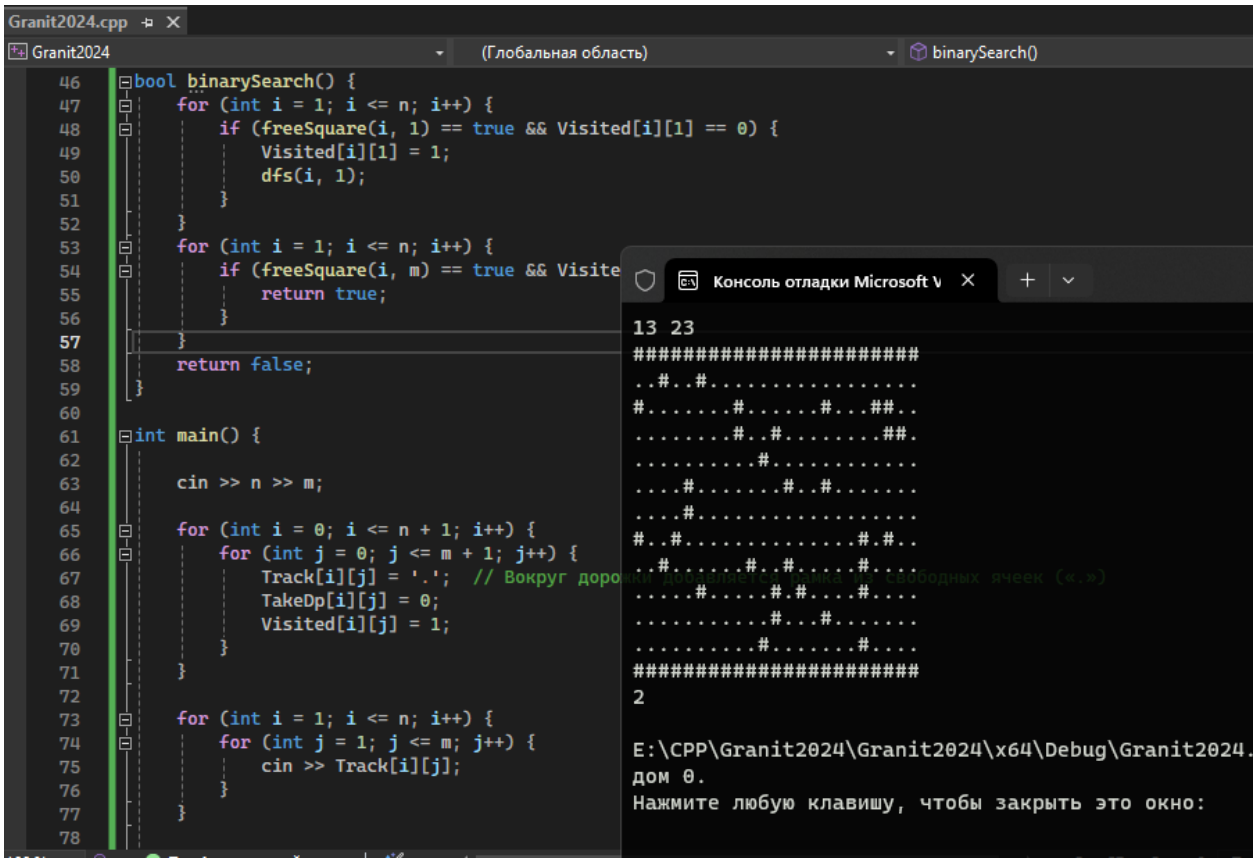
```
n, m, a = 0, 0, 0
track = [['' for _ in range(502)] for _ in range(502)]
take_dp = [[0 for _ in range(502)] for _ in range(502)]
visited = [[False for _ in range(502)] for _ in range(502)]

def free_square(i: int, j: int) -> bool:
    global n, m, a, take_dp
    if i + (a - 1) > n:
        return False
    i2 = i + a - 1
```

```
    j2 = min(j + a - 1, m)
    if take_dp[i2][j2] - take_dp[i - 1][j2] - take_dp[i2][j - 1] + take_dp[i - 1][j - 1] > 0:
        return False
    return True

def dfs(i: int, j: int) -> None:
    global visited
    if j + 1 <= m and not visited[i][j + 1] and free_square(i, j + 1):
        visited[i][j + 1] = True
        dfs(i, j + 1)
    if j - 1 >= 1 and not visited[i][j - 1] and free_square(i, j - 1):
        visited[i][j - 1] = True
        dfs(i, j - 1)
    if i - 1 >= 1 and not visited[i - 1][j] and free_square(i - 1, j):
        visited[i - 1][j] = True
        dfs(i - 1, j)
    if i + 1 <= n and not visited[i + 1][j] and free_square(i + 1, j):
        visited[i + 1][j] = True
        dfs(i + 1, j)

def binary_search() -> bool:
    global n, m, a, visited
    for i in range(1, n + 1):
        if free_square(i, 1) and not visited[i][1]:
            visited[i][1] = True
            dfs(i, 1)
    for i in range(1, n + 1):
        if free_square(i, m) and visited[i][m]:
            return True
    return False

def main():
    global n, m, a, track, take_dp, visited
    n, m = map(int, input().split())
```

```
for i in range(n + 2):
    for j in range(m + 2):
        track[i][j] = '.'
        take_dp[i][j] = 0
        visited[i][j] = False

for i in range(1, n + 1):
    row = input()
    for j in range(1, m + 1):
        track[i][j] = row[j - 1]

for i in range(1, n + 1):
    for j in range(1, m + 1):
        take_dp[i][j] = take_dp[i - 1][j] + take_dp[i][j - 1] - take_dp[i - 1][j - 1]
        if track[i][j] == '#':
            take_dp[i][j] += 1

left, right = 0, n
while right - left > 1:
    middle = (left + right) // 2
    a = middle

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            visited[i][j] = False

    if binary_search():
        left = middle
    else:
        right = middle

print(left)
```

```
if __name__ == "__main__":
    main()
```

Демонстрация на C++



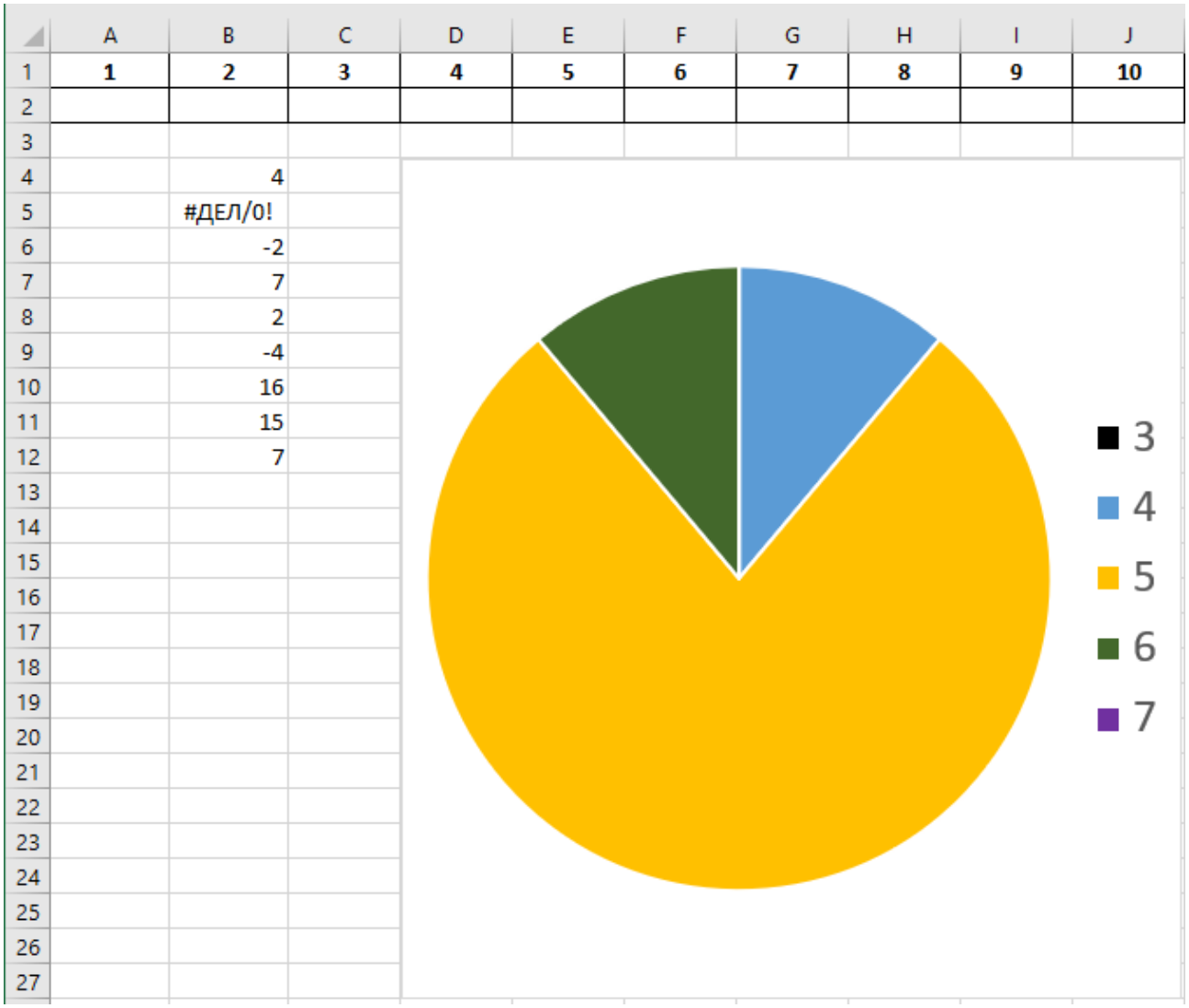
Ответ: 2

ИНФОРМАТИКА.

ВАРИАНТ 8.

Задача 1 (10 баллов)

В каждой из ячеек A2:J2 находится целое число. Определите их значения, используя круговую диаграмму и значения формул в блоке B4:B12. В формулах используются только ссылки на ячейки из диапазона A2:J2.



Задача усложняется тем, что отображение самих формул изменилось после копирования их в блок A5:A13.

	A
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")
6	=A3*D$2/($F3-G3)
7	=$F$2+F$2+$H3
8	=#ССЫЛКА!-E$2+#ССЫЛКА!
9	=ЕСЛИ(A3+D3;1;2)
10	=$A3+A3
11	=D3+C3+H3
12	=D3+C3*H3
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3

Решение:

1. Восстановим формулы в блоке B4:B12 исходя из формул блока A5:A13. Для получения формулы из ячейки A8 «=#ССЫЛКА!-E$2+#ССЫЛКА!» вспомним, что #ССЫЛКА! – это ссылка на ячейку, вышедшую за диапазон листа. Так как при копировании произошло смещение ссылок на 1 позицию вниз и 1 позицию влево, а также учитывая, что по условию задачи, в формулах используются только ссылки на ячейки из диапазона A2:J2, единственной ссылкой на ячейку способной выйти за диапазон листа является A2.

	A	B
4		=СЧЁТЕСЛИ($A$2:$J2;"<0")
5	=СЧЁТЕСЛИ($A$2:$J3;"<0")	=B2*E$2/($F2-H2)
6	=A3*D$2/($F3-G3)	=$F$2+G$2+$H2
7	=$F$2+F$2+$H3	=A2-F$2+A2
8	=#ССЫЛКА!-E$2+#ССЫЛКА!	=ЕСЛИ(B2+E2;1;2)
9	=ЕСЛИ(A3+D3;1;2)	=$A2+B2
10	=$A3+A3	=E2+D2+I2
11	=D3+C3+H3	=E2+D2*I2
12	=D3+C3*H3	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2
13	=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:H3);0)-I3	
14		

2. По круговой диаграмме, которая, в соответствии с легендой, отображает значения категорий 3,4,5,6 и 7, видно, что значения 3 и 7 равны 0, т.е. C2=0и G2=0. Замечание: на круговой диаграмме отрицательные значения отображаются как положительные.

3. Значение формулы «=СЧЁТЕСЛИ($A$2:$J2;"<0")» в ячейке B4 равно 4, следовательно в искомом диапазоне имеется 4 отрицательных числа.

4. Значение формулы «=B2*E$2/($F2-H2)» в ячейке B5 равно значению «#ДЕЛ/0!», это означает, что ячейки F2 и H2 равны.

5. Значение формулы «=$F$2+G$2+$H2» в ячейке B6 равно -2, следовательно F2+H2 = -2-G2 = -2–0 = -2, а так как они равны, то F2 = -1, H2 = -1.

6. Значение формулы «=A2-F$2+A$2» в ячейке B7 равно 7, следовательно 7 = A2+1+A2. Откуда A2 = 6/2 = 3.

7. Значение формулы «=ЕСЛИ(B2+E2;1;2» в ячейке B8 равно 2, следовательно условие в виде суммы «B2+E2» не выполнено, т.е. B2+E2 = 0. Делаем вывод, что значения в ячейках B2 и E2 одинаковые по модулю и различные по знаку.

8. Значение формулы «=$A2+B2» в ячейке B9 равно -4, следовательно B2 = -4-A2 = -4-3 = -7, а в соответствии с 7 пунктом E2 = 7.

9. Значение формулы в B10 «=E2+D2+I2» равно 16, а значение формулы в B11 «=E2+D2*I2» равно 15. Получаем систему уравнений $\begin{cases} E2 + D2 + I2 = 16 \\ E2 + D2 * I2 = 15 \end{cases} => \begin{cases} 7 + D2 + I2 = 16 \\ 7 + D2 * I2 = 15 \end{cases} => \begin{cases} D2 + I2 = 9 \\ D2 * I2 = 8 \end{cases}$. Решая систему уравнений, находим D2 = 1 и I2 = 8.

10. Значение формулы «=ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0)-J2» в ячейке B12 равно 7, следовательно J2 = ОКРУГЛВВЕРХ(СРЗНАЧ($A$2:I2);0) -7 = ОКРУГЛВВЕРХ (1,111111;0) -7 = 2-7 = -5.

Ответ:

	A	B	C	D	E	F	G	H	I	J
1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	3	-7	0	1	7	-1	0	-1	8	-5

Задача 2 (10 баллов)

На столе лежит две кучки драгоценных камней: в одной 12, в другой 8. Гномы ходят по очереди. За один ход можно взять один камень из одной из кучек (по выбору гнома) или взять по одному камню из двух сразу. Кто не может сделать ход (камней не осталось), проигрывает. Кто выигрывает при безошибочной игре – гном, делающий первый ход, или гном, делающий второй ход? Поясните алгоритм графически, используя ориентированные графы и обозначая проигрышные и выигрышные позиции. Напишите на алгоритмическом языке обобщенный алгоритм для любого количества камней.

Решение:

Необходимо использовать стратегию, беря такое количество камней, чтобы в каждой кучке оставалось чётное количество камней. Позиции, где число камней в каждой кучке чётное, будут проигрышными (для того, кто в них оказался), а где хотя бы одна

кучка с нечётным количеством камней — выигрышными. В нашем случае, когда в игре в одной 12, в другой 8 камней, выигрывает второй игрок. Для безошибочной игры, он должен ставить противника в проигрышную позицию, то есть брать столько камней, чтобы осталось чётное количество в каждой кучке.

Приведём стратегию выигрывающего гнома:

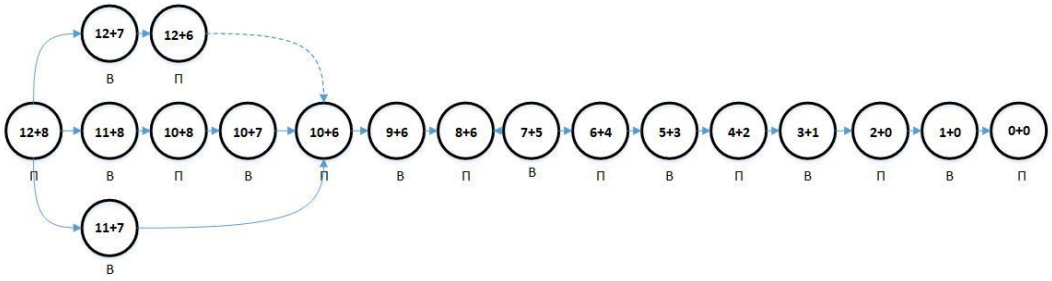
Ход_1: Игрок_1 (игрок делающий ход первым) берет 1 камень из любой кучки или по 1 камню из каждой кучки (в данном случае камней может остаться 11 и 7, 11 и 8, 12 и 7)

Ход_2: Игрок_2 (игрок, делающий ход вторым) берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней (в данном случае игрок 2 может свезти все к позиции 10 и 6, 10 и 8, 12 и 6).

Ход_3: Игрок_1 берет 1 камень из любой кучки или по 1 камню из каждой кучки.

Ход_4: Игрок_2 берет 1 камень из любой кучки или по 1 камню из каждой кучки, так чтобы на ход противника в каждой кучке оставалось чётное количество камней. И так далее до Победа Игрок_2. Проиграть невозможно.

Чтобы проанализировать игру, изобразим возможные варианты ходов гномов в виде ориентированного графа. Отметим выигрышные (В) и проигрышные (П) ходы:



Алгоритм для Игрок_2 (обобщенный):

Шаг 1 Игрок _1 взял m=1 или n=1 или (m=1 и n=1) камней

Если (M-m) нечётное (N-n) чётное

Шаг 2 Игрок _2 взял m=1 камень, так чтобы (M-m) кратно 2

Если Остаток = 0 – значит СТОП_Выиграл.

Если (N-n) нечётное (M-m) чётное

Шаг 2 Игрок _2 взял n=1 камень, так чтобы (N-n) кратно 2

Если (N-n) нечётное (M-m) нечётное

Шаг 2 Игрок_2 взял m=1 и n=1 камни, так чтобы (M-m) и (N-n) кратно 2

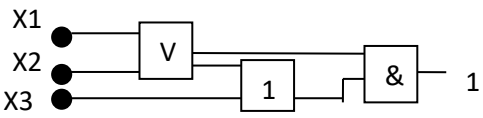
Если Остаток = 0 – значит СТОП_Выиграл.

Алгоритм подходит в общем случае, когда и M и N четны.

Задача 3 (10 баллов)

Дана логическая схема. Найти все наборы, при которых выражение, записанное в схеме, принимает значение истина. Подставить i-2 наборы значений из таблицы истинности в логическое выражение для определения, чему равно результирующее выражение. В ответе указать, чему будет равен результирующий столбец.

Логическая схема:



Выражение:

$x1 \leftrightarrow x2 \,\&\, \bar{x}3 \oplus x1$

Где & - это конъюнкция

1 - это исключающее или

V - это дизъюнкция

Решение:

Каждый i-ый набор -это тот, при котором выражение, записанное с помощью логической схемы дает значение, указанное по условию задачи (истина). В условии сказано подставить i-2 наборы, это значит, что необходимо подставить 1, 3 и 5 наборы, т.к. 3, 5 и 7 дали в результирующем столбце «истина».

X1	x2	x3	x1Vx2	x1Vx2 ⊕ x3	x1Vx2 ⊕ x3& x1Vx2
0	0	0	0	0	0

0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0

$x1 \leftrightarrow x2 \,\&\, \bar{x}3 \oplus x1$

x1	x2	x3	$\bar{x}\,3$	$x2 \wedge \bar{x}3$	$x1 \leftrightarrow x2 \,\&\, \bar{x}3$	$x1 \leftrightarrow x2 \,\&\, \bar{x}3 \oplus x1$
0	0	0	1	0	1	1
0	1	0	1	1	0	0
1	0	0	1	0	0	1

Ответ. 101

Задача 4.

Ваня решает задачу по обработке изображений. На первом рисунке представлено цветное изображение, пиксели которого закодированы с помощью цветовой модели RGB. Известно, что в исходном изображение в каждом канале или максимальная, или минимальная яркости. На втором рисунке представлено обработанное изображение. Ване необходимо понять, как именно было обработано изображение на первом рисунке, чтобы получилось изображение на втором рисунке. Запишите, в чем именно заключался алгоритм обработки. Схематически укажите на первом и втором рисунках значения яркости в RGB и HEX форматах.

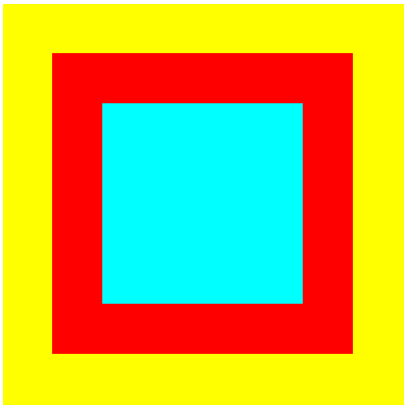


Рисунок 1

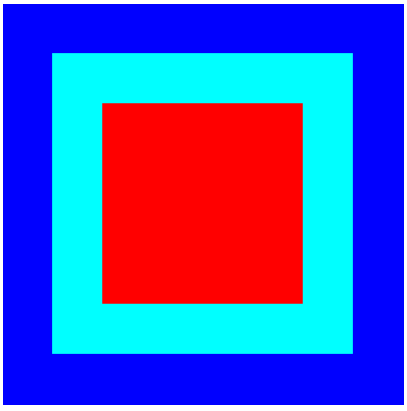


Рисунок 2

Решение:

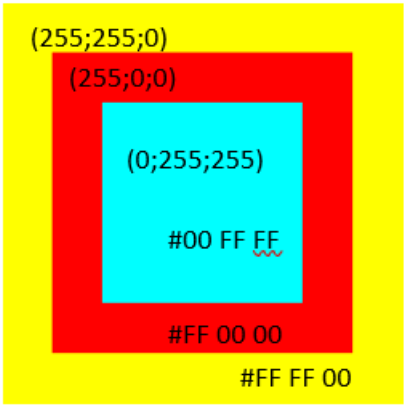
Как мы видим, в обработанном изображении красный цвет стал голубым. Получается, что после обработки, яркости красной составляющей приняли нулевые значения, а яркости зелёной и синей приняли значение 255. Делаем предположение, что была произведена инверсия изображения по трем каналам RGB, то есть алгоритм обработки выглядит следующим образом:

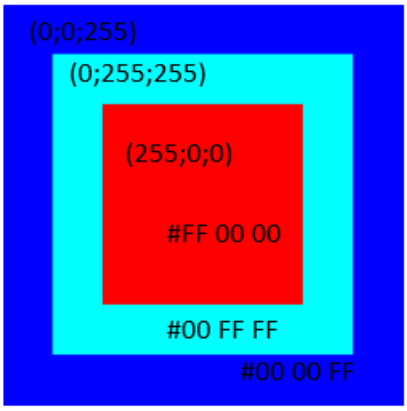
$$Y_R = 255 - Y_R$$
$$Y_G = 255 - Y_G$$
$$Y_B = 255 - Y_B$$

Жёлтый цвет стал синим, то есть яркости красной и зелёной составляющих приняли нулевые значения, а яркости синей составляющей приняли значение 255.

Голубой цвет стал красным, то есть яркости зелёной и синей составляющих приняли нулевые значения, а яркости красной составляющей приняли значение 255.

Далее укажем на рисунках значения яркости в RGB и HEX формате

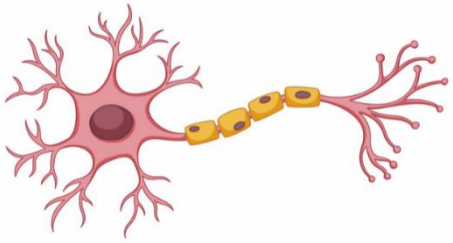




Задача 5.

ОТКРЫТИЯ АРКТИКИ

Однажды после очередной экспедиции в Арктику ученые университета привезли образец льдины, добытый из глубин арктических пластов. При исследовании его в лаборатории химии они обнаружили новые клетки красного цвета.



Клетка сама по себе ничего не делала, и ученые уже расстроились, но однажды очень жарким летом, сломались кондиционеры в лаборатории, и клетка проснулась. За ночь клетка размножилась и стала похожа на дерево, состоящее из соединительных каналов и N прикрепленных к ним таких же клеток, из которых R красные, а остальные зеленые. Позже, выяснилось, что новые клетки тоже размножаются, и каждую

последующую ночь все красные клетки вырастают в другое дерево, подобное тому, которое выросло из первой клетки в первую ночь.

Помогите, ученым написать программу, определяющую, сколько красных и сколько зеленых клеток будет в получившемся дереве через D дней.

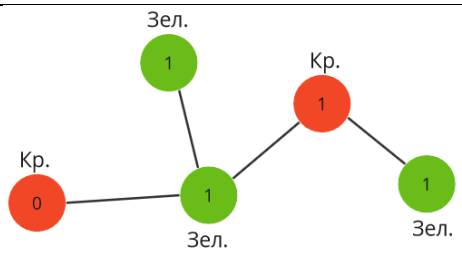
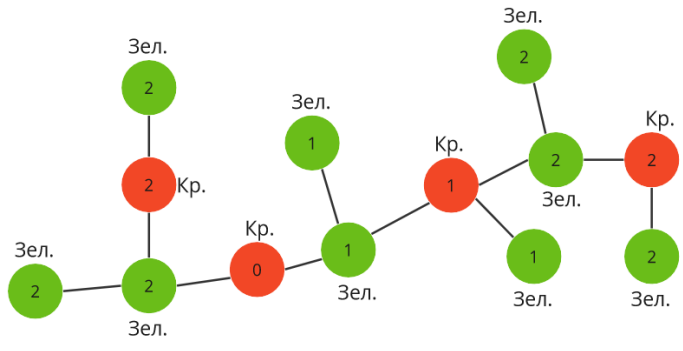
Получите ответ по модулю 1 000 000 007.

Входные данные. Три целых числа разделенных пробелом N, R и D, где N – количество клеток, которые появляются после первого дня. R – Количество красных клеток и D – количество дней.

Результат должен содержать два числа через пробел — количество красных и зеленых яиц за D дней, по модулю 1 000 000 007.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

<table>
<tr><td>Ввод:
5 2 2</td><td>Результат:
4 9</td></tr>
<tr><td colspan="2">Пояснение</td></tr>
<tr><td>После первого дня будет 2 красные клетки и 3 зеленых. Дерево может выглядеть так.</td><td> 
</td></tr>
<tr><td>Через два дня дерево клеток будет выглядеть уже так. Числа на клетке обозначают: через сколько дней она появилась.</td><td> 
</td></tr>
</table>

Исходные данные:

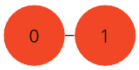
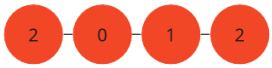
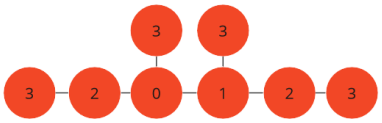
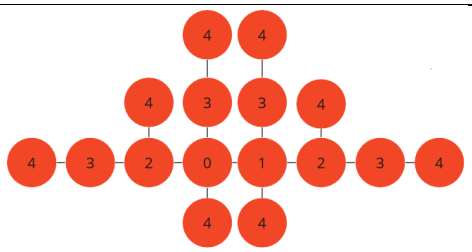
8 5 5

Решение:

Итак, у нас есть дерево с клетками, из которых R красных, а остальные (N - R) зеленые. Каждую ночь все красные клетки размножаются в новое дерево такого же размера, как и первое, а затем все красные клетки в этом новом дереве также размножаются в новое дерево и так далее.

Мы хотим найти количество красных и зеленых клеток через D дней.

Рассмотрим процесс размножения для красных клеток, если у нас каждую ночь рождается новая красная клетка.

После 1 дня		$2 = 2^1$
После 2 дня		$4 = 2^2$
После 3 дня		$8 = 2^3$
После 4 дня		$16 = 2^4$

Как можно заметит, что количество красных клеток будет равно их заданному количеству в степени равной заданному дню. Для зеленых клеток ситуация подобная. Проблема заключается в возрастании чисел и вычислении степени.

Для нахождения красных клеток через D дней, определим функцию pow(base, exp) которая используется для быстрого возведения числа base в степень exp по модулю MOD. Она также использует рекурсию, чтобы разбить вычисление степени на более мелкие шаги. В алгоритме используется взятие остатка от деления (mod), чтобы эффективно работать с большими числами и избегать переполнения.

Для нахождения зеленых клеток используется подобный рекурсивный алгоритм.

От участников требуется показать навыки реализации алгоритма быстрого возведения в степень.

Реализация на C++

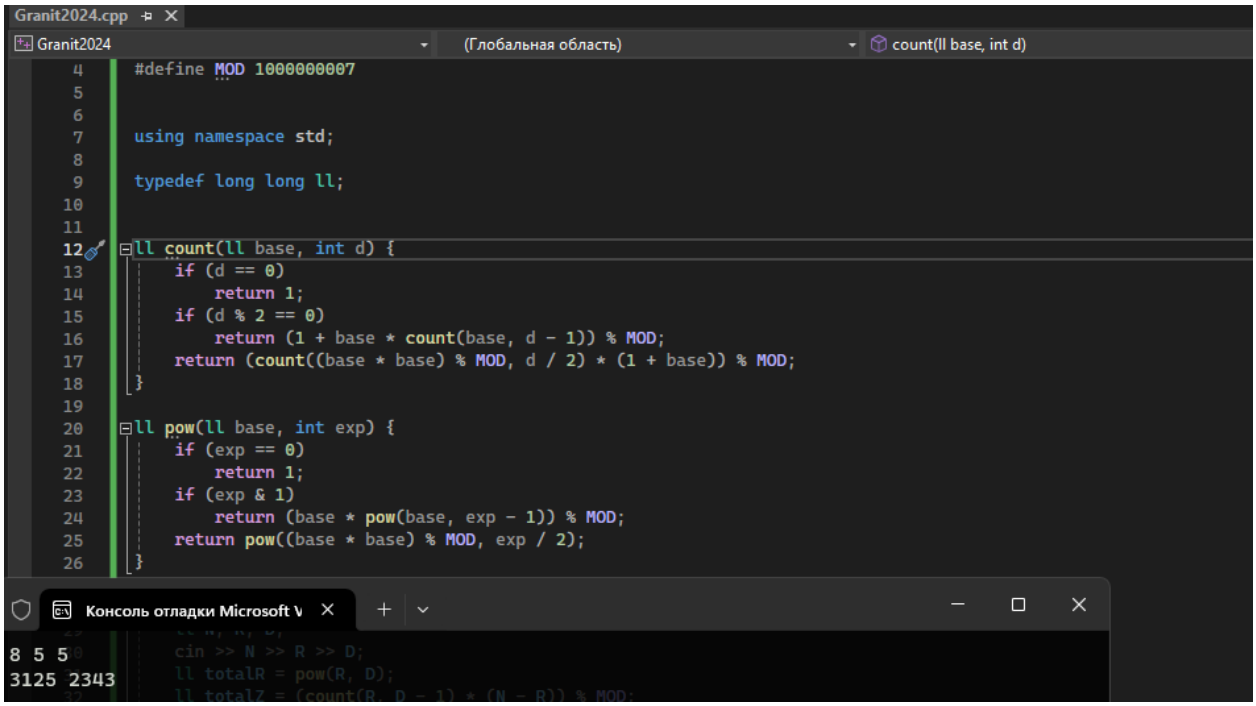
```
#define _CRT_SECURE_NO_WARNINGS
#include <cstdio>
#include <iostream>
#define MOD 1000000007


using namespace std;

typedef long long ll;


ll count(ll base, int d) {
    if (d == 0)
        return 1;
    if (d % 2 == 0)
        return (1 + base * count(base, d - 1)) % MOD;
    return (count((base * base) % MOD, d / 2) * (1 + base)) % MOD;
}

ll pow(ll base, int exp) {
    if (exp == 0)
        return 1;
    if (exp & 1)
        return (base * pow(base, exp - 1)) % MOD;
    return pow((base * base) % MOD, exp / 2);
}

int main() {
    ll N, R, D;
```

```
    cin >> N >> R >> D;
    ll totalR = pow(R, D);
    ll totalZ = (count(R, D - 1) * (N - R)) % MOD;
    printf("%lld %lld\n", totalR, totalZ);
    return 0;
}
```

Реализация на Python

```
MOD = 1000000007

def count(base, d):
    if d == 0:
        return 1
    if d % 2 == 0:
        return (1 + base * count(base, d - 1)) % MOD
    return (count((base * base) % MOD, d // 2) * (1 + base)) % MOD

def pow(base, exp):
    if exp == 0:
        return 1
    if exp & 1:
        return (base * pow(base, exp - 1)) % MOD
    return pow((base * base) % MOD, exp // 2)

N, R, D = map(int, input().split())
totalR = pow(R, D)
totalZ = (count(R, D - 1) * (N - R)) % MOD
print(totalR, totalZ)
```

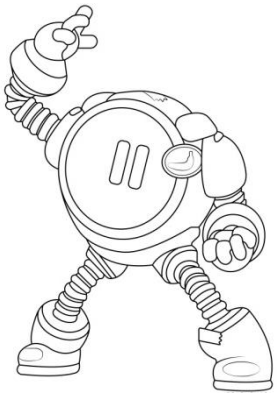
Демонстрация C++



Ответ: 3125 2343

Задача 6.

СОРЕВНОВАНИЯ ПО РОБОТОТЕХНИКЕ



В соревнование участвуют роботы квадратной формы, которые могут двигаться только в четырех направлениях параллельно сторонам трассы. На одном из этапов соревнований робот располагается на стартовой дорожке слева. Он должен заехать на трассу с левого края, пересечь лабиринт (не обязательно по кратчайшему пути) и выйти через правый край, чтобы попасть на финишную дорожку. Победителем становится участник с самым большим роботом (т.е. роботом с самой большой квадратной формой), который выполнит задание.

Организаторы хотят проверить трассу непосредственно перед соревнованием и выяснить размер роботов, которых нужно будет построить для соревнований. Напишите программу, которая, зная планировку трассы, вычисляла, какой должна быть максимальная длина стороны робота.

Входные данные: в первой строке, указаны ширина и длина трассы. Следующие строки n_i содержат m_i символов, описывающих *i-ю* дорожку:

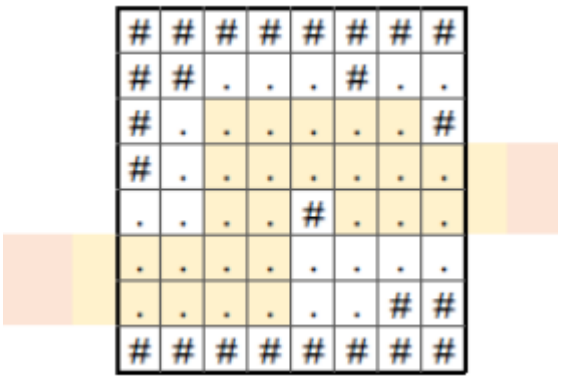
- « **.** » обозначает пустую ячейку, по которой может двигаться робот,
- « # » обозначает занятую ячейку - стену.

Верхний и нижний ряды всех дорожек состоят только из занятых ячеек.

Результат должен содержать одно целое число – максимальную длину стороны робота, такую, что робот такого размера может преодолеть трассу.

Несмотря на то, что в вашем задании указаны достаточно малые значения, ваша программа должна выполняться и для больших входных значений.

Пример:

Ввод:	Результат:	
8 8 ######## ##...#.. #......# #....... ....#... ....## ########	2	 Пример движения робота 2×2: Более крупные роботы не преодолеют этот путь.

Исходные данные:

```
14 23
#######################
#.#.........#....#.....
.#....#....#...........
##...........#.........
..............#........
.........#...#.........
...#........##.....#...
.#.##..................
........#...........#..
.......................
.....#.#.#....#....##..
#.#..............#....
...#.##......##........
#######################
```

Решение:

Поиск максимального размера робота для прохода по трассе использует подход основанный на бинарном поиске и поиске в глубину (DFS).

В начале программы определяются несколько глобальных переменных, таких как размеры карты **n** и **m**, размер квадрата a, массивы для отслеживания пути **track**, массив для хранения значений префиксных сумм **take_dp** и массив для отслеживания посещенных клеток **visited**.

Функция **free_square** проверяет, можно ли разместить квадрат с верхним левым углом в клетке (i, j) и размером a. Она также проверяет, что внутри этого квадрата нет препятствий.

Функция **dfs** выполняет поиск в глубину для обхода клеток карты. Она проверяет соседние клетки и вызывает себя рекурсивно, если они доступны.

Функция **binary_search** используется для выполнения бинарного поиска максимального размера квадрата. Она вызывает функцию **dfs** для обхода клеток карты и проверяет, можно ли достичь правой границы с использованием найденного размера квадрата.

В функции **main** происходит чтение входных данных, инициализация массивов **track** и таблицы префиксных сумм **take_dp**, которая показывает, сколько занятых клеток в прямоугольнике с углами (0,0) и (i,j).

После заполнения таблицы **take_dp** для каждой клетки трассы вычисляется значение, на основе которого определяется, является ли клетка доступной. Когда все доступные клетки найдены, выполняется поиск в глубину по доступным клеткам начиная с клеток всего левого столбца. Если поиск в глубину достигает клеток правого столбца, то размер стороны квадрата подходит для работа.

Реализация на C++

```
#include <iostream>
#include <vector>
#include <set>
#include <tuple>

using namespace std;

int n, m, a;
char Track[502][502];
int TakeDp[502][502];
bool Visited[502][502];

// В прямоугольнике с углами (I, J) и (I+a-1, J+a-1) нет занятой ячейки '#'
bool freeSquare(int I, int J) {
  if (I + (a - 1) > n) {
    return false;
  }
  int I2 = I + a - 1;
  int J2 = min(J + a - 1, m);
  if (TakeDp[I2][J2] - TakeDp[I - 1][J2] - TakeDp[I2][J - 1] + TakeDp[I - 1][J - 1] > 0) {
    return false;
  }
```

```
    return true;
}

void dfs(int I, int J) {
    if (Visited[I][J + 1] == 0 && freeSquare(I, J + 1) == true) {
        Visited[I][J + 1] = 1;
        dfs(I, J + 1);
    }
    if (Visited[I][J - 1] == 0 && freeSquare(I, J - 1) == true) {
        Visited[I][J - 1] = 1;
        dfs(I, J - 1);
    }
    if (Visited[I - 1][J] == 0 && freeSquare(I - 1, J) == true) {
        Visited[I - 1][J] = 1;
        dfs(I - 1, J);
    }
    if (Visited[I + 1][J] == 0 && freeSquare(I + 1, J) == true) {
        Visited[I + 1][J] = 1;
        dfs(I + 1, J);
    }
}

bool binarySearch() {
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, 1) == true && Visited[i][1] == 0) {
            Visited[i][1] = 1;
            dfs(i, 1);
        }
    }
    for (int i = 1; i <= n; i++) {
        if (freeSquare(i, m) == true && Visited[i][m] == 1) {
            return true;
        }
    }
```

```
    return false;
}

int main() {
    int T;
    cin >> T;
    for (int t = 0; t < T; t++) {
        cin >> n >> m;

        for (int i = 0; i <= n + 1; i++) {
            for (int j = 0; j <= m + 1; j++) {
                Track[i][j] = '.';  // Вокруг дорожки добавляется рамка из свободных ячеек («.»)
                TakeDp[i][j] = 0;
                Visited[i][j] = 1;
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                cin >> Track[i][j];
            }
        }

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                TakeDp[i][j] = TakeDp[i - 1][j] + TakeDp[i][j - 1] - TakeDp[i - 1][j - 1];
                if (Track[i][j] == '#') {
                    TakeDp[i][j]++;
                }
            }
        }

        int left = 0, right = n, middle;
        while (right - left > 1) {
```

```
        middle = (left + right) / 2;
        a = middle;

        for (int i = 1; i <= n; i++) {
          for (int j = 1; j <= m; j++) {
            Visited[i][j] = 0;
          }
        }

        if (binarySearch()) {
          left = middle;
        }
        else {
          right = middle;
        }
      }

      cout << left << endl;
    }
    return 0;
}
```

Реализация на Python

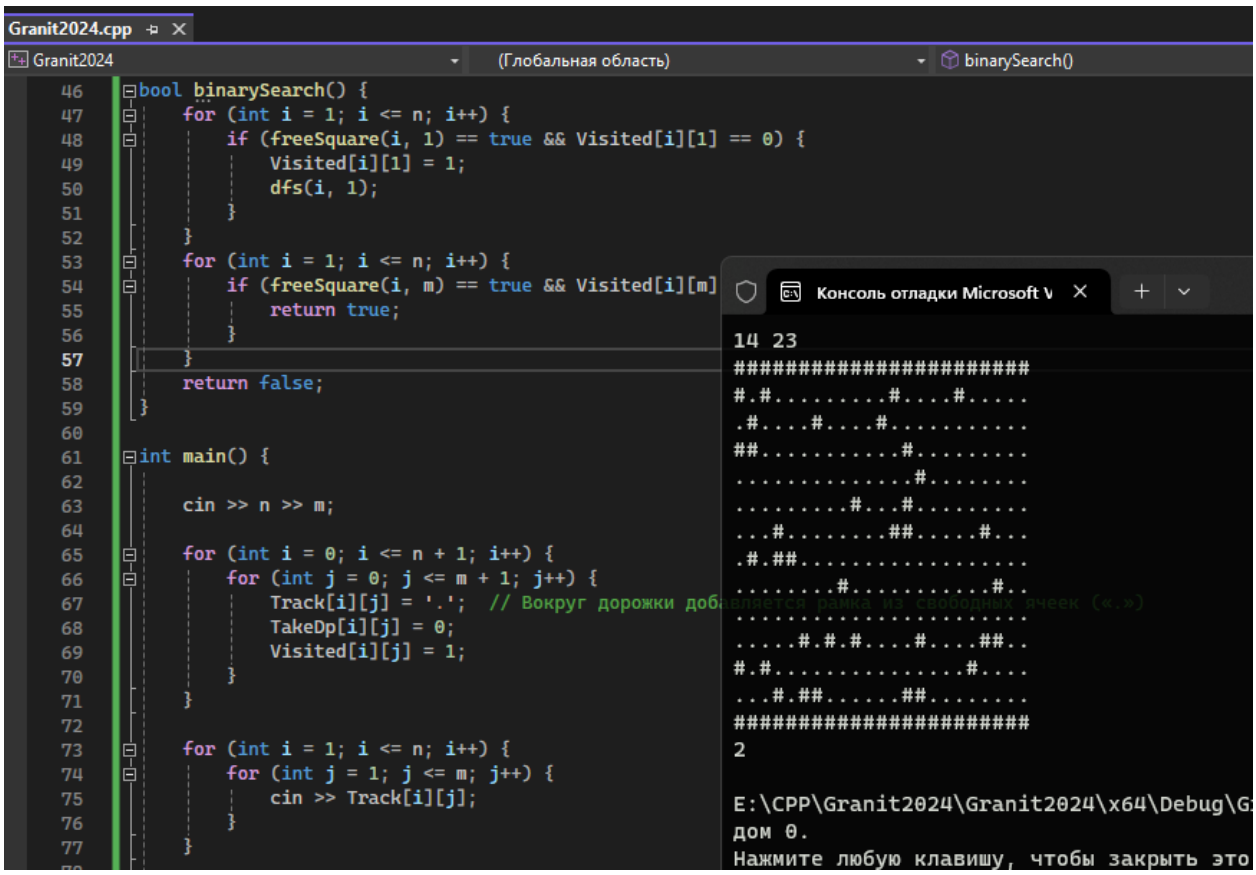
```
n, m, a = 0, 0, 0
track = [['' for _ in range(502)] for _ in range(502)]
take_dp = [[0 for _ in range(502)] for _ in range(502)]
visited = [[False for _ in range(502)] for _ in range(502)]

def free_square(i: int, j: int) -> bool:
    global n, m, a, take_dp
    if i + (a - 1) > n:
        return False
    i2 = i + a - 1
```

```
    j2 = min(j + a - 1, m)
    if take_dp[i2][j2] - take_dp[i - 1][j2] - take_dp[i2][j - 1] + take_dp[i - 1][j - 1] > 0:
        return False
    return True

def dfs(i: int, j: int) -> None:
    global visited
    if j + 1 <= m and not visited[i][j + 1] and free_square(i, j + 1):
        visited[i][j + 1] = True
        dfs(i, j + 1)
    if j - 1 >= 1 and not visited[i][j - 1] and free_square(i, j - 1):
        visited[i][j - 1] = True
        dfs(i, j - 1)
    if i - 1 >= 1 and not visited[i - 1][j] and free_square(i - 1, j):
        visited[i - 1][j] = True
        dfs(i - 1, j)
    if i + 1 <= n and not visited[i + 1][j] and free_square(i + 1, j):
        visited[i + 1][j] = True
        dfs(i + 1, j)

def binary_search() -> bool:
    global n, m, a, visited
    for i in range(1, n + 1):
        if free_square(i, 1) and not visited[i][1]:
            visited[i][1] = True
            dfs(i, 1)
    for i in range(1, n + 1):
        if free_square(i, m) and visited[i][m]:
            return True
    return False

def main():
    global n, m, a, track, take_dp, visited
    n, m = map(int, input().split())
```

```
for i in range(n + 2):
    for j in range(m + 2):
        track[i][j] = '.'
        take_dp[i][j] = 0
        visited[i][j] = False

for i in range(1, n + 1):
    row = input()
    for j in range(1, m + 1):
        track[i][j] = row[j - 1]

for i in range(1, n + 1):
    for j in range(1, m + 1):
        take_dp[i][j] = take_dp[i - 1][j] + take_dp[i][j - 1] - take_dp[i - 1][j - 1]
        if track[i][j] == '#':
            take_dp[i][j] += 1

left, right = 0, n
while right - left > 1:
    middle = (left + right) // 2
    a = middle

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            visited[i][j] = False

    if binary_search():
        left = middle
    else:
        right = middle

print(left)
```

```
if __name__ == "__main__":
    main()
```

Демонстрация на C++



Ответ: 2