Заключительный этап, 9 класс (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (1 балл) [XYZ_2024]

Дано выражение:

$$XYZ_9 + ZXY_{12} + YZX_{14} = 2024_{10}$$

В данном выражении X, Y и Z – допустимые различные цифры указанных систем счисления. Определите значения переменных Y и Z, если X = 5. В ответе укажите через пробел 2 целых положительных числа: сначала значение Y и затем значение Z.

Ответ: 6 2

Решение:

Так как Y и Z – допустимые цифры девятеричной системы счисления, то $0 \le Y, Z \le 8$

Переведём все числа данного выражения в десятичную систему:

$$XYZ_9 = 5 * 9 * 9 + Y * 9 + Z = 405 + 9Y + Z$$
$$ZXY_{12} = Z * 12 * 12 + 12 * 5 + Y = 144Z + 60 + Y$$
$$YZX_{14} = Y * 14 * 14 + Z * 14 + 5 = 196Y + 14Z + 5$$

Просуммируем слагаемые левой части:

$$405 + 9Y + Z + 144Z + 60 + Y + 196Y + 14Z + 5 = 206Y + 159Z + 470$$

Приравняем левую и правую часть, упростим:

$$206Y + 159Z + 470 = 2024$$
$$206Y + 159Z = 1554$$

Теперь необходимо более точно определить границы допустимых значений для Y и Z.

Во-первых,

$$1554 = 206Y + 159Z \ge 159(Y + Z)$$

Значит, $(Y + Z) \le 9$.

Во-вторых,

$$1554 = 206Y + 159Z \le 206(Y + Z)$$

Значит, $(Y + Z) \ge 8$.

В-третьих, так как число 1554 чётное, то либо оба слагаемых слева нечётные, либо оба чётные. Так как число 206Y не может быть нечётным вне зависимости от значения Y, то оба слагаемых чётные. Число 159Z чётное только при чётных значениях Z. Значит, возможные значения для числа 159Z – это 0, 318, 636, 954 и 1272. Соответствующие этим значениям значения для 206Y – 1554, 1236, 918, 600, 282. Заметим, что число 206Y делится нацело на 206. Из перечисленных 5 чисел таким свойством обладает только число 1236. Значит, Y=1236/206=6, Z=2. Ответ: 6 2.

Также возможно и решение задачи программированием, например, так:

```
from itertools import product
for y, z in product(range(9), repeat=2):
    num = int(f'5{y}{z}', 9) + int(f'{z}5{y}', 12) + int(f'{y}{z}5', 14)
    if y != z and num == 2024:
        print(y, z)
```

2. Кодирование информации. Объем информации (2 балла) [Canis aureus]

Для кодирования цветов часто используется RGB-палитра. В этом случае цвет каждого пикселя изображения кодируется с помощью трех отдельных числовых значений, называемыми цветовыми каналами (красный (R), зеленый (G) и синий (B)). Числовое значение по каждому цветовому каналу в общем случае может быть в диапазоне от 0 до 255. При сохранении каждого числового значения в памяти для него отводится минимальное, одинаковое для всех значений количество бит. Илья решил, что может хранить изображения используя меньше памяти.

Во-первых, он применил к каждому пикселю изображения следующий алгоритм сжатия:

1. Если значения всех цветовых каналов имеют одинаковую чётность - цвет не меняется.
2. Если только одно из значений нечётное - оно уменьшается на 1.
3. Если только одно из значений чётное - оно увеличивается на 1.

Во-вторых, он решил, что после сжатия будет отводить минимальное, одинаковое для всех значений количество бит не значению каждого цветового канала пикселя, а коду, который присваивается уникальной комбинации значений всех цветовых каналов каждого пикселя. Уникальной является комбинация, отличающаяся от любой другой значением хотя бы одного цветового канала.

Теперь Илья хочет оценить эффективность своей идеи. Определите, на сколько байт уменьшилось количество памяти, необходимое для хранения изображения размером 1920 на 1080 пикселей. Если необходимое количество памяти увеличилось – запишите в ответ отрицательное число. Например, если раньше требовалось 100 байт, а после преобразований потребуется 98 байт, то ответ будет равен 2, а если потребуется 102 байта, то ответ будет равен -2.

Ответ: 518400

Решение:

Определим, сколько бит необходимо для хранения одного пикселя до преобразований.

В задаче рассматривается кодирование значений минимально возможным одинаковым количеством бит,

следовательно, количество бит определяется по формуле Хартли: количество бит $= log_2$(количество значений), с округлением до ближайшего большего целого числа. Для хранения каждого цветового канала понадобится $log_2(255 - 0 + 1) = 8$ бит. Так как каналов 3, потребуется 24 бита на каждый пиксель.

Определим, сколько бит необходимо для хранения одного пикселя после преобразований.

В первую очередь вычислим, сколько возможных цветов (то есть, комбинаций из трёх значений цветовых каналов) было до сжатия. Так как каждый цветовой канал может иметь 256 различных значений, и значений цветовых каналов не зависят друг от друга, то возможных комбинаций $256 * 256 * 256 = 2^{24}$.

Теперь выясним, как изменится количество возможных цветов при сжатии. Численное значение каждого цветового канала можно представить в виде $2n + k$, где n – целое число от 0 до 127 включительно, а k равно 0 для чётного числа и 1 для нечётного. Значит, все возможные цвета (комбинации значений трёх цветовых каналов) можно разбить на 8 групп:

1. $(2n + 0, 2n + 0, 2n + 0)$
2. $(2n + 0, 2n + 0, 2n + 1)$
3. $(2n + 0, 2n + 1, 2n + 0)$
4. $(2n + 0, 2n + 1, 2n + 1)$
5. $(2n + 1, 2n + 0, 2n + 0)$
6. $(2n + 1, 2n + 0, 2n + 1)$
7. $(2n + 1, 2n + 1, 2n + 0)$
8. $(2n + 1, 2n + 1, 2n + 1)$

Применим к данным группам преобразования согласно алгоритму. Группы 1 и 8 не изменятся, так как все числа имеют одинаковую чётность. Группы 2, 3, 5 изменят значение на $(2n + 0, 2n + 0, 2n + 0)$, так как имеют только одно нечётное значение. То есть, перейдут в группу 1. Группы 4, 6, 7 изменят значение на $(2n + 1, 2n + 1, 2n + 1)$, так как имеют только одно чётное значение. То есть, перейдут в группу 8. Таким образом, из 8 групп мы получим только 2, то есть различных комбинаций станет в 4 раза меньше. До преобразования их было 2^{24}, т.е. теперь их будет 2^{22}. Для каждого пикселя мы храним информацию о его цвете, различных цветов у нас 2^{22}, значит потребуется по 22 бита на пиксель. То есть, на каждый пиксель требуется на 2 бита меньше.

Вычислим изменение необходимого количества памяти для всего изображения. Потребуется на $1920 * 1080 * 2$ бит меньше, т.е. на 4147200 бит = 518400 байт меньше.

3. Основы логики (1 балл)

[Четыре импликации]

Два набора значений переменных A, B и C называют не эквивалентными, если значение хотя бы одной переменной различается.

Сколько существует не эквивалентных друг другу наборов значений переменных, при которых равенство выполняется при любом значении D:

$$(((D \rightarrow (A \wedge B \wedge C)) \rightarrow (A \wedge \mathrm{B} \vee \mathrm{C})) \rightarrow (\mathrm{A} \vee \mathrm{B} \wedge \mathrm{C})) \rightarrow (A \vee \mathrm{B} \vee \mathrm{C}) = 1$$

В ответ укажите одно целое число.

Решение:

В первую очередь составим таблицу истинности для используемых выражений:

A	B	C	A∧B∧C	A∧B∨C	A∨B∧C	A∨B∨C
0	0	0	FALSE	FALSE	FALSE	FALSE
0	0	1	FALSE	TRUE	FALSE	TRUE
0	1	0	FALSE	FALSE	FALSE	TRUE
0	1	1	FALSE	TRUE	TRUE	TRUE
1	0	0	FALSE	FALSE	TRUE	TRUE
1	0	1	FALSE	TRUE	TRUE	TRUE
1	1	0	FALSE	TRUE	TRUE	TRUE
1	1	1	TRUE	TRUE	TRUE	TRUE

Рассмотрим таблицу истинности для импликации:

A	B	A->B
0	0	1
0	1	1
1	0	0
1	1	1

Заметим, что при B=1 результат импликации не зависит от A и равен 1. То есть, равенство $(\ldots) \rightarrow (A \vee \mathrm{B} \vee \mathrm{C}) = 1$ всегда верно при $(A \vee \mathrm{B} \vee \mathrm{C}) = 1$. Значит, 7 из 8 наборов значений A, B и C удовлетворяют условию. Проверим оставшийся набор. При A=0, B=0, C=0 выражение примет вид: $(((D \rightarrow 0) \rightarrow 0) \rightarrow 0) \rightarrow 0$. Рассмотрим значения этого выражения при D=0 и D=1

D	$D \rightarrow 0$	$(D \rightarrow 0) \rightarrow 0$	$((D \rightarrow 0) \rightarrow 0) \rightarrow 0$	$(((D \rightarrow 0) \rightarrow 0) \rightarrow 0) \rightarrow 0$
0	1	0	1	0

1	0	1	0	1

Таким образом, результат меняется в зависимости от D, что противоречит условию. То есть, 8-й набор не подходит, и ответ – 7.

4. Алгоритмизация и программирование. Формальные исполнители (2 балла)

[Бедная лошадка]

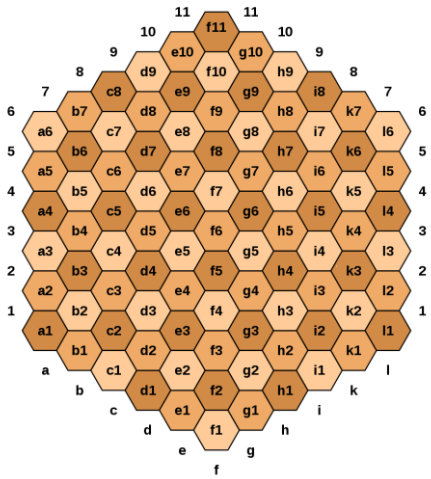
Недавно Ксюша узнала о существовании гексагональных шахмат. Больше всего её заинтересовали перемещения коней и слонов. Она задала набор дополнительных правил для перемещения коня:

1. Он может сделать сколько угодно корректных ходов.
2. Он не может наступать на клетки, находящиеся под боем у слонов (клетка находится под боем, если слон может попасть в неё за один корректный ход).
3. Он не может съедать слонов.
4. Слоны не могут двигаться.

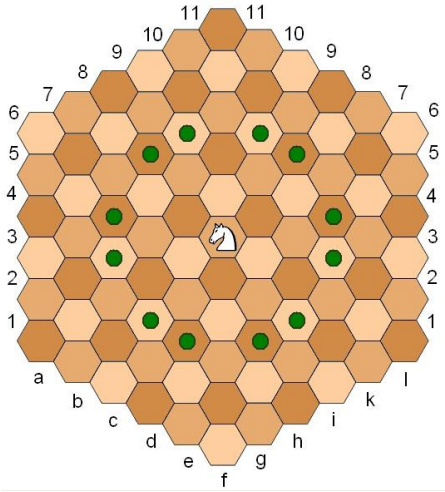
Ксюша решила определить, сколько есть клеток на поле, посещение которых конем не будет противоречить указанным условиям, но которые конь тем не менее не сможет посетить, если он будет начинать на клетке **e6**, а слоны будут стоять на клетках **d6**, **e9** и **i6**? В ответе напишите сначала количество клеток, а затем в возрастающем лексикографическом порядке через пробел эти клетки. Например, 3 a2 a3 b2. Если окажется, что таких клеток нет, напишите в ответ NULL.

Примечание:

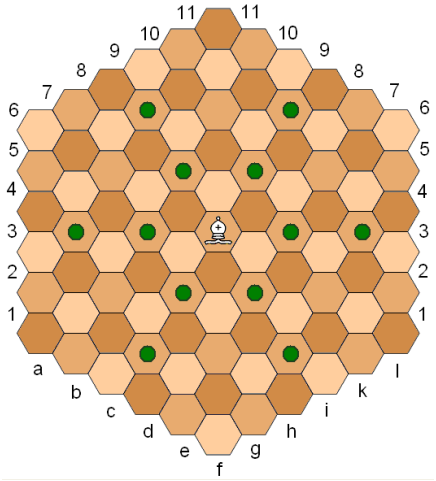
Нумерация клеток шахматной доски



Ход коня



Ход слона



Роль горизонталей выполняют косые линии полей, параллельные одной из невертикальных сторон доски, а роль диагоналей — линии полей одного цвета

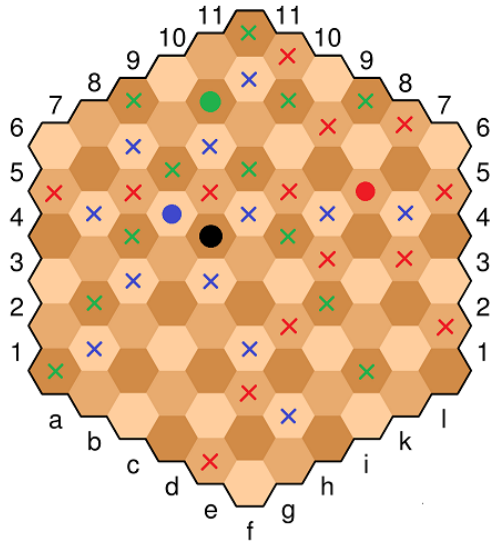
Конь ходит на два поля по вертикали или «горизонтали» и ещё на одно поле по другой вертикали или «горизонтали» (поворот на 120 градусов). Не обязательно, чтобы промежуточные клетки были свободны/достижимы, главное, чтобы были свободны начальная и конечная клетки.

Слон ходит на любое количество полей по диагоналям своего цвета.

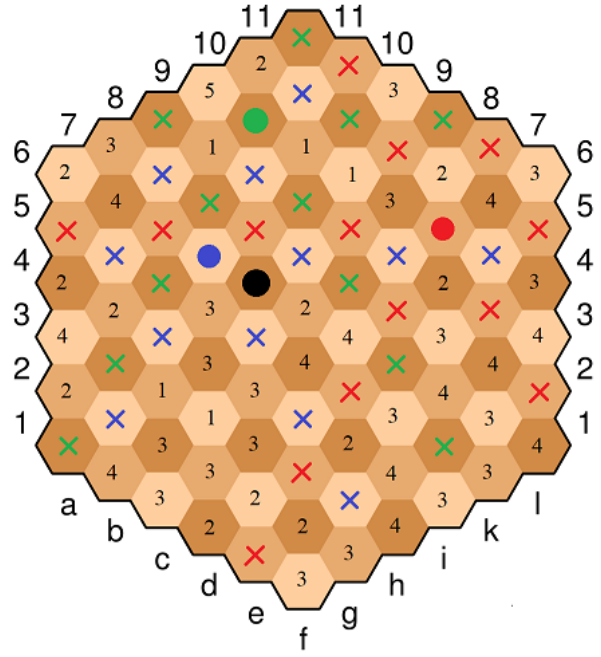
Ответ: NULL

Решение:

Для решения данной задачи необходимо проэмулировать возможные перемещения коня. Для начала обозначим стартовые позиции фигур. Конь обозначен чёрным кругом, слоны – синим, красным и зелёным кругами. Крестиками соответствующего цвета обозначим клетки, находящиеся под ударом у данного слона:



Теперь обозначим цифрой 1 все клетки, которые конь может достить из своего положения за 1 ход, цифрой 2 – те, что можно достичь за 1 ход из клеток, отмеченных цифрой 2, и так далее. Если клетка уже отмечена цифрой, менять её не будем.

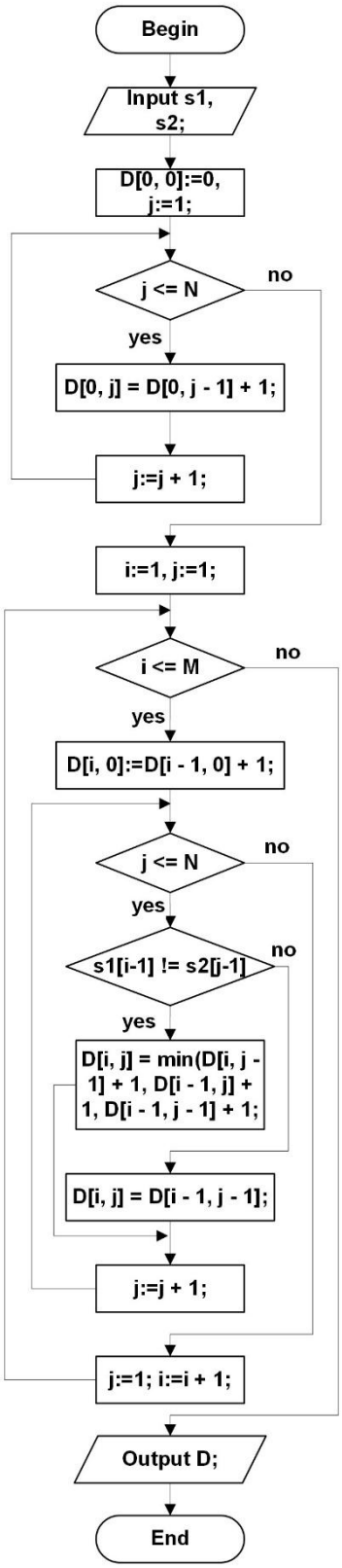


Как видно из рисунка, коню удалось посетить все клетки, разрешенные условиями, значит, ответ NULL.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (3 балла)

[Два не очень разных слова]

Приведена блок-схема алгоритма:



В данной диаграмме: s1, s2 – строки из строчных букв английского алфавита; M – длина строки s1; N – длина строки s2; D – матрица размером M+1 на N+1. Строки матрицы нумеруются от 0, сверху вниз. Столбцы нумеруются от 0, слева направо. Запись D[i, j] означает ячейку на пересечении строки i и столбца j. Символы строк также нумеруются от 0.

В результате выполнения была получена следующая матрица:

0 1 2 3 4 5 6
1 1 2 2 3 4 5
2 2 2 3 2 3 4
3 3 3 3 3 2 3
4 4 4 4 3 3 2
5 5 5 5 4 4 3

6 6 6 6 5 5 4
7 7 7 7 6 6 5

Определите, какой могла быть строка s2, если строка s1=”**monocle**” (символы кавычек не являются частью строки). В случае, если символ строки s2 не удаётся определить однозначно, замените его символом *. Например, если в строке из пяти символов первые два определить не удалось, а остальные 3 – это символы **y, m, p,** то в качестве ответа необходимо указать **ymp.

Ответ: **mono

Решение:

Приведенный алгоритм рассчитывает, сколько действий вида «заменить символ, вставить символ, удалить символ» требуется для преобразования одной строки в другую. (Заметим, что число преобразований не зависит от порядка строк). Значение в D[i, j] соответствует числу действий, необходимых для преобразования подстроки из первых i символов одной строки в подстроку из первых j символов другой.

В первую очередь необходимо понять, по какому принципу заполняется матрица D. Для начала заполняются 0-я строка и 0-й столбец:

0 1 2 3 4 5 6
1 ? ? ? ? ? ?
2 ? ? ? ? ? ?
3 ? ? ? ? ? ?
4 ? ? ? ? ? ?
5 ? ? ? ? ? ?
6 ? ? ? ? ? ?
7 ? ? ? ? ? ?

После чего остальная таблица заполняется по следующему принципу: если (i-1)-й символ строки s1 и (j-1)-й символ строки s2 равны, то в ячейку D[i, j] записывается значение из ячейки D[i-1, j-1]. Именно на это свойство нам следует опираться при определении совпадения символов. Если символы не совпадают, то мы выбираем наименьший из 3 вариантов: либо заменить текущий символ (тогда нам понадобится D[i-1, j-1]+1 действий), либо удалить его (тогда нам понадобится D[i-1, j]+1 действий), либо добавить символ (тогда нам понадобится D[i, j-1]+1 действий). Обратите внимание, что если в ячейках D[i-1, j-1] и D[i, j] содержатся одинаковые значения, но при этом одно или оба значения D[i-1, j], D[i, j-1] равняются D[i, j]-1, то однозначно определить, были ли символы s1[i-1] и s2[j-1] нельзя.

Пользуясь данными знаниями, выделим в результирующей таблице все D[i, j], которые равны D[i-1, j-1]:

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

Отсечём те из них, «происхождение» которых нельзя определить однозначно (значение слева или сверху на единицу меньше):

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

Оставшиеся выделенными символы являются результатом равенства символов. Для удобства допишем слева от таблицы слово s1.

0 1 2 3 4 5 6
m 1 1 2 2 3 4 5
o 2 2 2 3 2 3 4
n 3 3 3 3 3 2 3
o 4 4 4 4 3 3 2
c 5 5 5 5 4 4 3
l 6 6 6 6 5 5 4
e 7 7 7 7 6 6 5

Обозначим все неизвестные символы строки s2 звёздочками и будем заменять по мере выяснения однозначно определяемых символов. Так как число столбцов на 1 больше числа символов строки s2, то s2=******

Т.к. D[1, 3] написано в результате равенства s1[0] и s2[2], s2=**m***

Т.к. D[2, 4] написано в результате равенства s1[1] и s2[3], s2=**mo**

Т.к. D[3, 5] написано в результате равенства s1[2] и s2[4], s2=**mon*

Т.к. D[4, 6] написано в результате равенства s1[3] и s2[5], s2=**mono

Значение D[4, 4] также написано в результате равенства, но s2[3] мы уже установили из D[2, 4].

Оставшиеся символы однозначно установить невозможно, значит ответ **mono.

6. Телекоммуникационные технологии (1 балл)

[Локальное потепление]

Некоторый сервер с недостаточным охлаждением умеет обрабатывать запросы двух типов. Запрос типа А он обрабатывает за 8 секунд, а запрос типа Б – за 12 секунд. Однако, за каждую секунду, во время которой сервер обрабатывает запрос (вне зависимости от типа запроса), его температура увеличивается на 2 градуса. Если температура сервера достигнет 140 градусов, он необратимо выйдет из строя. Для того чтобы избежать этого, системный администратор решил после обработки каждого запроса (вне зависимости от типа) делать перерыв в X секунд. За каждую секунду перерыва температура сервера снижается на 0.5 градуса. Определите минимально возможное натуральное значение X, при котором сервер никогда не нагреется до 140 градусов, если запросы приходят в следующем порядке: А, А, Б, А, А, Б, А, А, Б и т.д. Температура сервера на момент начала работы – 50 градусов.

Ответ: 38

Решение:

Заметим, что порядок появления запросов состоит из повторяющихся последовательностей А, А, Б. Очевидно, что если спустя 3 таких запроса и 3 перерыва (после каждого из запросов) суммарное изменение температуры будет положительным, то рано или поздно процессор нагреется до 140 градусов и выйдет из строя. Составим выражение, описывающее изменение температуры за такой период: суммарная длительность процессов А, А, Б составит 8+8+12=28 секунд. То есть, за это время от нагреется на 28*2=56 градусов. А остынет – на 3*X*0.5=1.5X. Значит, изменение температуры составит 56-1.5X. Так как процессор не должен нагреваться, $56 - 1.5X <= 0$. То есть, $X \geq 37,(3)$. Наименьшее натуральное число, удовлетворяющее данному неравенству – 38.

7. Технологии обработки информации в электронных таблицах, технологии сортировки и фильтрации данных (2 балла)

[Остаться самим собой]

Дана таблица в режиме отображения формул:

	A	B	C	D	E	F	G
1		2	3	4	5	6	
2	2	=IF(MOD(A1; POW(B$1; $A2))=0; DIVIDE($A$1; POW(B$1; $A2)); MOD($A$1; POW(B$1; $A2)))					
3	3						
4	4						
5	5						
6	6						
7	7						
8	8						

Формулу из ячейки B2 скопировали во все ячейки диапазона B2:F8. После чего оказалось, что число из ячейки A1 встречается в диапазоне B2:F8 ровно 4 раза. Определите минимальное и максимальное возможное число в ячейке A1, при котором это могло произойти. В ответ запишите сначала меньшее из значений, а затем большее из них.

Примечание: таблица соответствия имён используемых функций.

	Google Sheets	**Excel**	**LibreOffice**
Условное вычисление	IF	ЕСЛИ	IF
Возведение в степень	POW	СТЕПЕНЬ	POWER
Остаток от деления	MOD	ОСТАТ	MOD
Частное от целочисленного деления	DIVIDE	ЧАСТНОЕ	QUOTIENT

Ответ: 65536 78124

Решение:

Обозначим как X величину POW(B$1; $A2). Формула из ячейки B2 записывает в ячейку либо остаток от деления числа из ячейки A1 на X, если число из ячейки A1 делится не на X и результат деления в обратном случае. Следовательно, число из ячейки A1 (для краткости далее будем называть его просто A1) может попасть в одну из ячеек из диапазона B2:F8 двумя способами: либо если A1 делится на X и при этом A1/X=A1, т.е. X = 1, что не выполняется в данной таблице. Либо A1 не делится на X и тогда в ячейке будет значение A1 mod X. Данная величина будет равна A1 только в том случае, если A1 меньше X. То есть, A1 встретится в диапазоне B2:F8 четыре раза, если ровно 4 возможных значения POW(B$1; $A2) будут больше A1. Для каждого числа из диапазона B2:F8 определим значение X:

	A	B	C	D	E	F
1		2	3	4	5	6
2	2	4	9	16	25	36
3	3	8	27	64	125	216
4	4	16	81	256	625	1296
5	5	32	243	1024	3125	7776
6	6	64	729	4096	15625	46656
7	7	128	2187	16384	78125	279936
8	8	256	6561	65536	390625	1679616

Если A1 будет больше либо равно 78125, то только 3 числа в таблице будут больше A1. Значит, максимально возможное значение – 78124. Если A1 будет 65535 или меньше, то 5 чисел таблицы будут больше A1. Значит, A1 должно быть хотя бы 65536. Значит, ответ на данную задачу 65536 78124.

8. Технологии программирования (3 балла)

[Окна]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	1 секунда
Ограничение по памяти	256 мегабайт

На экране монитора размером $w \times h$ расположены n прямоугольных окон нового текстового редактора. Размер экрана измеряется в символах, то есть текстовая строка длины w символов занимает всю ширину экрана от левого края до правого, а всего на экране могут поместиться друг под другом ровно h таких строк. Занумеруем строки экрана от 1 до h, а столбцы — от 1 до w, и будем обозначать координатами (i, j) позицию символа на пересечении i-й строки и j-го столбца.

Каждое из n окон содержит только текстовое поле, которое по ширине и высоте помещает в себе целое число символов. Окна могут быть разного размера и расположены так, что каждая из $w \times h$ позиций символов на экране принадлежит текстовому полю ровно одного окна. Иными словами, окна не могут накладываться, но могут касаться границами, и в совокупности покрывают всю площадь экрана.

Изначально все текстовые поля в окнах пустые, то есть не содержат никакой текст. Команды вывода строки на экран задаются тройками вида (r, c, s), означающими, что надо поставить курсор на c-ю слева позицию в r-ю сверху строку монитора и, начиная с этой позиции, напечатать строку s. Символы строки печатаются по очереди слева направо, каждый следующий символ занимает следующую позицию в той же строке и записывается в ней вместо того, что стояло на этой позиции ранее. Так происходит, пока курсор не дойдет до правой границы текущего окна, после чего поведение курсора зависит от *режима*, в котором работает то окно, в котором он находится.

Окна могут работать в трех режимах. При достижении курсором правой границы окна:

В режиме «clip» вывод останавливается, и оставшаяся часть строки просто не выводится.

В режиме «wrap» курсор переносится на первую (ближайшую к левой границе окна) позицию **этого же окна** в следующей строке, после чего вывод продолжается. Если же текущая строка является последней строкой в текущем окне, вывод останавливается;

В режиме «overflow», если сейчас курсор находится в позиции экрана (r, c), он переносится в позицию $(r, c + 1)$, то есть в следующую позицию на экране, оставаясь в той же строке и игнорируя правую границу окна; после чего вывод продолжается. Если же текущая позиция является последней позицией на экране, то есть $c = w$, курсор переносится на первую позицию на экране в следующей строке $(r + 1,1)$. Если и строка была последняя, то есть курсор достиг правой-нижней позиции на экране ($r = h$ и $c = w$), вывод останавливается.

Для каждого окна известно, в каком режиме оно изначально работает. Обработайте команды вывода строк и команды изменения режимов окон и выведите состояние экрана после обработки всех команд.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке ввода дано единственное целое число t — количество наборов входных данных в тесте ($1 \le t \le 50$).

Каждый набор входных данных начинается со строки, в которой даны три целых числа n, w и h — количество окон на экране и размеры экрана, соответственно ($1 \le w, h; 1 \le n \le w \cdot h \le 2000$).

Следующие n строк набора входных данных описывают окна; i-я строка описывает i-е окно и содержит через пробел четыре целых числа $r_{i,1}$, $c_{i,1}$, $r_{i,2}$ и $c_{i,2}$ — номер строки и столбца левого-верхнего угла окна и номер строки и столбца правого-нижнего угла окна ($1 \le r_{i,1} \le r_{i,2} \le h$; $1 \le c_{i,1} \le c_{i,2} \le w$), а также строку mode_i, равную «clip», «wrap» или «overflow» — изначальный режим, в котором работает i-е окно. Гарантируется, что весь экран покрыт окнами и что каждая позиция на экране принадлежит ровно одному окну.

Затем в отдельной строке следует целое число q — количество команд, которые вам предстоит обработать ($1 \le q \le 2000$).

В i-й из следующих q строк дано описание i-й команды в формате

«print r_i c_i s_i», если это команда вывода строки s_i, начиная с позиции (r_i, c_i) ($1 \le r_i \le h$; $1 \le c_i \le w$; $1 \le |s_i|$);

«mode t_i m_i», если это команда изменения режима окна номер t_i на m_i ($1 \le t_i \le n$; $m_i \in$ «clip», «wrap», «overflow»).

Гарантируется, что все s_i состоят из маленьких букв латинского алфавита (символы от ‘a’ до ‘z’), и что сумма длин s_i по всем командам в рамках одного набора входных данных не превосходит 10 000.

Формат выходных данных

Для каждого набора входных данных выведите состояние экрана после обработки всех q команд. Состояние экрана — это h строк по w символов, каждый из которых равен букве на соответствующей позиции экрана, либо ‘.’, если соответствующая позиция пустая и не содержит символ.

Пример

Стандартный ввод	**Стандартный вывод**
2 2 2 2 1 1 2 1 clip 1 2 2 2 overflow 4 print 1 1 abcd print 1 2 xxxx mode 1 wrap print 1 2 yyyy 4 3 3 1 1 1 1 overflow 1 2 1 3 clip 2 1 3 1 wrap 2 2 3 3 wrap 8 print 1 1 abcd mode 2 overflow print 1 2 efgh mode 3 overflow print 2 1 ijkl print 2 3 x mode 4 overflow print 2 3 yzt	ay y. aef ijy zt.

Замечание

Для удобства восприятия наборы входных данных и соответствующий им вывод в примерах в условии отделяются друг от друга пустыми строками. В реальных тестах этих **пустых строк нет**!

Решение:

Это задача на реализацию. Требовалось внимательно прочитать условие и реализовать то, что
в нем просили, добавив минимальные необходимые оптимизации.

Заведем

- text[r][c] - символ, стоящий на позиции (r, c);
- id[r][c] - номер окна, покрывающего позицию с координатами (r, c);
- mode[i] - режим, в котором работает окно номер i;
- left[i] и bottom[i] - номер левого столбца и нижней строки i-го окна, соответственно.

Изначально необходимо просто заполнить таблицу id в соответствии с данными в условии, после чего приступить к обработке запросов. Для каждого запроса будем поддерживать текущее положе- ние курсора (r, c) и очередной символ строки, который надо вывести. Обозначим также за id_{cur} номер текущего окна, то есть id[r][c].

Тогда сначала присвоим в text[r][c] текущий символ, а затем переместим курсор.

- Если $c < w$ и id[r][$c + 1$] = id_{cur}, то есть мы еще не дошли до правой границы окна, просто увеличим c на 1. Иначе, посмотрим на mode[id_{cur}] - режим текущего окна.
- Если режим равен «clip», сделаем break из цикла, отвечающего за вывод, чтобы его остано- вить.
- Если режим равен «wrap», перенесем курсор в $r \leftarrow r + 1$ и $c \leftarrow$ left[id_{cur}]. Следует отдельно проверить, что если r = bottom[id_{cur}], то тоже надо сделать break.
- Если же окно работает в режиме «overflow», то либо курсор переместится в (r, $c + 1$), либо в
- ($r + 1$, 1), либо просто следует сделать break, если достигнут правый нижний угол экрана.

Каждый запрос в таком случае обрабатывается за $O(|s_i|)$, и в сумме все решение работает за
$O(wh + |s_i|)$. Если же каждый раз искать номер текущего окна перебором, не храня id[r][c],
решение с большой вероятностью не будет проходить по времени.

В другом варианте этой задачи режимам «clip» и «overflow» соответствовали «fill» и «break», а «clip» был заменен на «repeat». В режиме «repeat» достаточно было просто перемещать курсор на (r, left[id_{cur}]).

9. Технологии программирования (3 балла)

[Делители факториала]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	4 секунды
Ограничение по памяти	256 мегабайт

Математика порой бывает довольно сложной. Однако, когда речь заходит о сложной математике, мало кто может сказать, что тема «факториалы» однозначно попадает в эту категорию. С задачей посчитать факториал целого числа, скорее всего, справится любой участник олимпиады.

Поэтому вам предстоит ответить на немного более сложные запросы. Всего запросов q и i-й запрос задается одним целым числом n_i. Для каждого запроса вам необходимо посчитать $\sigma_0(n_i!)$, то есть количество положительных делителей числа $n_i!$.

Поскольку реальное количество делителей числа $n!$ может быть слишком большим уже даже при $n \approx 100$, посчитайте ответ на каждый запрос по модулю $10^9 + 7$.

Формат входных данных

В первой строке ввода дано целое число q — количество запросов ($1 \le q \le 300$).

Во второй строке перечислены q целых чисел n_i — параметры запросов ($1 \le n_i \le 10^5$).

Формат выходных данных

Для каждого запроса посчитайте $\sigma_0(n_i!) \bmod (10^9 + 7)$ и выведите в ответ **остаток их суммы по модулю** 998 244 353.

Пример

Стандартный ввод	Стандартный вывод
10 1 2 3 4 5 6 7 10 100 1000	558634260

Замечание

В примере из условия независимые ответы для каждого n_i получаются такие:

$\sigma_0(1!) = 1$;

$\sigma_0(2!) = 2$;

$\sigma_0(3!) = 4$;

$\sigma_0(4!) = 8$;

$\sigma_0(5!) = 16$;

$\sigma_0(6!) = 30$;

$\sigma_0(7!) = 60$;

$\sigma_0(10!) = 270$;

$\sigma_0(100!) = 39\,001\,250\,856\,960\,000$, и по модулю $10^9 + 7$ это равно 583951250;

$\sigma_0(1000!) \bmod (10^9 + 7) = 972926972$, само значение мы приводить не будем, так как оно слишком большое.

Решение

Можно запустить линейное решето Эратосфена, которое позволяет для каждого числа x найти $\mathrm{lp}[x]$ – минимальное простое число в разложении x на простые.

Затем, вместо того, чтобы независимо отвечать на все запросы, просто предподсчитаем ответ для всех возможных n, и будем для ответа на запрос возвращать предподсчитанное значение. Для этого будем поддерживать $\deg[i]$ – степень вхождения i-го простого числа в $n!$.

Для начала запомним, что $\sigma_0(1!) = 1$, и затем будем перебирать n от 2 до 10^5, каждый раз пересчитывая значение ответа. Поскольку $n! = (n-1)! \cdot n$, то изменятся только степени вхождения простых, на которые делится n. Используя lp, можно за $O(\log n)$ разложить n на простые и для каждого p^{a_i} в его разложении поделить ответ на предыдущее значение $(\deg[i]+1)$, увеличить $\deg[i]$ на a_i и умножить ответ обратно на $(\deg[i]+1)$.

Модули $10^9 + 7$ и $10^9 + 9$ - простые, поэтому деление по ним можно осуществлять с помощью умножения на обратное, а для этого тоже можно воспользоваться двоичным возведением в степень, потому что по малой теореме Ферма $x^{M-2} = x^{-1}$.

Такое решение в сумме работает $O(A \log A)$ времени.