Программирование, отборочный этап, 11 класс

Вариант 1

Задача 1 (8 баллов)

Условие

Написать программу, которая определит, насколько изменится число, если минимальную по значению цифру, входящую в него, увеличить на 2.
Входные данные: натуральное число в десятичной системе счисления, не превышающее 10^9.
Выходные данные: разность между числом, полученным в соответствии с заданием, и исходным числом.

Пример

Входные данные	Выходные данные
1234	2000
4573206	20

Примечание: минимальная цифра в числе не повторяется.

Проверочные тесты

Входные данные	Ожидаемый результат
1234	2000
4573206	20
13	20
31	2
324567899	20000000
987654320	2

Пример решения

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
	string n;
	cin >> n;
	int minK = 1;
```

```
      int minNum = int(n[n.length() - 1]-48);
      int k = 10;
      for (int i = n.length() - 2; i >= 0; i--) {
      if (n[i]-48 < minNum) {
            minNum = n[i]-48;
            minK = k;
      }
      k *= 10;
      }
      cout << 2 * minK << endl;
      return 0;
}
```

Задача 2 (12 баллов)

Условие

Для некоторого десятичного числа определить, сколько цифр 3 содержит его запись в четверичной системе счисления.
Входные данные: натуральное число в десятичной с/с, не превышающее 10^9.
Выходные данные: количество цифр 3 в 4-й с/с.

Пример

Входные данные	Выходные данные
390	0
219	2

Проверочные тесты

Входные данные	Ожидаемый результат
390	0
219	2
108	1
3	1
1638	0
59	2

Пример решения

```
#include <iostream>
#include <vector>


using namespace std;
using ll = long long;

int main() {
      int ans = 0;
      ll num;
      cin >> num;
      int ext = 1;
      int c = 1;
      while (ext <= num) {
      ext *= 4;
      ++c;
      }
      ext /= 4;
      --c;
      for (int i = 0; i < c; ++i) {
      if (num / ext == 3) {
            ++ans;
      }
      num %= ext;
      ext /= 4;
      }
      cout << ans;
      return 0;
}
```

Задача 3 (16 баллов)

Условие

Петя изучает алгоритмы сортировки. Чтобы лучше разобраться в работе сортировки пузырьком, он решил упорядочить массив букв не по алфавиту, а в некоторой произвольной последовательности (например, в порядке расположения букв на клавиатуре, или любой другой). Также Петя хочет узнать, сколько перестановок символов для выполнения сортировки требуется произвести.

Входные данные: в первой строке записано натуральное N, не превышающее 26 - количество букв латинского алфавита, которые могут встречаться в сортируемом массиве. Далее в N строках через пробел записаны заглавная латинская буква и число, соответствующее порядку этой буквы с точки зрения Пети. Далее - строка из заглавных латинских букв, по длине не превышающая 100, для которой требуется посчитать число перестановок при сортировке пузырьком по возрастанию

Выходные данные: число перестановок элементов.

Пример

Входные данные	Выходные данные	Перестановки
3 A 3 B 2 Z 1 ABZ	3	BZA - 2 ZBA - 1
5 A 10 C 5 D 7 X 4 Y 1 ACDAX	6	CDAXA - 3 CDXAA - 1 CXDAA - 1 XCDAA - 1

Проверочные тесты

Входные данные	Ожидаемый результат
3 A 3 B 2 Z 1 ABZ	3
5 A 10 C 5 D 7 X 4 Y 1 ACDAX	6
1 A 1 A	0

2 A 1 B 2 AA	0
4 A 1 B 2 C 4 D 3 ABABA	3
5 A 5 B 4 C 3 D 2 E 1 ABCDE	10

Пример решения

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
using namespace std;

struct Prefs
{
    char symb;
    int idx;
};


int dict_find(char x, vector<Prefs> dict)
{
    for (auto i : dict)
    {
      if (i.symb == x)
            return i.idx;
    }
    return -1;
}


int main()
{
    int n;
    cin >> n;
    vector<Prefs> dict(n);
    for (int i = 0; i < n; i++)
      cin >> dict[i].symb >> dict[i].idx;
    string str;
```

```
    cin >> str;
    vector<Prefs> nstr(str.size());
    for (int i = 0; i < str.size(); i++)
    {
      nstr[i].symb = str[i];
      nstr[i].idx = dict_find(str[i], dict);
    }
    int ans = 0;
    for (int i = 0; i < nstr.size() - 1; i++)
    {
      for (int j = 0; j < nstr.size() - i - 1; j++)
      {
            if (nstr[j].idx > nstr[j + 1].idx)
            {
                  swap(nstr[j], nstr[j + 1]);
                  ans++;
            }
      }
    }
    cout << ans;

    return 0;
}
```

Задача 4 (20 баллов)

Условие

Дана квадратная целочисленная матрица. Требуется переписать все её элементы в одномерный массив по следующему правилу: выбор элементов начинается из правого верхнего и левого нижнего угла и продолжается "змейкой" в сторону главной диагонали. На первой итерации после угловых элементов выбор производится в сторону левого верхнего угла. В конце записываются элементы главной диагонали от левого верхнего угла до правого нижнего.

Входные данные: в первой строке число N - количество строк и столбцов в квадратной матрице, N не превышает 20. Далее в N строках по N чисел через пробел, каждое не превышает 100 по модулю.

Выходные данные: получившийся массив, записанный в одну строку через пробел.

Пример

Входные данные	Выходные данные
3 7 3 1 8 5 4 6 9 2	1 2 3 4 5 6 7 8 9

4 13 11 3 1 12 14 9 5 4 10 15 7 2 6 8 16	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Проверочные тесты

Входные данные	Ожидаемый результат
3 7 3 1 4 8 5 2 6 9	1 2 3 4 5 6 7 8 9
4 13 11 3 1 12 14 9 5 4 10 15 7 2 6 8 16	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 5	5
2 1 2 3 4	2 3 1 4
5 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 -1 -2 -3 -4 -5	5 -1 4 6 0 -2 5 -3 9 7 3 1 2 6 8 2 4 8 0 -4 1 7 3 9 -5

Пример решения

```
n = int(input())
matrix = []
for i in range(n):
      matrix.append(input().split())

for i in range(n - 1):
      s = 0
      if i % 2 == 0:
      while i - s >= 0:
            print(matrix[i - s][n - 1 -s], end=' ')
```

```
            print(matrix[n - 1 - s][i - s], end=' ')
            s += 1
      else:
      while s <= i:
            print(matrix[s][n - 1 - i + s], end=' ')
            print(matrix[n - 1 - i + s][s], end=' ')
            s += 1


for i in range(n):
      print(matrix[i][i], end=' ')
```

Задача 5 (20 баллов)

Условие

На плоскости разбросаны точки. Требуется составить такой выпуклый многоугольник с вершинами в части из этих точек, чтобы все остальные точки лежали внутри него.

Входные данные: в первой строке число N - количество точек, не превышающее 20. Далее в N строках через пробел координаты X и Y точек - вещественные числа, не превышающие по модулю 100, с точностью до двух знаков после запятой (которая выступает разделителем целой и дробной частей).

Выходные данные: номера точек (считаются с единицы), образующих искомый многоугольник, записанные в порядке обхода вершин по часовой стрелке и начинающиеся с самой верхней, записанные в одну строку через пробел. В случае, если верхних точек с одинаковой ординатой несколько, начинать вывод с любой из них.

Гарантируется, что искомый многоугольник существует.

Пример

Входные данные	Выходные данные
4 0 0 1 10,23 10,4 0 3 3,75	2 3 1

5 1 1 10,12 0,34 1 10 10 11 5 5	4 2 1 3

Проверочные тесты

Входные данные	Ожидаемый результат
4 0 0 1 10,23 10,4 0 3 3,75	2 3 1
5 1 1 10,12 0,34 1 10 10 11 5 5	4 2 1 3
3 0 0 5 -1 -5 -1	1 2 3
4 1 0 0 -1 -1 0 0 1	4 1 2 3
5 -1 -1 -2 -2 -3 -3 -10,95 -1 0 0	5 1 2 3 4

4 0,95 0,97 0,99 0,96 0,99 0,99 0,11 0,11	3 2 4 1

Пример решения

```
from collections import deque
from decimal import Decimal

n = int(input())

P = list(range(n))
A = []
for _ in range(n):
    x, y = input().split()
    A.append([Decimal(x.replace(',', '.')), Decimal(y.replace(',', '.'))])

for i in range(1, n):  # make first point in P the point with lowest X
    x1 = A[P[i]][0]
    x2 = A[P[0]][0]
    if x1 < x2:
    P[i], P[0] = P[0], P[i]


def rotate(a, b, c):  # >0? c on left side of ab; <0? on right side
    return (b[0] - a[0]) * (c[1] - b[1]) - (b[1] - a[1]) * (c[0] - b[0])


for i in range(2, n):
    j = i
    while j > 1 and (rotate(A[P[0]], A[P[j-1]], A[P[j]]) < 0):
    P[j], P[j-1] = P[j-1], P[j]
    j -= 1

S = [P[0], P[1]]
for i in range(2, n):
    while rotate(A[S[-2]], A[S[-1]], A[P[i]]) < 0:
    del S[-1]
    S.append(P[i])

S = list(reversed(S))
highest_y = 0
for p in S:
    highest_y = max(highest_y, A[p][1])

while A[S[0]][1] != highest_y:
    S.append(S.pop(0))

print(' '.join(str(y + 1) for y in S))
```

Задача 6 (24 балла)

Условие

В школьном музее вычислительной техники ребята решили включить старый запылившийся компьютер из первых серий персональных ЭВМ и написать для него какую-нибудь программу на ассемблере.

Особенность компьютера заключается в том, что он поддерживает только текстовый режим работы видеоадаптера. Это значит, что в памяти отдельно хранятся битовые карты представления символов на дисплее (с помощью них можно настроить используемый шрифт) и отдельно - коды символов, которые соответствуют этим картам.

Одному из учеников стало интересно, какая самая длинная замкнутая последовательность пикселей (контур) образуется на экране для заданного шрифта и определённого набора символов, выведенного на монитор.

Известно, что возможных символов в шрифте может быть не более 100, а битовая карта одного символа имеет размер 16x8 пикселей. Единичный бит означает, что пиксель включён на мониторе (отображается белым цветом), 0 - выключен (чёрный цвет).

Входные данные: в первой строке записано натуральное число K (не превышает 100) - количество используемых символов шрифта, и через пробел натуральные N и M (не превышают 10 каждое) - размеры фрагмента дисплея, на котором нужно найти самый длинный контур. Далее записано K матриц из цифр 0 и 1, каждая из 16 строк и 8 цифр в строке.

Затем записано N строк по M чисел через пробел, где каждое число лежит в диапазоне от 1 до K и является кодом символа, соответствующего определённой матрице.

Выходные данные: количество пикселей в самом длинном замкнутом контуре или число 0, если такого контура нет.

Контуром считается непрерывная последовательность белых пикселей по вертикали, горизонтали или диагонали.

Пример

Входные данные	Выходные данные
2 2 2 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000110 01001010 01001010 01001010 01010010 01010010 01010010 01100010 01000010 01000010 01000010 01000010 1 1 2 1	61

Проверочные тесты

Входные данные	Ожидаемый результат
2 2 2 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000110 01001010 01001010 01001010 01010010 01010010 01010010 01100010 01000010 01000010 01000010 01000010 1 1 2 1	61
1 1 1 00000000 00111100 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 1	0

1 2 1 01000010 01111110 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01111110 01000010 1 1	16
1 2 1 01000010 01111110 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01111110 00000000 01111110 01000010 1 1	28
1 2 1 01000010 01111110 00000000 01111110 01000010 01111110 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 1 1	30

1 1 2 00000000 11000011 01000010 01000010 11000011 00000000 01111110 01000010 01000010 01111110 00000000 00000000 00000000 00000000 00000000 00000000 1 1	16
1 1 2 00000000 11110111 00010100 00010100 11110111 00000000 00011100 00010100 00010100 00011100 00000000 00000000 00000000 00000000 00000000 00000000 1 1	18
1 2 2 01000010 11000011 00000000 00000000 00000000 00000000 00000000 00111110 01000010 01000010 01000010 01111110 00000000 00000000 11000011 01000010 1 1 1 1	17

1 2 2 00100100 00100100 11100111 00000000 00000000 00000000 00000000 00111000 00101000 00111000 00000000 00000000 00000000 11100111 00100100 00100100 1 1 1 1	20

Пример решения

```
#include <bits/stdc++.h>

using namespace std;

#define all(x) x.begin(), x.end()

#ifdef LOCAL

#else
#endif

using ll = long long;
using pii = pair<int, int>;
using pll = pair<ll, ll>;
using ld = long double;

#define double ld
#define ft first
#define sd second

ld x0 = 1e7;
ld y2 = 1e7;

vector<vector<int>> gr;
vector<vector<bool>> used;
int res = 0;
pii st;
int cur = 0;

void dfs(int x, int y){
```

```
	used[x][y] = true;
	cur++;
	for (int i = max(x - 1, 0); i < min((int)gr.size(), x + 2); i++){
	for (int j = max(y - 1, 0); j < min((int)gr[0].size(), y + 2); j++){
		if (i == x && j == y) continue;
		if (st == pii({i, j}) && cur > 2){
			res = max(res, cur);
		}
		if (!used[i][j] && gr[i][j] == 1) {
			dfs(i, j);
		}
	}
	}
	cur--;
	used[x][y] = false;
}

void solve(){
	int k, n, m;
	cin >> k >> n >> m;
	vector<vector<vector<int>>> cr(k);
	for (int i = 0; i < k; i++){
	cr[i].resize(16, vector<int>(8));
	for (int j = 0; j < 16; j++){
		string s;
		cin >> s;
		for (int l = 0; l < 8; l++) cr[i][j][l] = s[l] - '0';
	}
	}
	gr.resize(16 * n, vector<int>(8 * m));
	used.resize(16 * n, vector<bool>(8 * m));
	for (int i = 0; i < n; i++){
	for (int j = 0; j < m; j++){
		int nm;
		cin >> nm;
		for (int i1 = 0; i1 < 16; i1++){
			for (int j1 = 0; j1 < 8; j1++){
				gr[16 * i + i1][8 * j + j1] = cr[nm - 1][i1][j1];
			}
		}
	}
	}
	for (int i = 0; i < gr.size(); i++){
	for (int j = 0; j < gr[0].size(); j++){
		if (gr[i][j] == 1) {
			st = {i, j};
			dfs(i, j);
		}
	}
	}
	cout << res << endl;
}
```

```
int main() {
#if LOCAL
#else
      //freopen("inputik.txt", "r", stdin);
      //freopen("outputik.txt", "w", stdout);
#endif
      ios::sync_with_stdio(false);
      cin.tie(nullptr);
      cout.tie(nullptr);
      solve();
}
```

Программирование, отборочный этап, 11 класс

Вариант 2

Задача 1 (8 баллов)

Условие

Написать программу, которая определит, насколько изменится число, если минимальную по значению цифру, входящую в него, увеличить на 3.

Входные данные: натуральное число в десятичной системе счисления, не превышающее 10^9. В числе не может быть цифр 7, 8 и 9.

Выходные данные: разность между числом, полученным в соответствии с заданием, и исходным числом.

Пример

Входные данные	Выходные данные
1234	3000
163205	30

Примечание: минимальная цифра в числе не повторяется.

Проверочные тесты

Входные данные	Ожидаемый результат
1234	3000
163205	30
13	30
31	3
324567899	30000000
987654320	3

Пример решения

```
#include <iostream>
#include <vector>


using namespace std;


int main() {
      string num;
      cin >> num;
      int minn = 10;
      int ext = -1;
      int c = 1;
      for (int i = num.size() - 1; i > -1; --i) {
      if (minn > num[i] - '0') {
            minn = num[i] - '0';
            ext = c;
      }
      c *= 10;
      }
      cout << ext * 3;
      return 0;
}
```

Задача 2 (12 баллов)

Условие

Для некоторого десятичного числа определить, сколько цифр 2 содержит его запись в четверичной системе счисления.
Входные данные: натуральное число в десятичной с/с, не превышающее 10^9.
Выходные данные: количество цифр 2 в 4-й с/с.

Пример

Входные данные	Выходные данные
455	0
158	2

Проверочные тесты

Входные данные	Ожидаемый результат
455	0
158	2
39	1
2	1
14199	0
46	2

Пример решения

```
var n, c, i: int64;
s, s1: string;
begin
  readln(n);
  while n<>0 do begin
     str(n mod 4, s1);
     s+=s1;
     n:=n div 4;
  end;
  for i:=1 to length(s) do begin
     if s[i]='2' then c+=1;
  end;
  write(c);
end.
```

Задача 3 (16 баллов)

Условие

Петя изучает алгоритмы сортировки. Чтобы лучше разобраться в работе сортировки пузырьком, он решил упорядочить массив букв не по алфавиту, а в некоторой произвольной последовательности (например, в порядке расположения букв на клавиатуре, или любой другой). Также Петя хочет узнать, сколько перестановок символов для выполнения сортировки требуется произвести.

Входные данные: в первой строке записано натуральное N, не превышающее 26 - количество букв латинского алфавита, которые могут встречаться в сортируемом массиве. Далее в N строках через пробел записаны заглавная латинская буква и число, соответствующее порядку этой буквы с точки зрения Пети. Далее - строка из заглавных латинских букв, по длине не превышающая 100, для которой требуется посчитать число перестановок при сортировке пузырьком по убыванию.

Выходные данные: число перестановок элементов.

Пример

Входные данные	Выходные данные	Перестановки
3 B 1 A 2 Z 3 BAZ	3	AZB - 2 ZAB - 1
5 A 4 C 5 D 7 X 10 Y 1 ACDAX	7	CDAXA - 3 DCXAA - 2 DXCAA - 1 XDCAA - 1

Проверочные тесты

Входные данные	Ожидаемый результат
3 B 1 A 2 Z 3 BAZ	3
5 A 4 C 5 D 7 X 10 Y 1 ACDAX	7
1 A 1 A	0

2 A 1 B 2 AA	0
4 A 1 B 2 C 4 D 3 ABABA	3
5 E 5 D 4 C 3 B 2 A 1 ABCDE	10

Пример решения

```
#include <iostream>
#include <vector>
#include <map>


using namespace std;


map<char, int> mapp;


int main() {
      int num_of_sym;
      cin >> num_of_sym;
      for (int i = 0; i < num_of_sym; ++i) {
      char sym;
      cin >> sym;
      int num;
      cin >> num;
      mapp[sym] = num;
      }
      string s;
      cin >> s;
      int s_size = s.size();
      int c = 0;
      for (int i = 0; i < s_size - 1; ++i) {
      for (int j = 0; j < s_size - 1 - i; ++j) {
            if (mapp[s[j]] < mapp[s[j + 1]]) {
                  swap(s[j], s[j + 1]);
                  ++c;
            }
      }
```

```
    }
    cout << c;
    return 0;
}
```

Задача 4 (20 баллов)

Условие

Дана квадратная целочисленная матрица. Требуется переписать все её элементы в одномерный массив по следующему правилу: выбор элементов начинается из правого верхнего и левого нижнего угла и продолжается "змейкой" в сторону главной диагонали. На первой итерации после угловых элементов выбор производится в сторону правого нижнего угла. В конце записываются элементы главной диагонали от левого верхнего угла до правого нижнего.
Входные данные: в первой строке число N - количество строк и столбцов в квадратной матрице, N не превышает 20. Далее в N строках по N чисел через пробел, каждое не превышает 100 по модулю.
Выходные данные: получившийся массив, записанный в одну строку через пробел.

Пример

Входные данные	Выходные данные
3 7 5 1 8 3 6 4 9 2	1 2 3 4 5 6 7 8 9
4 13 7 5 1 14 9 3 8 10 15 11 6 4 12 16 2	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Проверочные тесты

Входные данные	Ожидаемый результат
3 7 5 1 6 8 3 2 4 9	1 2 3 4 5 6 7 8 9
4 13 7 5 1 8 14 9 3 6 10 15 11 2 4 12 16	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 5	5
2 1 2 3 4	2 3 1 4
5 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 -1 -2 -3 -4 -5	5 -1 0 -2 4 6 3 1 9 7 5 -3 0 -4 4 8 8 2 2 6 1 7 3 9 -5

Пример решения

```
n = int(input())
matr = [list(map(int, input().split())) for i in range(n)]
ans = []
pick = 1
while pick < n:
      for i in range(pick):
      ans.append(matr[i][n-pick + i])
      ans.append(matr[n-pick + i][i])

      if pick != n - 1:
      for i in range(pick, -1, -1):
            ans.append(matr[i][n-pick-1+i])
            ans.append(matr[n-pick-1+i][i])

      pick += 2

for i in range(n):
      ans.append(matr[i][i])

print(*ans)
```

Задача 5 (20 баллов)

Условие

На плоскости разбросаны точки. Требуется составить такой выпуклый многоугольник с вершинами в части из этих точек, чтобы все остальные точки лежали внутри него.

Входные данные: в первой строке число N - количество точек, не превышающее 20. Далее в N строках через пробел координаты X и Y точек - вещественные числа, не превышающие по модулю 100, с точностью до двух знаков после запятой (которая выступает разделителем целой и дробной частей).

Выходные данные: номера точек (считаются с единицы), образующих искомый многоугольник, записанные в порядке обхода вершин против часовой стрелки и начинающиеся с самой верхней, записанные в одну строку через пробел. В случае, если верхних точек с одинаковой ординатой несколько, начинать вывод с любой из них.

Гарантируется, что искомый многоугольник существует.

Пример

Входные данные	Выходные данные
4 0 0 1 10,23 10,4 0 3 3,75	2 1 3
5 1 1 10,12 0,34 1 10 10 11 5 5	4 3 1 2

Проверочные тесты

Входные данные	Ожидаемый результат

4 0 0 1 10,23 10,4 0 3 3,75	2 1 3
5 1 1 10,12 0,34 1 10 10 11 5 5	4 3 1 2
3 0 0 5 -1 -5 -1	1 3 2
4 1 0 0 -1 -1 0 0 1	4 3 2 1
5 -1 -1 -2 -2 -3 -3 -10,95 -1 0 0	5 4 3 2 1
4 0,95 0,97 0,99 0,96 0,99 0,99 0,11 0,11	3 1 4 2

Пример решения

```
def f2(A,B,C):
      return (B[0]-A[0])*(C[1]-B[1])-(B[1]-A[1])*(C[0]-B[0])
def f1(A):
      n = len(A)
      lst1 = list(range(n))
      for i in range(1,n):
      if A[lst1[i]][1]>A[lst1[0]][1]:
            lst1[i], lst1[0] = lst1[0], lst1[i]
      lst2 = [lst1[0]]
      del lst1[0]
      lst1.append(lst2[0])
      while True:
      right = 0
      for i in range(1,len(lst1)):
```

```
            if f2(A[lst2[-1]],A[lst1[right]],A[lst1[i]])<0:
                  right = i
      if lst1[right]==lst2[0]:
            break
      else:
            lst2.append(lst1[right])
            del lst1[right]
      for i in range(len(lst2)):
      lst2[i]+=1
      return lst2

a=[]
for i in range(int(input())):
      k= input().split()
      if ',' in k[0]:
      k[0]=k[0].split(',')[0]+'.'+k[0].split(',')[1]
      if ',' in k[1]:
      k[1]=k[1].split(',')[0]+'.'+k[1].split(',')[1]
      a.append([float(k[0]), float(k[1])])
print(*f1(a))
```

Задача 6 (24 балла)

Условие

В школьном музее вычислительной техники ребята решили включить старый запылившийся компьютер из первых серий персональных ЭВМ и написать для него какую-нибудь программу на ассемблере.

Особенность компьютера заключается в том, что он поддерживает только текстовый режим работы видеоадаптера. Это значит, что в памяти отдельно хранятся битовые карты представления символов на дисплее (с помощью них можно настроить используемый шрифт) и отдельно - коды символов, которые соответствуют этим картам.

Одному из учеников стало интересно, какая самая длинная замкнутая последовательность пикселей (контур) образуется на экране для заданного шрифта и определённого набора символов, выведенного на монитор.

Известно, что возможных символов в шрифте может быть не более 100, а битовая карта одного символа имеет размер 16x8 пикселей. Единичный бит означает, что пиксель включён на мониторе (отображается белым цветом), 0 - выключен (чёрный цвет).

Входные данные: в первой строке записано натуральное число K (не превышает 100) - количество используемых символов шрифта, и через пробел натуральные N и M (не превышают 10 каждое) - размеры фрагмента дисплея, на котором нужно найти самый длинный контур. Далее записано K матриц из цифр 0 и 1, каждая из 16 строк и 8 цифр в строке.

Затем записано N строк по M чисел через пробел, где каждое число лежит в диапазоне от 1 до K и является кодом символа, соответствующего определённой матрице.

Выходные данные: количество пикселей в самом длинном замкнутом контуре или число 0, если такого контура нет.

Контуром считается непрерывная последовательность белых пикселей по вертикали, горизонтали или диагонали.

Пример

Входные данные	Выходные данные
2 2 2 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000110 01001010 01001010 01001010 01010010 01010010 01010010 01100010 01000010 01000010 01000010 01000010 1 1 2 1	61

Проверочные тесты

Входные данные	Ожидаемый результат
2 2 2 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000110 01001010 01001010 01001010 01010010 01010010 01010010 01100010 01000010 01000010 01000010 01000010 1 1 2 1	61

1 1 1 00000000 00111100 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 1	0
1 2 1 01000010 01111110 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01111110 01000010 1 1	16
1 2 1 01000010 01111110 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01111110 00000000 01111110 01000010 1 1	28

1 2 1 01000010 01111110 00000000 01111110 01000010 01111110 00000000 01111110 01000010 01000010 01000010 01000010 01000010 01000010 01000010 01000010 1 1	30
1 1 2 00000000 11000011 01000010 01000010 11000011 00000000 01111110 01000010 01000010 01111110 00000000 00000000 00000000 00000000 00000000 00000000 1 1	16
1 1 2 00000000 11110111 00010100 00010100 11110111 00000000 00011100 00010100 00010100 00011100 00000000 00000000 00000000 00000000 00000000 00000000 1 1	18

1 2 2 01000010 11000011 00000000 00000000 00000000 00000000 00000000 00111110 01000010 01000010 01000010 01111110 00000000 00000000 11000011 01000010 1 1 1 1	17
1 2 2 00100100 00100100 11100111 00000000 00000000 00000000 00000000 00111000 00101000 00111000 00000000 00000000 00000000 11100111 00100100 00100100 1 1 1 1	20

Пример решения

```
#include <bits/stdc++.h>

using namespace std;

#define all(x) x.begin(), x.end()

#ifdef LOCAL

#else
#endif

using ll = long long;
```

```
using pii = pair<int, int>;
using pll = pair<ll, ll>;
using ld = long double;

#define double ld
#define ft first
#define sd second

ld x0 = 1e7;
ld y2 = 1e7;

vector<vector<int>> gr;
vector<vector<bool>> used;
int res = 0;
pii st;
int cur = 0;

void dfs(int x, int y){
      used[x][y] = true;
      cur++;
      for (int i = max(x - 1, 0); i < min((int)gr.size(), x + 2); i++){
      for (int j = max(y - 1, 0); j < min((int)gr[0].size(), y + 2); j++){
            if (i == x && j == y) continue;
            if (st == pii({i, j}) && cur > 2){
                  res = max(res, cur);
            }
            if (!used[i][j] && gr[i][j] == 1) {
                  dfs(i, j);
            }
      }
      }
      cur--;
      used[x][y] = false;
}

void solve(){
      int k, n, m;
      cin >> k >> n >> m;
      vector<vector<vector<int>>> cr(k);
      for (int i = 0; i < k; i++){
      cr[i].resize(16, vector<int>(8));
      for (int j = 0; j < 16; j++){
            string s;
            cin >> s;
            for (int l = 0; l < 8; l++) cr[i][j][l] = s[l] - '0';
      }
      }
      gr.resize(16 * n, vector<int>(8 * m));
      used.resize(16 * n, vector<bool>(8 * m));
      for (int i = 0; i < n; i++){
      for (int j = 0; j < m; j++){
            int nm;
```

```
            cin >> nm;
            for (int i1 = 0; i1 < 16; i1++){
                  for (int j1 = 0; j1 < 8; j1++){
                        gr[16 * i + i1][8 * j + j1] = cr[nm - 1][i1][j1];
                  }
            }
      }
      }
      for (int i = 0; i < gr.size(); i++){
      for (int j = 0; j < gr[0].size(); j++){
            if (gr[i][j] == 1) {
                  st = {i, j};
                  dfs(i, j);
            }
      }
      }
      cout << res << endl;
}

int main() {
#if LOCAL
#else
      //freopen("inputik.txt", "r", stdin);
      //freopen("outputik.txt", "w", stdout);
#endif
      ios::sync_with_stdio(false);
      cin.tie(nullptr);
      cout.tie(nullptr);
      solve();
}
```

КРИТЕРИИ ОЦЕНИВАНИЯ ОЛИМПИАДНЫХ ЗАДАНИЙ ОТБОРОЧНОГО ЭТАПА

Максимальная сумма баллов за выполненные задания варианта – 100.

Распределение баллов по заданиям:

Номер задачи	1	2	3	4	5	Ситационная задача
Баллы	8	12	16	20	20	24