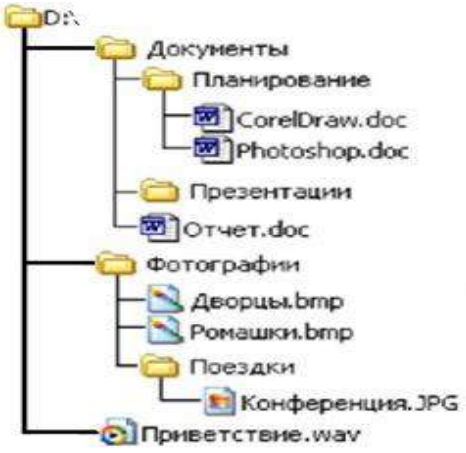
Задания заключительного этапа
Открытой олимпиада школьников «Северо-Кавказского федерального университета» среди учащихся образовательных организаций «45 параллель» по информатике за 2022-2023 учебный год

5-6 классы

Задача 1 (10 баллов)

На рисунке представлен граф-дерево, отображающий иерархию файловой системы диска D:.



На диске D: в папке Поездки пользователь создал подкаталог Сириус и сохранил в нём файл с именем Награждение и расширением jpg.

Запишите полное имя сохраненного файла

Для получения максимального балла за правильно решенную задачу, решение всех задач необходимо сопровождать подробным комментарием!!!

Задача 2 (15 баллов)

В базе данных компьютерного салона были сформированы запросы о проданных за неделю клавиатурах, мышах и джойстиках фирм Defender и Logitech.

№ запроса	Запрос	Количество записей
1	Продано клавиатур, мышей и джойстиков фирмы Defender	43
2	Продано клавиатур Logitech и мышей Logitech,	35
3	Продано всего клавиатур и мышей	72
4	Продано джойстиков фирмы Defender	?

Сколько записей будет соответствовать четвёртому запросу?

Задача 3 (20 баллов)

Крокодил Гена подарил Чебурашке на день рождения телефон. Чебурашке необходимо задать пароль для доступа к телефону. Пароль представляет собой слово из русских букв.

Для критериев построения пароля не верно ни одно из утверждений:

– длина слова меньше 6, но больше 8 или равна 8;

– слово начинается с согласной буквы;

– в нем нет ни одной буквы И;

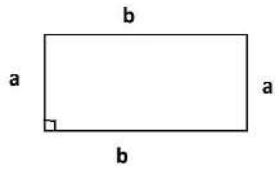
– в слове две гласные буквы могут стоять рядом;

– слово содержит только одну из букв И, Б, В.

Предложите Чебурашке свой вариант пароля. Решением задачи может быть бессмысленная совокупность символов (10 баллов) или слово русского языка (20 баллов).

Задача 4 (25 баллов)

Работники фермерского хозяйства запланировали прополку клубничных полей. Чертеж клубничного поля представлен на рисунке.



Согласно плану, на прополку поля выделяется X дней. Ежедневно работники пропалывают квадрат поля со стороной Y. Квадраты могут накладываться друг на друга. Определите Ymin при котором можно прополоть клубничное поле, потратив на это X дней?

<u>Пример.</u> Пусть клубничное поле имеет размеры a=b=4, время уборки поля – 4 дня. Разделим поле на квадраты. Из рисунка 2 видно, чтобы прополоть поле за 4 дня, Ymin должно быть равным 2.

1		2	
3		4	

План прополки клубничных полей

	A, км	B, км	X, дн	Ymin, км^2
Поле 1	6	6	4	?
Поле 2	7	16	7	?
Поле 3	14	12	7	?

Определите Ymin для первого, второго и третьего клубничных полей.

Запишите словесный алгоритм нахождения Ymin.

<u>Задача 5 (30 баллов)</u>

Автономный Добробот получает на вход последовательность символов X, к которой применяет следующий цикл преобразования:

1. Все вхождения символа «*» Дороброботзаменяет на “##”;
2. Все символы «!» заменяет на «*»;
3. Символы «#» заменяет на «!».

Например, X= «*!*!*». После одного цикла преобразования последовательность X = «!!* !! *!!».

Определите количество символов «*» и «#» в преобразованной последовательности X (10 балов) при условии, что:

– исходная последовательность X=*!*!**!

– цикл преобразования был применен к X 12 раз.

Постройте блок-схему алгоритма преобразования (10 баллов) и напишите программу (10 баллов). *<u>Программу можно написать на любом известном Вам языке программирования или на школьном алгоритмическом языке. Обязательно укажите название используемого языка программирования!!!</u>*

Задача 1.	**Ответ:** D:\Фотографии\Поездки\Сириус\Награждение.jpg **Критерии оценивания:** 10 баллов – правильный ответ. 5 баллов – ответ с недочетами. Например, в ответе вместо \ стоят / или название папки Сириус написана с маленькой буквы. 0 баллов – неверный ответ/задача не решена.
Задача 2.	**Ответ:** 6 **Критерии оценивания:** 15 баллов – правильный ответ. 5 баллов – ответ с недочетами. Например, в ответе вместо \ стоят / или название папки Сириус написана с маленькой буквы. 0 баллов – неверный ответ/задача не решена. **Решение:**

<table>
<tr><td></td><td>все клавиатуры + все мыши = 72 записи
клавиатуры Logitech + мыши Logitech = 35 записей
клавиатуры Defender + мыши Defender = 72 – 35 = 37 записей
джойстиков Defender = 43 - клавиатуры SVEN - мыши Defender = 43-37 = 6 записей</td></tr>
<tr><td>Задача 3.</td><td>Ответ: слово, соответствующее критериям.
Критерии оценивания:
20 баллов – слово из русского языка, соответствующее критериям.
10 баллов – бессмысленное слово, соответствующее критериям.
5 баллов – в пояснении к задаче правильно прописанные критерии, но неправильно сформированное слово.
0 баллов – неправильный ответ/задача не решена.
Решение:
Сформулируем критерии, которым должен соответствовать пароль (с учетом инверсии):
1. 7 <= длина строки <8.
2. Две гласные буквы не должны стоять рядом.
3. Пароль начинается не с согласной буквы.
4. В пароле присутствует хотя бы одна буква И.
5. Обязательно наличие хотя бы двух букв из букв И, Б, В.
Пример ответа 1: ОПРИНБИ (20 баллов).
Пример ответа 2: АБРИКОС (10 баллов).</td></tr>
<tr><td>Задача 4.</td><td>Ответ:
<table>
<tr><td></td><td>A, км</td><td>B, км</td><td>X, дней</td><td>Ymin, км</td></tr>
<tr><td>Поле 1</td><td>6</td><td>6</td><td>4</td><td><u>3</u></td></tr>
<tr><td>Поле 2</td><td>7</td><td>16</td><td>7</td><td><u>4</u></td></tr>
<tr><td>Поле 3</td><td>14</td><td>12</td><td>7</td><td><u>6</u></td></tr>
</table>
Словесный алгоритм имеет множество вариантов
Критерии оценивания:

25 баллов – правильный ответ (для всех полей) и правильно составленный алгоритм.
15 баллов – правильный ответ (для всех полей).
10 баллов – правильно составленный алгоритм.
5 баллов – правильный ответ (для полей 1 и 2).
0 баллов – неправильный ответ/задача не решена.
Решение:
1. Вычисляем площадь поля A*B
2. Вычисляем площадь квадрата для ежедневной уборки A*B/X
3. Вычисляем сторону квадрата Y = $+\sqrt{A * B/X}$
4. Если Y целое Ymin = Y, иначе Ymin = округлённому в сторону большего Y (т. к. по условию допускается наложение фигур).
<table>
<tr><td></td><td>A, км</td><td>B, км</td><td>X, дней</td><td>Y, км</td><td>Ymin, км</td></tr>
<tr><td>Поле 1</td><td>6</td><td>6</td><td>4</td><td>$+\sqrt{6 * 6/4}$</td><td><u>3</u></td></tr>
<tr><td>Поле 2</td><td>7</td><td>16</td><td>7</td><td>$+\sqrt{7 * 16/7}$</td><td><u>4</u></td></tr>
</table>
</td></tr>
</table>

	Поле 3	14	12	7	$+\sqrt{14 * 12/7}$ округляем до 6	**<u>6</u>**

Задача 5.

Ответ: $K_* = 256$, $K_\# = 0$

Блок-схема и программа имеет множество вариантов

Критерии оценивания:

30 баллов – правильный ответ, правильно составленная блок-схема и написанный текст программы.

10 баллов – правильный ответ.

10 баллов – правильно составленная блок-схема.

10 баллов – правильно написанный текст программы.

0 баллов – неправильный ответ/задача не решена.

Решение:

При решении задачи основной акцент делается не на синтаксическом содержании последовательности X, а на количестве символов в последовательности.

После одного цикла преобразования последовательности X. Количество символов «*» $K_* = K_!$, а $K_! = 2* K_*$, $K_\# = 0$. Делаем вывод, после цикла преобразования $K_\#$ всегда будет 0 и далее его вычислять не следует.

При исходной последовательность X=*!*!**!, применив цикл преобразования к X 12 раз, получаем:

	K_*	$K_!$	$K_\#$
Исходные	4	3	0
1	3	8	0
2	8	6	0
3	6	16	0
4	16	12	0
5	12	32	0
6	32	24	0
7	24	64	0
8	64	48	0
9	48	128	0
10	128	96	0
11	96	256	0
12	256	192	0

Таким образом, задав начальное количество символов и организовав цикл повторяющийся 12 раз. Получим результат.

В теле цикла количество восклицательных знаков, полученное на предыдущем шаге присваивается величине K_*, количество звездочек предыдущего шага удваивается и присваивается $K_!$

7-8 классы

<u>Задача 1 (10 баллов)</u>

На рисунке представлен граф-дерево, отображающий иерархию файловой системы диска C:.

В папке OFFICE хранятся файлы.

Имена этих файлов: молоток.mp3, лотерея1.doc, потолок.m3d, лото.exe, пилотаж.mp4, теплота.doc, флот.jpg, золото.rtf, шалот1.pdf, шалот_1.pdf, лотос.jpeg, плотина.mp3, шалот2.jpg.

Запишите в алфавитном порядке имена оставшихся в папке OFFICE файлов после проведения следующих операции над файлами этой папки:

1. Удалили группу файлов, соответствующих маске *лот*.?3* и вторую группу по маске ?лот*.*
2. Осуществили перенос нескольких файлов в каталог TEXT, согласно фильтру *ло??.*
3. Скопировали в папку PICTURE файлы, соответствующие маске *.j???

Запишите в алфавитном порядке имена оставшихся в папке OFFICE файлов.

Для получения максимального балла за правильно решенную задачу, решение всех задач необходимо сопровождать подробным комментарием!!!

Задача 2 (15 баллов)

Логическая функция $Y = \neg C\&A \vee A\&B$.

Таблица истинности функции *Y*

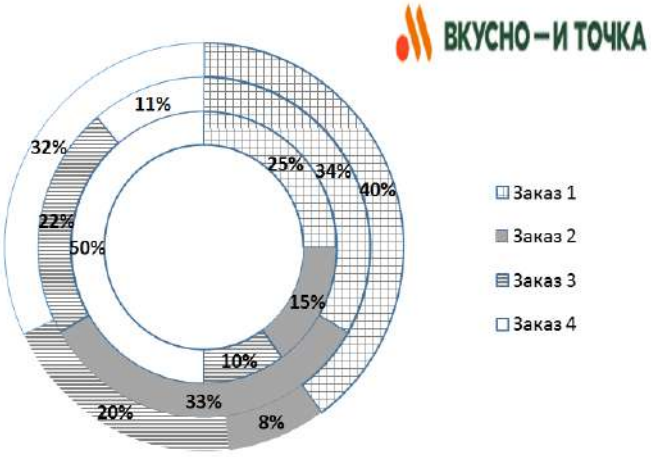
Переменная 1	0	0	0	0	1	1	1	1
Переменная 2	0	0	1	1	0	0	1	1
Переменная 3	0	1	0	1	0	1	0	1
Функция *Y*	0	1	0	1	0	0	0	1

Определите, каким строкам таблицы истинности функции *Y* соответствуют переменные *A, B, C*.

Задача 3 (20 баллов)

Сотрудник службы доставки подготовил отчет о выполненных заказах за смену в виде таблицы с точными значениями и диаграммы распределения товаров по заказам. Данные об одном из заказов в таблице скрыты. Восстановите скрытые данные.

ВКУСНО – И ТОЧКА	Молочный коктейль Тропик, шт	Картофельные Дольки средние, шт	Двойной Биг Хит, шт
Заказ 1	10	12	20
Заказ 2	6	12	4
Заказ 3	?	?	?
Заказ 4	20	4	16



Задача 4 (25 баллов)

По городу было установлено *X* терминалов. Для удобства учета терминалов возникла необходимость их нумерации. Для этого было закуплено *Z* литер-цифр.

Постройте блок-схему (10 баллов) и напишите программу (15 баллов), которая по заданному *Z*, определяет сколько терминалов установлено по городу или выдает сообщение, что закуплено неверное количество литер-цифр.

Программу можно написать на любом известном Вам языке программирования или на школьном алгоритмическом языке. Обязательно укажите название используемого языка программирования!!!

Формат ввода	натуральное число *Z*
Формат вывода	– число *X (1≤X≤900)*, если количество литер в числах от 1 до *X* соответствует *Z*; – ERROR, если закупщики ошиблись
Пример работы программы	
Пример ввода	*Пример вывода*
19	14
22	ERROR

Задача 5 (30 баллов)

В Кванториуме поставили автомат «Сладкоежка», который по веденому номеру выдает соответствующее номеру кондитерское изделие.

В ассортименте автомата 27 кондитерских изделий.

Кнопок у автомата – 6. Одна из кнопок – подтверждение заказа, пять других обозначены латинскими буквами – *A, B, C, D, E*.

Учащимся группы «Программеры» Кванториума предложили написать программу, *которая определяет, какие из кнопок A, B, C, D, E необходимо нажать, чтобы получить желаемое* кондитерское изделие.

Постройте блок-схему (10 баллов) и напишите программу (20 баллов), *которая определяет, какие из кнопок A, B, C, D, E необходимо нажать, чтобы получить желаемое* кондитерское изделие.

Формат ввода	натуральное число от 1 до 27
Формат вывода	обозначения копок (*A, B, C, D, E*), которые необходимо

	нажать, для того чтобы *получить желаемое* кондитерское изделие

Пример работы программы	
Пример ввода	*Пример вывода*
4	C
32	Нет такой сладости
11	*B D E*

Задача 1.

Ответ:

шалот_1.pdf
пилотаж.mp4
молоток.mp3
лотос.jpeg
лотерея1.doc

Критерии оценивания:

10 баллов – правильный ответ.
5 баллов – ответ с незначительными недочетами.
0 баллов – неверный ответ/задача не решена.

Решение:

Исходный список	Удалили файлы, соответствующие маске: *лот*.?3*	Удалили файлы, соответствующие маске: ?лот*.*	перенос нескольких файлов в каталог TEXT, согласно фильтру:*ло??.*
1.молоток.mp3 2. лотерея1.doc 3. потолок.m3d 4. лото.exe 5. пилотаж.mp4 6. теплота.doc 7. флот. jpg 8. золото.rtf 9. шалот1.pdf 10. шалот_1.pdf 11. лотос.jpeg 12. плотина.mp3 13. шалот2.jpg	1.молоток.mp3 2. лотерея1.doc 3. потолок.m3d 4. лото.exe 5. пилотаж.mp4 6. теплота.doc 7. флот. jpg 8. золото.rtf 9. шалот1.pdf 10. шалот_1.pdf 11. лотос.jpeg 12. плотина.mp3 13. шалот2.jpg	1.молоток.mp3 2. лотерея1.doc 3. потолок.m3d 4. лото.exe 5. пилотаж.mp4 6. теплота.doc 8. золото.rtf 9. шалот1.pdf 10. шалот_1.pdf 11. лотос.jpeg 13. шалот2.jpg	1.молоток.mp3 2. лотерея1.doc 3. потолок.m3d 5. пилотаж.mp4 10. шалот_1.pdf 11. лотос.jpeg
В алфавитном порядке Я-А	шалот_1.pdf потолок.m3d пилотаж.mp4 молоток.mp3 лотос.jpeg		

		лотерея1.doc			

Задача 2.	**Ответ:** Переменная 1 – *C* Переменная 2 – *B* Переменная 3 – *A* **Критерии оценивания:** 15 баллов – правильный ответ. 5 баллов – ответ с одной правильно определенной переменной (обоснование в решении). 0 баллов – неверный ответ/задача не решена. **Решение**: Данное выражение является дизъюнкцией двух конъюнкций. Можем заметить, что в обоих слагаемых есть множитель *A*. Т. е. при $A = 0$ сумма будет равна 0. Так, для переменной А подходит только третий столбец. Шестое значение функции равно 0 при $A = 1$. Такое возможно только при $C = 1$, $B = 0$, т. е. переменная 1 − *C*, а переменная 2 − *B*.
Задача 3.	**Ответ:** 4,8,10 **Критерии оценивания:** 20 баллов – правильный ответ. 5 баллов – ответ с одной правильно определенной переменной (обоснование в решении). 0 баллов – неверный ответ/задача не решена. **Решение**: При сопоставлении таблицы и диаграммы, следует учесть, что кольцо диаграммы это количество одного «блюда» в каждом из четырех заказов – 100%.
Задача 4.	**Ответ:** Существует множество вариантов решения задачи **Критерии оценивания:** 25 баллов – правильно составленная блок-схема и написанный текст программы. 15 баллов – правильно составленная блок-схема. 10 баллов – правильно написанный текст программ. 0 баллов – неправильный ответ/задача не решена. **Решение:** Учитывается, что у однозначных номеров используется одна из литер от 1 до 9. При построении двузначного номера по две литеры вкл. ноль и т.п. В алгоритме целесообразно использовать целочисленное деление и остаток от него) Среди чисел *X* из диапазона $1 \le X \le 900$ – 9 однозначных чисел, 100 двухзначных чисел и 801 трехзначное число. Следовательно, максимальное количество литер *Z* для 900 терминалов $Zmax=9+2*90+3*801= 2592$. Если при вводе *Z,* его значение изначально не соответствует условию $1 \le Z \le 2592$. Выдается *ERROR* и выполнение программы прекращается. Иначе:

<table>
<tr><td></td><td>1. Проверяется условие $Z \leq 9$. Если оно истинно выводится значение X равное Z.
2. Если $9 < Z \leq 189$, проверяется условие $((Z-9) \bmod 2)=0$. Если остаток ноль выводится значение X равное сумме $(Z-9) \operatorname{div} 2$ и 9, иначе выдается ERROR.
3. Если $189 < Z \leq 2592$, проверяется условие $((Z-189) \bmod 3)=0$. Если остаток ноль выводится значение X равное сумме $(Z-209) \operatorname{div} 3$ и 99, иначе выдается ERROR.</td></tr>
<tr><td>Задача 5.</td><td>Ответ:
Существует множество вариантов решения задачи
Критерии оценивания:
30 баллов – правильно составленная блок-схема и написанный текст программы.
15 баллов – правильно составленная блок-схема.
15 баллов – правильно написанный текст программ.
0 баллов – неправильный ответ/задача не решена.
Решение:
Задача сводится к двоичному кодированию чисел. Максимальный номер по условию задачи – 27. Для его кодирования достаточно пяти двоичных разрядов. 5 кнопок – 5 двоичных разрядов. Из примера выполнения программы видно, что старший разряд – кнопка A, младший E. Нажатая кнопка – 1.
В начале выполнения программы проверяется условие $1 \leq N \leq 27$, где N – номер выбранного кондитерского изделия. Если N не попадает в указанный диапазон выдается сообщение об отсутствии сладости с таким номером. Иначе число N переводится в двоичную систему. Т. е. двоичным единицам ставится в соответствии одна из пяти кнопок
Например, $N=11$ – число попадает в указанный диапазон. При переводе в двоичную систему получаем число 1011. Определяем соответствие кнопок
<table>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
</table>
Результат B D E</td></tr>
</table>

Задания для 9-11 классов

Задача 1. Операционная система

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вася решил участвовать в конкурсе юных программистов. Для участия в конкурсе необходимо написать собственную операционную систему. Для начала Вася запланировал написать подпрограмму, которая будет рисовать рамки окон в интерфейсе его ОС.

Поле для рисования представляет собой прямоугольник $h \times w$ пикселей, строки

занумерованы сверху вниз от 1 до h, столбцы — слева направо от 1 до w.

На поле последовательно рисуются n рамок, i-я рамка представляет собой границы прямоугольника с противоположными углами в точках $(r_{i,1}, c_{i,1})$ и $(r_{i,2}, c_{i,2})$.

Требуется вывести получившееся изображение в виде h рядов по w символов, пискель, который не был использован при изображении рамок, следует вывести с использованием символа «.», а пиксели i-й рамки с использованием i-го символа латинского алфавита (первая рамка изображается буквами «a», вторая — «b», и т.д.)

Формат ввода

Первая строка содержит целые числа h, w и n — размеры поля и число рамок ($2 \le h, w \le 80, 1 \le n \le 26$). Следующие n строк содержат по четыре целых числа каждая: $r_{i,1}$, $c_{i,1}$, $r_{i,2}$ и $c_{i,2}$ ($1 \le r_{i,1} < r_{i,2} \le h$,. $1 \le c_{i,1} < c_{i,2} \le w$).

Формат вывода

Выведите результат вывода описанных во вводе рамок.

Пример

Ввод	**Вывод**
10 10 2 1 1 7 8 3 3 4 9	aaaaaaaa.. a......a.. a.bbbbbbb. a.bbbbbbb. a......a.. a......a.. aaaaaaaa.. ....

Решение:

```
#define TASKNAME "text"

#include <bits/stdc++.h>

#define all(a) (a).begin(), (a).end()
#define zero(a) memset(a, 0, sizeof(a))
#define sz(a) (int)a.size()
#define fst first
#define snd second
#define y1 osrughosduvgarligybakrybrogvba
#define y0 aosfigdalrowgyalsouvgrlvygalri
#define mp make_pair
#define pb push_back
#define eprintf(...) fprintf(stderr, __VA_ARGS__)

using namespace std;
```

```
typedef long long ll;
typedef long double ld;
typedef vector <int> vi;
typedef vector <vi> vvi;
typedef vector <bool> vb;
typedef vector <ll> vll;
typedef pair <int, int> pii;
typedef pair <ll, ll> pll;
typedef pair <ll, int> pli;
typedef pair <int, ll> pil;
typedef vector <pii> vpii;

#ifdef WIN32
   #define LLD "%I64d"
#else
   #define LLD "%lld"
#endif

template <typename T>
T sqr(T x) {
   return x * x;
}

template <typename T>
T abs(T x) {
   return x > 0 ? x : -x;
}

const double EPS = 1e-9;
const int INF = 1e9;
const ll INFLONG = (ll)1e18;

const int maxn = 1010;

char ans[maxn][maxn];

int main()
{
  int n, m, k;
  scanf("%d%d%d", &n, &m, &k);
  for (int i = 0; i < n; i++)
  for (int j = 0; j < m; j++)
     ans[i][j] = '.';
  for (int it = 0; it < k; it++) {
     int x1, y1, x2, y2;
     scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
```

```
    --x1, --y1, --x2, --y2;
    int x = x1;
    for (int i = 0; i < 2; i++) {
        for (int j = y1; j <= y2; j++) {
            ans[x][j] = 'a' + it;
        }
        x = x2;
    }
    int y = y1;
    for (int i = 0; i < 2; i++) {
        for (int j = x1; j <= x2; j++) {
            ans[j][y] = 'a' + it;
        }
        y = y2;
    }
  }
  for (int i = 0; i < n; i++, printf("\n"))
  for (int j = 0; j < m; j++)
    printf("%c", ans[i][j]);
  return 0;
}
```

Задача 2. Пирамида

Ограничение времени	3 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вася решил обучать свою младшую сестру Машу и развивать у нее алгоритмическое мышление. Он дал ей коробку пластиковых чашек и предложил составить из них пирамиду.

Он объяснил Маше, как из чашек можно строить пирамиды. Каждая чашка в пирамиде, не считая основания, ставится на края четырех других. Таким образом, если в основании пирамиды находится прямоугольник, состоящий из $N \times M$ чашек, то следующий уровень будет состоять из $(N-1) \times (M-1)$ чашек, следующий из $(N-2) \times (M-2)$, и так далее...

Уровни продолжаются до тех пор, пока можно поставить чашку на края четырех других чашек из предыдущего уровня.

Маша хочет узнать, сколько уйдет чашек на строительство пирамиды с определенным размером основания. Маше сложно посчитать такие числа, тем более, что они могут быть большими, поэтому она задала этот вопрос старшему брату. Помогите Васе ответить на все вопросы любопытной сестры.

Формат ввода

В первой строке задано одно натуральное число t — число оснований пирамид, которые Маша планирует строить ($1 \le t \le 10^5$).

В следующих t строках заданы пары чисел N_i, M_i — размеры оснований ($1 \le N_i, M_i \le 10^9$).

Формат вывода

Для каждого основания в отдельной строке выведите число чашек, которые потребуются, чтобы построить пирамиду с соответствующим основанием.

Пример

Ввод	**Вывод**
3 1 1 2 5 3 3	1 14 14

Решение:

Пусть N ⩽ M. Если это не так поменяем значения местами, ответ не изменится. Тогда количество стаканчиков в пирамидке равно N ·M + (N −1)·(M −1)+· · ·+1·(M −N +1) = ∑N−1 i=0 (N −i)·(M −i).

Для частичного решения достаточно было посчитать эту сумму с помощью цикла. Сложность такого решения O(t · min(N, M)).

Во второй группе тестов N = M, тогда ∑N−1 i=0 (N −i)·(M −i) = ∑N−1 i=0 (N −i) 2 , что в свою очередь при замене переменной j = N − i будет равно ∑N j=1 j 2 = N·(N+1)·(2N+1) 6 (общеизвестная формула). Сложность O(t), так как теперь на каждый запрос отвечаем за O(1). ∑ Применим такую же замену в первоначальную формулу, получим N j=1 j · (M − N + j) = ∑N j=1(M − N)j + j 2 = (M − N) · ∑N j=1 j + ∑N j=1 j 2 . Таким образом мы получили две суммы, которые мы уже умеем считать по формулам. Получаем N·(N+1)·(3M−N+1) 6 . По полученной формуле видно, что ответ будет порядка N3 .

Соответственно, чтобы получить полный балл, нужно применить длинную арифметику.

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

#include<bits\stdc++.h>
using namespace std;

int main()
```

```
{
   int t;
   cin >> t;
   while (t--)
   {
      long long n, m;
      cin >> n >> m;
      if (n > m)
         swap(n, m);
      long long ans = 1ll * n * (n + 1) * (3 * m - n + 1) / 6;
      cout << ans << endl;
   }
   return 0;
```

Задача 3. Сказочные палиндромы

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Маша попросила Васю почитать вместе с ней сказку. Вместо чтения он предложил Маше поискать в тексте сказки палнидромы. Палиндром — строка A(1)A(2)…A(n), для которой выполняются равенства

A(1)=A(n), A(2)=A(n-1), …, A(n/2)=A(n+1-n/2),

где A(1), A(2), …, A(n) — некоторые символы.

В качестве задания Вася предложил сестре найти и подсчитать в тексте сказки количество слов, являющихся палиндромами. Маше сложно, ведь она недавно научилась читать. Помогите девочке справиться с заданием.

Словом считается последовательность, состоящая только из букв. Если в тексте несколько слов, то они обязательно отделяются друг от друга разделительными символами. Слово является палиндромом, если оно читается одинаково в любом направлении.

Формат ввода

Задана непустая строка длиной до 100000 символов, состоящая из строчных букв латинского алфавита, пробелов, точек и запятых.

Формат вывода

Выведите сколько слов в тексте является палиндромами.

Пример 1

Ввод	**Вывод**
aba	1

Пример 2

Ввод	**Вывод**
abc,aba,d	2

Решение:

Решение задачи подразумевает правильное понимание условия задачи и аккуратную программную реализацию. Важным для полного решения задачи является:

1. умение считать строку, содержащую пробелы;
2. понимание понятия “палиндром”;
3. умение выделить слово в тексте;
4. умение программно проверить, является ли последовательность символов палиндромом.

```
#include<cstdio>
#include<cctype>
using namespace std;
int main()
{
   freopen("input.txt","r",stdin);
   freopen("output.txt","w",stdout);
   char s[100002]={};
   gets(s);
   bool word = false;
   int word_start,ans=0;
   for (int i=0;i==0||s[i-1];++i)
   {
      if (isalpha(s[i]))
      {
         if (!word)
         {
            word = true;
            word_start = i;
         }
      }
      else if (word)
      {
         word = false;
         bool pal = true;
         for (int j=word_start;j<i;++j)
            pal = pal && s[j]==s[i+word_start-j-1];
         if (pal)
            ans++;
      }
   }
   printf("%d\n",ans);
```

```
    fclose(stdin);
    fclose(stdout);
    return 0;
}
```

Задача 4. Арбузная бахча

Ограничение времени	2 секунды
Ограничение памяти	256 Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вася решил помочь местному фермеру собрать урожай арбузов. Поле, на котором растут арбузы называется бахча. Бахча состоит из n строк и m столбцов квадратных клеток, в каждой из которых растет арбуз. Будем обозначать клетку на пересечении x-й строки и y-го столбца как (x, y).

Так как работа по сбору арбузов утомительна, Василий решил создать программу, отвечающую на запросы следующего вида. Вася называет клетку (x, y) и одно из четырех направлений (вверх, вниз, влево, вправо) и просит найти ближайшую к (x, y) клетку в выбранном направлении, в которой еще есть арбуз, либо сказать, что такой клетки нет.

Формат ввода

В первой строке даны три целых числа n, m и q — размеры бахчи и количество запросов ($1 \le n, m \le 2\,000$; $1 \le q \le 10^6$). В следующих строках даны запросы. Каждый запрос начинается с символа, означающего действие, затем идут два целых числа x_i и y_i ($1 \le x_i \le n$, $1 \le y_i \le m$).

Символ «c» означает, что Вася сорвал арбуз в данной клетке (x_i, y_i). Гарантируется, что арбуз может находиться в каждой клетке не более одного раза.

Иначе, необходимо определить ближайшую к клетке (x_i, y_i) в выбранном направлении клетку, в которой еще есть арбуз:

1) символ «u» означает направление "вверх",

2) символ «d» — направление "вниз",

3) символ «l» — направление "влево",

4) символ «r» — направление "вправо".

Формат вывода

На каждый запрос на поиск арбуза выведите искомую клетку, или «-1», если такой клетки нет.

Пример

Ввод	**Вывод**
3 4 6 u 2 3 c 2 4 r 2 4 c 2 3	1 3 -1 2 2 3 3

Ввод **Вывод**

l 2 4
d 1 3

Будем поддерживать для каждой клетки ближайшую клетку, в которой еще не сорван арбуз, в каждом из четырех направлений. Когда поступает запрос на нахождение клетки с арбузом, пройдемся по этим переходам в нужном направлении, пока не дойдем до клетки с арбузом или до границы поля. После этого, во всех клетках, по которым мы прошли, обновим ссылку, теперь они будут указывать на клетку, которая сейчас является ответом. Таким образом, получился аналог системы непересекающихся множеств с эвристикой переподвешиваний. Значит, амортизированное время работы "--- $O(\log n)$ на запрос.

В частичном решении можно было поддерживать клетки с арбузами, и при каждом запросе перебирать клетки в нужном направлении, пока не встретится клетка с арбузом. Асимптотика решения "--- $O(q \cdot (n + m))$.

Во второй подгруппе можно было обработать все запросы на выкапывание арбуз, а потом с помощью метода динамического программирования найти ближайшую клетку с арбузом для каждой клетки в каждом направлении. Рассмотрим, как найти ближайшую клетку с арбузом для каждой клетки в направлении вверх. Будем перебирать клетки по строкам от верхней до нижней. Для очередной клетки, если соседняя сверху клетка содержит арбуз, она является ближайшей, содержащей арбуз. Иначе, ответ для текущей клетки совпадает с ответом для соседней сверху клетки. Направления вниз, влево и вправо решаются аналогично. Асимптотика решения "--- $O(n \cdot m + q)$.

```
#ifdef LOCAL
#  define _GLIBCXX_DEBUG
#else
#  define cerr __get_ce
#endif
#include <bits/stdc++.h>
#define ff first
#define ss second
#define szof(x) ((int)x.size())

using namespace std;
typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
int const INF = (int)1e9 + 1e3;
ll const INFL = (ll)1e18 + 1e6;
mt19937 tw(9450189);
uniform_int_distribution<ll> ll_distr;
ll rnd(ll a, ll b) { return ll_distr(tw) % (b - a + 1) + a; }

int dx[4] = {1, 0, -1, 0};
int dy[4] = {0, 1, 0, -1};
```

```
void solve() {
  int n, m, q;
  cin >> n >> m >> q;
  vector<vector<vector<pii>>> go(4, vector<vector<pii>>(n, vector<pii>(m)));

  for (int i = 0; i < 4; ++i) {
        for (int j = 0; j < n; ++j) {
              for (int k = 0; k < m; ++k) {
                    go[i][j][k] = {j + dx[i], k + dy[i]};
              }
        }
  }

  vector<vector<int>> field(n, vector<int>(m));

  for (int i = 0; i < q; ++i) {
        char c;
        cin >> c;
        int x, y;
        cin >> x >> y;
        --x; --y;
        if (c == 'c') {
              field[x][y] = 1;
        } else {
              int dir = -1;
              if (c == 'd') {
                    dir = 0;
              } else if (c == 'r') {
                    dir = 1;
              } else if (c == 'u') {
                    dir = 2;
              } else if (c == 'l') {
                    dir = 3;
              } else {
                    assert(false);
              }

              x += dx[dir];
              y += dy[dir];
              int memx = x, memy = y;
              while (0 <= x && x < n && 0 <= y && y < m && field[x][y]) {
                    tie(x, y) = go[dir][x][y];
              }

              int resx = x, resy = y;
              x = memx, y = memy;
```

```
                while (0 <= x && x < n && 0 <= y && y < m && field[x][y]) {
                        auto tmp = go[dir][x][y];
                        go[dir][x][y] = {resx, resy};
                        tie(x, y) = tmp;
                }

                if (0 <= resx && resx < n && 0 <= resy && resy < m) {
                        cout << resx + 1 << " " << resy + 1 << "\n";
                } else {
                        cout << "-1\n";
                }
        }
    }
}

int main() {
#ifdef LOCAL
    auto start_time = clock();
    cerr << setprecision(3) << fixed;
#endif
    cout << setprecision(15) << fixed;
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int test_count = 1;
    // cin >> test_count;
    for (int test = 1; test <= test_count; ++test) {
        solve();
    }
#ifdef LOCAL
    auto end_time = clock();
    cerr << "Execution time: " << (end_time - start_time) * (int)1e3 / CLOCKS_PER_SEC << "
ms\n";
#endif
}
```

Задача 5. Сломанный ключ

Ограничение времени	2 секунды
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вася и Петя играют в криптографическую игру. Чтобы выиграть, Вася собрал сложный числовой ключ, состоящий из последовательности натуральных чисел. Чтобы ключ сработал, числа в последовательности должны быть отсортированы по возрастанию. Но случилось непредвиденное. К Васиному компьютеру подбежала Маша и, пока Васи не было в комнате, испортила последовательность чисел в ключе — она перестала быть отсортированной. Васе необходимо её снова отсортировать, но в игре для этого можно использовать только две операции:

- Взять любое число в последовательности и переместить его в **конец** числового ключа.
- Взять любое число и переместить его в **начало** ключа.

Таким образом, если в последовательности изначально были числа $a_1, a_2, \ldots a_{i-1}, a_i, a_{i+1} \ldots a_n$ и было выбрано *i*-ое число, то если применить первую операцию, ключ станет выглядеть как $a_1, a_2, \ldots a_{i-1}, a_{i+1} \ldots a_n, a_i$, а в случае применения второй операции — как $a_i, a_1, a_2, \ldots a_{i-1}, a_{i+1} \ldots a_n$.

Гарантируется, что с помощью этих двух операций всегда можно отсортировать последовательность чисел. Помогите Василию найти наименьшее количество таких операций, необходимых, чтобы исправить ключ.

Формат ввода

В первой строке содержится одно целое число n — длина числового ключа ($1 \le n \le 300\,000$).

Во второй строке заданы n целых чисел a_i, разделенных пробелами ($1 \le a_i \le 10^9$).

Формат вывода

Выведите единственное целое число — минимальное число операций, которые нужно применить к данной последовательности чисел, чтобы отсортировать её.

Пример 1

Ввод	**Вывод**
5 3 1 2 4 5	2

Пример 2

Ввод	**Вывод**
5 5 4 3 2 1	4

Пример 3

Ввод	**Вывод**
6 2 3 1 6 4 5	2

Примечания

В первом примере достаточно сначала переставить *2* в начало последовательности, а затем *1*.

Во втором примере можно оставить *1* на месте, а все остальные числа переставить в конец. Или оставить *5* на месте, а все остальные числа по очереди переставить в начало. В любом случае будет минимум четыре операции.

В третьем примере достаточно переставить *1* в начало, а *6* в конец последовательности.

Решение:

Очевидно, что в задаче идет речь о сортировке массива.

Заметим пару полезных фактов: если с помощью операций перенести k элементов в начало и m элементов в конец массива, то:

• Можно выполнять операции с такой очередностью, что перенесенные элементы будут в любом необходимом порядке, например отсортированном.

• Элементы, которые перенесены в начало являются k наименьшими числами исходного массива, а в свою очередь элементы, перенесенные в конец должны быть m наибольшими числами в массиве.

Таким образом, чтобы массив был отсортирован, все элементы, которые не переносились должны быть в отсортированном порядке относительно друг друга.

Для облегчения задачи выполним масштабирование: отсортируем первоначальные элементы в отдельном массиве, а затем уберем повторы. Затем в исходном массиве заменим элементы, на их индексы в полученном массиве. Например, если изначально массив выглядел, как {3, 1, 2, 3, 40, 55}, после масштабирования он будет выглядеть следующим образом: {2, 0, 1, 2, 3, 4}. Заметим, что так как масштабирование не влияет на порядок элементов, ответ на задачу для исходного массива и нового будет идентичным. Более того, теперь в массиве все значения не превосходят его длину.

После масштабирования выполняется следующее свойство, если в массиве есть элемент x, то в нем также есть элемент x + 1, либо x является наибольшим элементом массива. Тогда заметим, что после удаления из массива элементов, к которым была применена операция оставшиеся элементы будут образовывать последовательность следующего вида: {x, x, . . . , x, x+1, x+1 . . . x+1, . . . x+q}, то есть разобьется на отрезки из соседних чисел. Иначе массив не будет отсортирован.

Тогда заметим:

• Если к хотя бы одному элементу со значением x применяется операция по переносу в начало, то нигде среди оставшихся элементов кроме как в начале не могут находится элементы с таким же значением.

• Аналогичное верно для переносов в конец.

Для начала для каждого значения массива посчитаем его первое и последнее вхождение в исходный массив, составим из них отрезок [lx; rx]. Тогда оставшиеся элементы представляются, как последовательно соединенные отрезки, а также для каждого отрезка верно:

• Все значения за которые он отвечает были переставлены, тогда отрезок не входит в оставшиеся элементы.

• Все элементы из этого отрезка остались нетронутыми, тогда он входит в оставшиеся элементы.

• Некоторые (не все) элементы со значениями, за которые отвечал отрезок, могли быть перенесены вперед и тогда отрезок [lx, mx] будет являться началом последовательности оставшихся элементов, где mx некоторое вхождение x.

• Аналогично для переносов в конец.

Некоторые отрезки вполне могут «не мешать друг другу», например, если рассмотреть некоторое значение x и будет верно, что rx < lx+1, заменим эти два отрезка на отрезок [lx; rx+1]. После всех возможных замен получится, что последовательность оставшихся элементов выглядит следующим образом: Либо начало отрезка и конец следующего за ним, либо три подряд отрезка, у крайних из которых взяты только части. Все нужные значения mx необходимые для нахождения частей отрезков предлагается искать двоичным поиском. Итоговая сложность решения O(n log n).

```
#ifdef DEBUG
#  define _GLIBCXX_DEBUG
#endif
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#define ff first
#define ss second
#define szof(x) ((int)x.size())
#ifndef LOCAL
#  define cerr __get_ce
#endif
using namespace std;
typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef unsigned long long ull;

using namespace __gnu_pbds;
template <typename T> using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;
template <typename K, typename V> using ordered_map = tree<K, V, less<K>, rb_tree_tag, tree_order_statistics_node_update>;

int const INF = (int)1e9 + 1e3;
ll const INFL = (ll)1e18 + 1e6;
#ifdef LOCAL
  mt19937 tw(9450189);
#else
  mt19937 tw(chrono::high_resolution_clock::now().time_since_epoch().count());
#endif
uniform_int_distribution<ll> ll_distr;
ll rnd(ll a, ll b) { return ll_distr(tw) % (b - a + 1) + a; }


void solve() {
  int n;
  cin >> n;
  vector<int> arr;
  for (int i = 0; i < n; ++i) {
```

```
		int num;
		cin >> num;
		arr.push_back(num);
	}

	vector<int> vals = arr;
	sort(vals.begin(), vals.end());
	vals.erase(unique(vals.begin(), vals.end()), vals.end());

	for (int& num : arr) {
		num = lower_bound(vals.begin(), vals.end(), num) - vals.begin();
	}

	vector<vector<int>> where(szof(vals));
	for (int i = 0; i < n; ++i) {
		where[arr[i]].push_back(i);
	}

	int ans = 0;

	int cnt = 0;
	while (cnt < szof(vals)) {
		int to = cnt + 1;
		while (to < szof(vals) && where[to - 1].back() < where[to].front()) {
			++to;
		}

		int sum = 0;
		for (int i = cnt; i < to; ++i) {
			sum += szof(where[i]);
		}

		if (cnt > 0) {
			sum += lower_bound(where[cnt - 1].begin(), where[cnt - 1].end(), where[cnt].front()) - where[cnt - 1].begin();
		}

		if (to < szof(vals)) {
			sum += where[to].end() - lower_bound(where[to].begin(), where[to].end(), where[to - 1].back());
		}

		ans = max(ans, sum);

		cnt = to;
	}
```

```
  for (int i = 0; i < szof(vals) - 1; ++i) {
        for (int j = 0; j < szof(where[i]); ++j) {
              ans = max(ans, j + 1 + (int) (where[i + 1].end() - lower_bound(where[i + 1].begin(), where[i + 1].end(), where[i][j])));
        }
  }

  cout << n - ans << "\n";
}

int main() {
#ifdef LOCAL
  auto start_time = clock();
  cerr << setprecision(3) << fixed;
#endif
  cout << setprecision(15) << fixed;
  ios::sync_with_stdio(false);
  cin.tie(nullptr);

  int test_count = 1;
  // cin >> test_count;
  for (int test = 1; test <= test_count; ++test) {
        solve();
  }

#ifdef LOCAL
  auto end_time = clock();
  cerr << "Execution time: " << (end_time - start_time) * (int)1e3 / CLOCKS_PER_SEC << " ms\n";
#endif
}
```

Председатель организационного комитета
Открытой олимпиады СКФУ среди учащихся
образовательных организаций
«45 параллель» **_____Иванов В. А.**

Критерии оценивания письменных олимпиадных работ

1. Задание считаются решенными, если получен верный результат, выполнены необходимые действия и их обоснование, ведущие к этому результату. Максимальная сумма баллов за работу – 100.

2. Любое полностью правильное выполнение заданий оценивается в наибольшее количество баллов.

3. Правильный ответ, приведенный без обоснования или полученный из неправильных рассуждений, не учитывается.

4. Полный балл выставляется при правильном и полном выполнении задания.

5. Если задание не выполнено или при выполнении допущена принципиальная ошибка, тозадание оценивается в «0» баллов.

6. Если задание выполнено, но: – допущена грубая ошибка – снимается 50% от числа баллов, которыми оценено данное задание; – допущена негрубая ошибка – снимается 30% от числа баллов, которыми оценено данное задание; – допущены грамматические ошибки, небрежности – снимается 10% за каждую грамматическую ошибку или небрежность, но не более 15 баллов со всей работы.

7. К грубым ошибкам относятся ошибки, которые обнаруживают незнание учащимися формул, правил, основных свойств, теорем и неумение их применять; незнание приемов решения задач, а также вычислительные ошибки, если они не являются опиской.

8. К недочетам относятся: описки, недостаточность или отсутствие пояснений, обоснований в решениях.

9. Все ошибки, выявленные в ходе проверки олимпиадных работ, отмечаются красной пастой.

10. Количество победителей и призеров олимпиады не должно превышать 45 % от общего количества участников соответствующего этапа олимпиады.

Председатель организационного комитета
Открытой олимпиады СКФУ среди учащихся
образовательных организаций
«45 параллель» **_____Иванов В. А.**

Задания заключительного этапа
Открытой олимпиада школьников «Северо-Кавказского федерального университета» среди учащихся образовательных организаций «45 параллель» по информатике за 2023-2024 учебный год

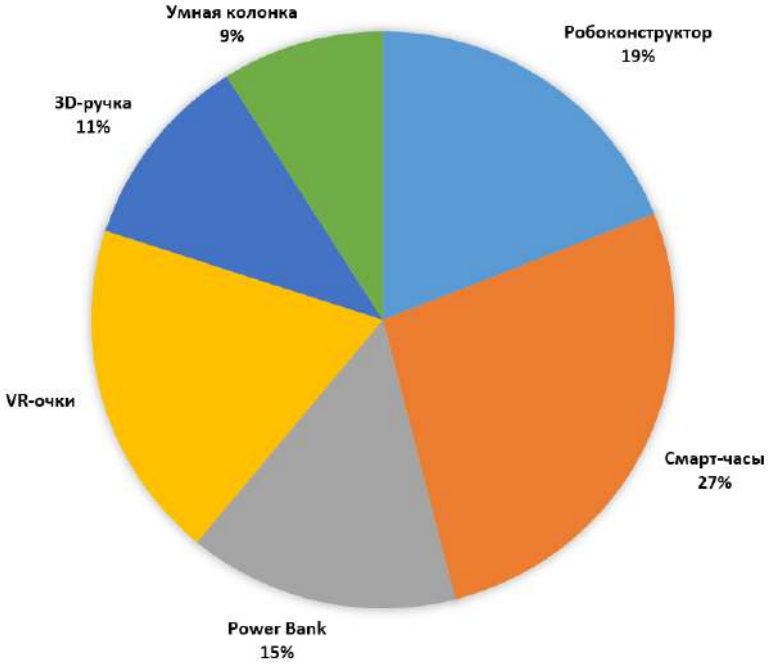
5-6 классы

Задача 1 (10 баллов)

400 школьников приняли участие в опросе «Гаджет-мечта». Каждый школьник выбрал из списка название гаджета, который он хочет получить в подарок от родителей. По результатам опроса построили диаграмму.

Определите, сколько школьников мечтают об VR-очках?

В ответе укажите только число (пробелы, запятые, точки и другие символы в ответе не использовать).



Задача 2 (15 баллов)

Победители Инфохакатона Денис Ф., Семён А. и Юрий С. посещают по две образовательные программы Кванториума. Образовательные программы Кванториума и их шифры представлены таблице

Образовательные программы			
Базовые		**Дополнительные**	
Шифр	Название	Шифр	Название
Б1	Аэроквантум	Д1	Хайтек
Б2	VR/AR-квантум		
Б3	Робоквантум	Д2	Квантоматематика
Б4	IT-квантум		

Информация о победителях:

1. На образовательных программах ребята не пересекаются.
2. Ни Денис, ни Юрий не посещают Робоквантум.
3. Учащийся программы Хайтек, учащийся IT-квантум и Семён все трое родом из Пятигорска.
4. Учащийся программы Квантоматематика, учащийся Аэроквантум и Юрий втроем учатся в одной школе.
5. Денис – самый юный из троих победителей и не посещает Квантоматематику.
6. Программы, которые посещает Юрий, относятся к базовым образовательным программам.

Определите, какие программы посещает Денис?

В ответе через пробел укажите только шифры программ (! запятые, точки и другие символы в ответе не использовать).

Пример ввода ответа: ***Б4 Д1***

Задача 3 (20 баллов)

При раскопках в пригороде Херсонеса археологи обнаружили 8 глиняных кувшинов с монетами. После подсчёта монет в каждом сосуде ученые заметили, что число монет в кувшинах подчиняется некоторой закономерности. В первых четырех кувшинах *оказалось соответственно 840, 420, 280 и 210 монет* древних обитателей Северного Причерноморья.

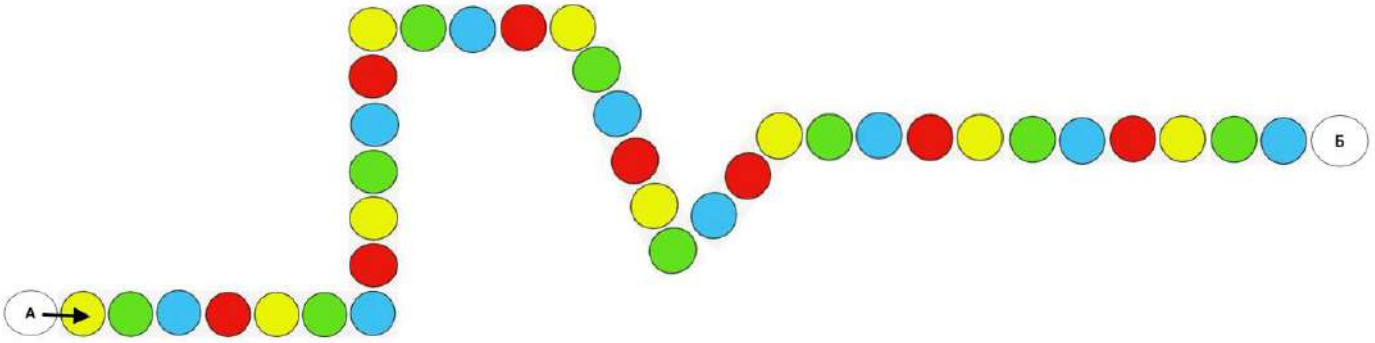
Укажите сколько монет в седьмом и восьмом кувшинах соответственно?

При вводе ответа числа разделите пробелом (запятые, точки и другие символы в ответе не использовать).

Пример ввода ответа: ***145 23***

Задача 4 (25 баллов)

Исполнитель Улитка должна добраться из пункта А в пункт Б пройдя через все ячейки цепи.



Известно, что исполнитель Улитка соблюдает правила:

– за минуту перемещаться на целое число ячеек;

– из ячейки А начинает двигаться со скоростью 1 ячейка в минуту;

– в ячейку Б прибывает со скоростью 1 ячейка в минуту;

– каждую минуту может не изменять скорость;

– каждую минуту может увеличивать/уменьшать скорость на 1;

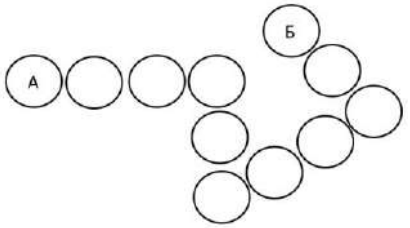
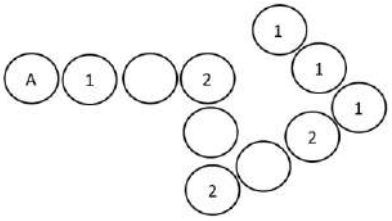
– обязательно завершает текущее перемещение (заканчивается минута) перед изменением направления движения.

Сформируйте алгоритм передвижения Улитки из пункта А в пункт Б за минимальное время. В ответе укажите минимальную по количеству чисел последовательность, соответствующую количеству пройденных клеток за каждую минуту (соблюдая описанные правила).

При вводе ответа отделяйте числа последовательности пробелами (! запятые, точки и другие символы в ответе не использовать).

Например, если бы маршрут имел следующий вид:

Маршрут	Расчет	*Ввод ответа:*

		***1 2 2 2 1 1 1 1***

Задача 5 (30 баллов)

Исполнитель Роботизированная рука (РР) заклеивает цветной пленкой ячейки мозаичного витража.

Система команд РР:

– Лх (переместиться влево на х ячеек);

– Пх (переместиться вправо на х ячеек);

– Вх (переместиться вверх на х ячеек);

– Нх (переместиться вниз на х ячеек);

– Кс (заклеить текущую ячейку мозаики пленкой цвета номер С);

– *повторить N раз* (повторить N раз команды, указанные до ключевого слова *кц* (окончание тела цикла)).

Роботизированная рука выполнила программу:

повторить 3 раз
  вниз 2;
  клеить 5;
  повторить 3 раз
    вверх 3;
    клеить 5;
  кц;
  клеить 5;
кц;

1. Сколько ячеек РР заклеил пленкой цвета 5, исполнив программу?
2. Какую команду программы можно удалить, не изменив рисунка витража? В ответе укажите номер строки, в которой записана команда (например, лишняя команда *вверх 3*, это 5-ая по счёту строка программы*)*

В ответе через пробел укажите два числа (! запятые, точки и другие символы в ответе не использовать). Первое число – ответ на первый вопрос. Второе – ответ на второй вопрос.

*Пример ввода ответа:**7 5***

Задача 1.	**Ответ:** 76 **Критерии оценивания:** 10 баллов – правильный ответ. 0 баллов – неверный ответ/неверно введенный ответ /задача не решена. **Решение:** VR-очки, % = 100 - (19+27+15+11+9) = 19 400 школьников – 100 % Х школьников – 19 % Х = 76
Задача 2.	**Ответ:** Б1 Д1 или

Ответ: Д1 Б1

Критерии оценивания:

15 баллов – правильный ответ.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

Сформируем таблицу

	Б1	Б2	Б3	Б4	Д1	Д2
Денис						
Семён						
Юрий						

Ни Денис, ни Юрий не посещают Робоквантум.

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X			
Семён						
Юрий			X			

Учащийся программы Хайтек, учащийся IT-квантум и Семён все трое родом из Пятигорска. С учётом высказывания: «На образовательных программах ребята не пересекаются»

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X			
Семён				X	X	
Юрий			X			

Учащийся программы Квантоматематика, учащийся Аэроквантум и Юрий втроем учатся в одной школе.

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X			
Семён				X	X	
Юрий	X		X			X

Денис не посещает Квантоматематику.

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X			X
Семён				X	X	
Юрий	X		X			X

Юрий является учащимся основных образовательных программ.

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X			X
Семён				X	X	
Юрий	X		X		X	X

На образовательных программах ребята не пересекаются. Ребята посещают по две образовательные программы.

	Б1	Б2	Б3	Б4	Д1	Д2
Денис			X		V	X
Семён			V	X	X	V
Юрий	X		X		X	X

=>

	Б1	Б2	Б3	Б4	Д1	Д2
Денис	V	X	X	X	V	X
Семён	X	X	V	X	X	V
Юрий	X	V	X	V	X	X

Задача 3.	**Ответ:** 120 105

Критерии оценивания:

20 баллов – правильный ответ.

10 баллов – ответ правильный, но нарушен порядок ввода чисел (**Ответ:** 105 120)

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

Каждый кувшин содержит определенную часть от первого кувшина.

	1	2	3	4	5	6	7	8
	840	420	280	210	168	140	120	105
	1	1/2	1/3	1/4	1/5	1/6	1/7	1/8

Задача 4.

Ответ: 1 2 2 2 3 3 4 3 2 3 4 3 2 1 1

Критерии оценивания:

25 баллов – правильный ответ.

19 баллов – правильный ответ, без учета условия, что Улитка в ячейку Б прибывает со скоростью 1 ячейка в минуту (**Ответ:** 1 2 2 2 3 3 4 3 2 3 4 4 3).

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Задача 5.

Ответ: 12 8

Критерии оценивания:

30 баллов – правильный ответ на оба вопроса

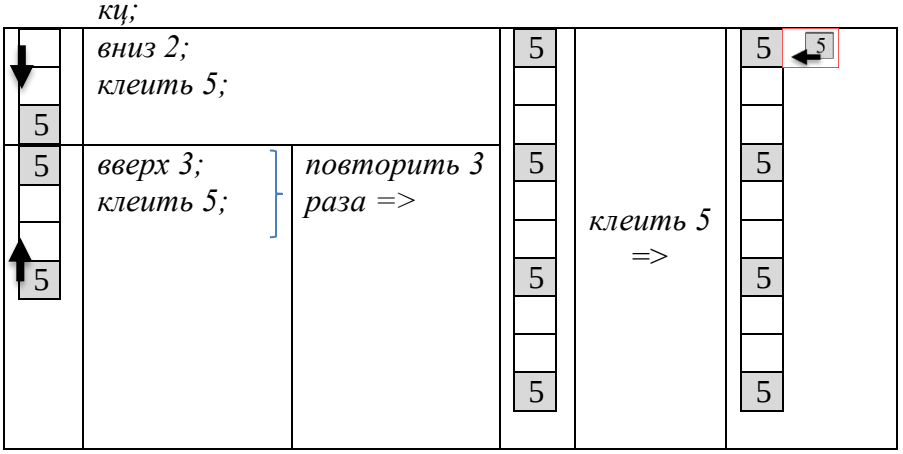
15 баллов – правильный ответ на первый вопрос.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

Проанализируем программу:

повторить 3 раз
  вниз 2;
  клеить 5;
  повторить 3 раз
    вверх 3;
    клеить 5;
  кц;
  клеить 5;
кц;



Все проделанное повторяется три раза, согласно внешнему циклу.

После первой прокрутки цикла РР заклеил пленкой цвета 5 четыре ячейки.

4*3= 12 ячеек.

Из двух последних столбцов схемы видно, что команда из восьмой строки программы *клеить 5* не изменяет результат работы алгоритма.

7-8 классы

Задача 1 (10 баллов)

Содержимое папки *Образцы* представлено во второй колонке таблицы.

1	Маска.png
2	Маскарад.docx
3	Парад.doc
4	Каскад.jpg
5	Парадокс.doc
6	Радость.jpeg
7	Такса.docx
8	Доска.jpg
9	Барбоскин.png

Учитель выполнил следующие действия:

1.Из папки *Образцы* в папку *Разное* скопировал файлы, соответствующие маске *аск*.???

2. Удалил файлы в папке *Образцы* по маскам ??акс*.d* и *ос??.*

3. В папку *Интересное и*з папки *Образцы* перенес файлы, соответствующие маске *ск??.*g

4. В папке *Образцы* упорядочил имена файлов в порядке убывания

Укажите порядок оставшихся файлов в папке *Образцы, перечислив соответствующие файлам номера из первой колонки таблицы.*

При вводе ответа отделяйте числа последовательности пробелами (запятые, точки и другие символы в ответе не использовать).

*Пример ввода ответа:**4 1 3***

Задача 2 (15 баллов)

Дана таблица «Метизы».

Номер	Деталь	Вес	В	Ш	Г	Количество
1	Подшипник	5	3	1	18	5
2	Винт	6	1	3	18	7
3	Болт	4	1	4	16	7
4	Гайка	3	6	0	15	2
5	Шайба	2	3	3	12	17
6	Ролик	3	2	4	11	7

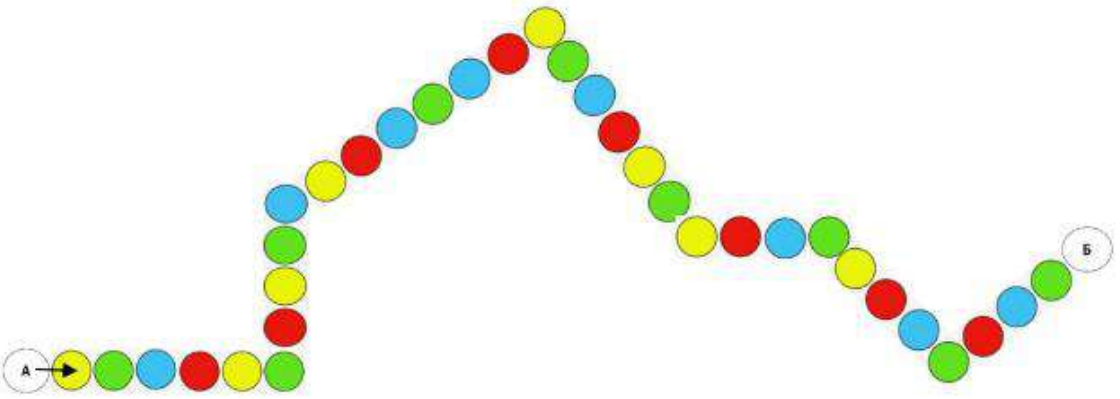
Сколько записей из этой таблицы удовлетворяют условию:

Вес >=3 И (Г<16 ИЛИ Количество>5)

При вводе ответа указывать только число (запятые, точки и другие символы в ответе не использовать).

Задача 3 (20 баллов)

Исполнитель Улитка должна добраться из пункта А в пункт Б пройдя через все ячейки цепи.



Известно, что исполнитель Улитка соблюдает правила:

– за минуту перемещаться на целое число ячеек;

– из ячейки А начинает двигаться со скоростью 1 ячейка в минуту;

– в ячейку Б прибывает со скоростью 1 ячейка в минуту;

– каждую минуту может не изменять скорость;

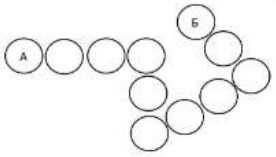
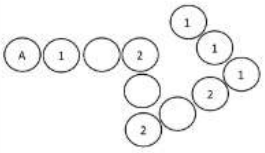
– каждую минуту может увеличивать/уменьшать скорость на 1;

– обязательно завершает текущее перемещение (заканчивается минута) перед изменением направления движения.

Сформируйте алгоритм передвижения Улитки из пункта А в пункт Б за минимальное время. В ответе укажите:

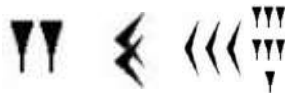
1. Время, которое потратит исполнитель
2. Минимальную по количеству чисел последовательность, соответствующую количеству пройденных клеток за каждую минуту (с учетом описанных правил).

При вводе ответа соблюдайте порядок ответов и разделяйте числа последовательности пробелами (запятые, точки и другие символы в ответе не использовать). Например, если бы маршрут имел следующий вид:

Маршрут	Расчет
	
Ввод ответа:	*1 1 1 2 2 2 1 1 1 1*

Задача 4 (25 баллов)

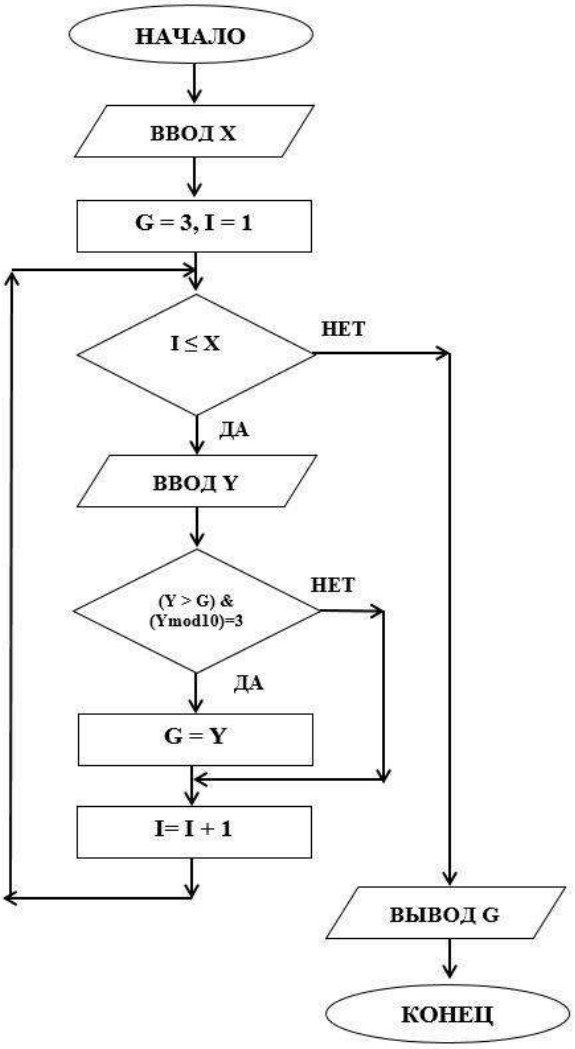
В таблице записаны числа в устаревших системах счисления (СС). Сформируйте равенства с учетом указанной системы СС. Заполнив пропуски предложенными числами.

Вавилонская СС	Десятичная СС	Двоичная СС
	?	?
Египетская СС	**Десятичная СС**	**Восьмеричная СС**
	?	?

Кириллическая СС	Десятичная СС	Шестнадцатеричная СС
ЗТМИ	?	?

Задача 5 (30 баллов)

Изучите алгоритм, представленный ниже в виде блок-схемы.
Переменные X, Y, I, G предназначены для работы с натуральными числами. X≤1000, Y≤1000.



Выберите истинные высказывания, которые соответствуют алгоритму:
- в алгоритме вводится количество чисел в последовательности, а затем сами числа
- в алгоритме вводится количество чисел в последовательности
- X Y, I, G – натуральные числа
- X, Y, I, G – целочисленные беззнаковые переменные
- в последовательности всегда имеется число, оканчивающееся на 3.
- хотя бы одно из чисел последовательности кратно 3
- 1≤X≤1000
- -0≤X≤1000

– 0≤Y≤1000

– результат работы алгоритма – нахождение максимального числа последовательности

– алгоритм выводит число, оканчивающееся на 3

Задача 1.

Ответ: 7 5 3 2 1

Критерии оценивания:

10 баллов – правильный ответ.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

1. После первого действия содержимое папки *Образцы* не изменилось.

2. Не были удалены файлы по маске ??акс*.d*, т.к. ни один файл из писка не соответствует данной маске. А после удаления файлов по маске *ос??.* в папке *Образцы* остались файлы

1	Маска.png
2	Маскарад.docx
3	Парад.doc
4	Каскад.jpg
5	Парадокс.doc
7	Такса.docx
9	Барбоскин.png

3. После переноса в папку *Интересное* файлов по маске *ск??.*g в папке *Образцы* остались:

1	Маска.png
2	Маскарад.docx
3	Парад.doc
5	Парадокс.doc
7	Такса.docx

4.В папке *Образцы* после четвертого действия имеем последовательность файлов

7	Такса.docx
5	Парадокс.doc
3	Парад.doc
2	Маскарад.docx
1	Маска.png

Задача 2.

Ответ: 4

Критерии оценивания:

15 баллов – правильный ответ.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

Г<16 ИЛИ Количество>5

Номер	Деталь	Вес	В	Ш	Г	Количество
1	Подшипник	5	3	1	18	5
2	Винт	6	1	3	18	7
3	Болт	4	1	4	16	7
4	Гайка	3	6	0	15	2
5	Шайба	2	3	3	12	17
6	Ролик	3	2	4	11	7

Вес >=3 И (Г<16 ИЛИ Количество>5)

Номер	Деталь	Вес	В	Ш	Г	Количество
2	Винт	6	1	3	18	7
3	Болт	4	1	4	16	7
4	Гайка	3	6	0	15	2
5	Шайба	2	3	3	12	17
6	Ролик	3	2	4	11	7

Результат

Номер	Деталь	Вес	В	Ш	Г	Количество
2	Винт	6	1	3	18	7
3	Болт	4	1	4	16	7
4	Гайка	3	6	0	15	2
6	Ролик	3	2	4	11	7

Задача 3.

Возможны четыре правильных ответа

Ответ1: 14 1 2 3 4 4 3 3 3 3 2 2 2 1 1

Ответ2: 14 1 2 3 4 4 3 3 3 3 2 2 1 2 1

Ответ3: 14 1 2 3 4 3 4 3 3 3 2 2 2 1 1

Ответ4: 14 1 2 3 4 3 4 3 3 3 2 2 1 2 1

Критерии оценивания:

20 баллов – правильный ответ.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Решение:

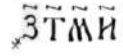
Определим время, которое потратит исполнитель, зная скорость движения за каждую минуту

1	2	3	4	4	3	3	3	3	2	2	2	1	1	Скорость, ячеек/мин
1	1	1	1	1	1	1	1	1	1	1	1	1	1	Время, мин

Результат – 14 мин.

Задача 4.

Ответ:

Вавилонская СС	Десятичная СС	Двоичная СС
	7237	1 1100 0100 0101
Египетская СС	Десятичная СС	Восьмеричная СС
	8437	20 365
Кириллическая СС	Десятичная СС	Шестнадцатеричная СС
	8347	209B

Критерии оценивания:

25 баллов – правильный ответ.

0 баллов – неверный ответ/неверно введенный ответ /задача не решена.

Задача 5.

Ответ: К истинным высказывания, которые соответствуют алгоритму, относятся:

- ✓ В алгоритме вводится количество чисел в последовательности, а затем сами числа
- ✓ В алгоритме вводится количество чисел в последовательности
- ✓ X, Y, I, G – целочисленные беззнаковые переменные
- ✓ В последовательности всегда имеется число, оканчивающееся на 3.
- ✓ $1 \leq X \leq 1000$
- ✓ $0 \leq Y \leq 1000$

	✓ Алгоритм выводит число, оканчивающееся на 3 **Критерии оценивания:** 30 баллов – отмечены все истинные высказывания. от 3 до 6 – за каждое истинное высказывание. 0 баллов – неверный ответ /задача не решена.

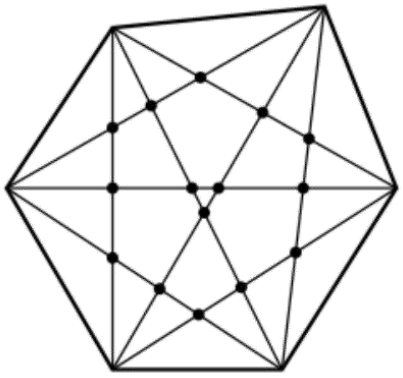
Задания для 9-11 классов

A. Сад камней

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Фэн-шуй – это китайская геомантия, наука и искусство о жизни в гармонии человека с собой и окружающим миром. Ландшафтные сады в Китае и Японии создаются по всем правилам фэн-шуй.

Васю весьма заинтересовал Фэн-шуй, и он решил разбить сад камней во дворе бабушкиного дома. Формой сада Вася выбрал выпуклый многоугольник с N сторонами. Многоугольник строится с таким условием, что никакие три диагонали в нем не пересекаются в одной точке. Камни должны быть размещены в точках пересечения диагоналей многоугольника. Соответственно, чтобы подсчитать максимально возможное количество камней необходимо найти количество точек пересечения пар диагоналей в этом многоугольнике. Помогите Васе определить максимально возможное количество камней для его японского сада камней.



На рисунке ниже изображен многоугольник с 6 сторонами.

Примечание: многоугольник является выпуклым, если все его внутренние углы меньше 180 градусов.

Формат ввода

Вводится одно целое число N $(3 \le N \le 100)$ - количесвто сторон многоугольника.

Формат вывода

Выведите целое число K — максимальное количество камней в саду Васи, т.е. количество точек пересечения диагоналей многоугольника.

Пример 1

Ввод	**Вывод**
3	0

Пример 2

Ввод	Вывод
4	1

Пример 3

Ввод	Вывод
6	15

Решение:

```
#include <stdio.h>
int main()
{
  int rez, i, n;
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);
  scanf("%d", &n);
  rez = 0;
  for (i=2; i<n-1; i++) {
    rez += (i-1)*(n-1-i);
  }
  rez *= n;
  rez /= 4;
  printf("%d\n", rez);
  fclose(stdin);
  fclose(stdout);
  return 0;
}
```

B. Бабушкин рецепт

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Васина бабушка — большая поклонница криптографии. Она шифрует все свои записи. Но бабушка стала забывчивой и часто не может вспомнить алгоритм шифрования, который она использовала для той или иной записи.

Так случилось и с любимым Васиным блюдом, рецепт которого бабушка зашифровала и теперь не может расшифровать. Васе предстоит сложная задача помочь бабушке расшифровать рецепт. Чтобы это сделать Васе необходимо в заданной строке текста найти наибольшую подстроку такую, которая встречается дважды в исходной строке, и эти

вхождения не накладываются друг на друга. Вася решил составить соответствующую программу. Помогите ему.

Формат ввода

Вводится непустая строка, состоящая из строчных букв латинского алфавита.

Длина строки не превосходит *1 000* символов.

Формат вывода

Выведите одно число — максимальную длину подстроки.

Если нет подстрок, встречающихся дважды, выведите 0.

Пример 1

Ввод	**Вывод**
abcdea	1

Пример 2

Ввод	**Вывод**
abcdeabcdf	4

Решение:

```
#include <iostream>
#include <algorithm>
#include <cmath>
#include <iomanip>
#include <string>
//#define int long long;
using namespace std;
int main()
{
	string st;
	cin >> st;
	int n = st.size(), ans = 0;
	for (int i = 0; i < n - 1; i++) {
		for (int j = i + 1; j < n; j++)
			if (st[j] == st[i]) {
				int k = 0, in2 = j, in1 = i;

	    while (st[in1] == st[in2] && in1 < j) {
					k++;
					in1++;
					in2++;
				}
			ans = max(k, ans);
```

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

C. Деревня

Ограничение времени	3 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Василий решил запустить свой квадрокоптер и посмотреть на бабушкину деревню с высоты птичьего полета. Деревня предстала перед Васиным взором как поле из $N \times M$ клеток, вертикальные стороны которого параллельны направлению с севера на юг. Каждая клетка этого поля принадлежит какому-либо домохозяйству – усадьбе. Усадьбы представляют собой связные по стороне области клеток. Дороги в деревне – все единичные отрезки, разделяющие две разные усадьбы или находящиеся на границе деревни.

Вася решил прогуляться по деревне и начал свою прогулку на северной границе. Он может идти по дорогам по следующим направлениям: запад, восток или юг. При этом, ему нельзя проходить по одной дороге дважды, даже двигаясь в различных направлениях. Вася решил прогуляться до южной границы деревни. Таким образом, Васин путь должен разделить деревню на две части: ту, которая находится восточнее Васиного пути, и ту, которая находится западнее от него. Помогите Васе определить минимальную возможную разницу площадей этих двух частей деревни.

Формат ввода

В первой строке вводятся два целых числа N и M – размеры деревни ($1 \leqslant N, M \leqslant 300$). В следующих N строках вводятся по M заглавных латинских букв — описание бабушкиной деревни. Связные по стороне области клеток, состоящие из одинаковых букв, являются усадьбами.

Формат вывода

В строке вывода должно появиться одно целое число K — минимальную возможную разницу площадей двух частей, на которые Васин путь разделит бабушкину деревню.

Пример 1

Ввод	**Вывод**
3 3 ABB BAB AAB	1

Пример 2

Ввод	Вывод
2 3 AAA BBB	0

Примечания

Иллюстрация к первому примеру.

Усадьбы представляют собой области разного цвета, черные отрезки "— дороги . Красная пунктирная линия "— возможный путь движения Василия.

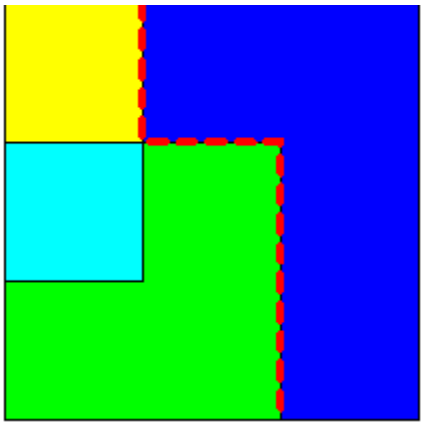
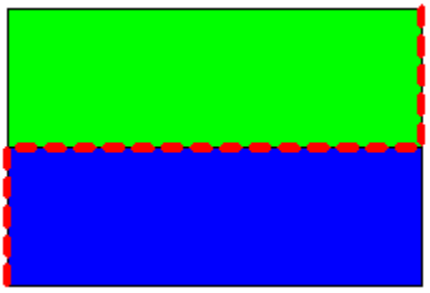


Иллюстрация ко второму примеру.



Решение:

```
#ifdef LOCAL
#  define _GLIBCXX_DEBUG
#else
#  define cerr __get_ce
#endif
#include <bits/stdc++.h>
#define ff first
#define ss second
#define szof(x) ((int)x.size())

using namespace std;
typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
int const INF = (int)1e9 + 1e3;
ll const INFL = (ll)1e18 + 1e6;
```

```
mt19937 tw(9450189);
uniform_int_distribution<ll> ll_distr;
ll rnd(ll a, ll b) { return ll_distr(tw) % (b - a + 1) + a; }

void solve() {
	int n, m;
	cin >> n >> m;
	vector<vector<char>> field(n, vector<char>(m));
	for (int i = 0; i < n; ++i) {
		string s;
		cin >> s;
		for (int j = 0; j < m; ++j) {
			field[i][j] = s[j];
		}
	}
	vector<vector<bool>> d(m + 1, vector<bool>(n * m * 2 + 1));
	for (int i = 0; i <= m; ++i) {
		d[i][n * m] = 1;
	}

	vector<vector<bool>> next(m + 1, vector<bool>(n * m * 2 + 1));

	for (int i = 0; i < n; ++i) {
		if (i) {
			for (int j = 0; j < m; ++j) {
				if (field[i - 1][j] != field[i][j]) {
					for (int k = 0; k <= n * m * 2; ++k) {
						d[j + 1][k] = d[j + 1][k] | d[j][k];
					}
				}
			}
			for (int j = m; j > 0; --j) {
				if (field[i - 1][j - 1] != field[i][j - 1]) {
					for (int k = 0; k <= n * m * 2; ++k) {
						d[j - 1][k] = d[j - 1][k] | d[j][k];
					}
				}
			}
		}
		for (int j = 0; j < m + 1; ++j) {
			fill(next[j].begin(), next[j].end(), 0);
		}
		for (int j = 0; j <= m; ++j) {
			if (j == 0 || j == m || field[i][j - 1] != field[i][j]) {
				for (int k = 0; k <= n * m * 2; ++k) {
					int to = k + j - (m - j);
					if (0 <= to && to <= n * m * 2) {
						next[j][to] = next[j][to] | d[j][k];
					}
				}
			}
		}
```

```
                swap(next, d);

        }

        int ans = INF;
        for (int i = 0; i < m + 1; ++i) {
                for (int j = 0; j <= n * m * 2; ++j) {
                        if (d[i][j]) {
                                ans = min(ans, abs(j - n * m));
                        }
                }
        }

        cout << ans << "\n";
}
int main() {
#ifdef LOCAL
        auto start_time = clock();
        cerr << setprecision(3) << fixed;
#endif
        cout << setprecision(15) << fixed;
        ios::sync_with_stdio(false);
        cin.tie(nullptr);

        int test_count = 1;
        // cin >> test_count;
        for (int test = 1; test <= test_count; ++test) {
                solve();
        }

#ifdef LOCAL
        auto end_time = clock();
        cerr << "Execution time: " << (end_time - start_time) * (int)1e3 / CLOCKS_PER_SEC << "
ms\n";
#endif
}
```

D. Грядки

Ограничение времени	3 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Бабушка дала Васе задание прополоть грядки у себя в огороде. Но бабушка тоже увлекается фэн-шуем, поэтому её грядки весьма сложны и полоть их непросто. Чтобы не повредить растущие овощи двигаться надо циклически.

Каждая грядка схематически представляет из себя поле $n_i \times m_i$ ячеек, в каждой из которых написана одна латинская буква — вид овоща. Циклом на таком поле называется

последовательность ячеек, в которой каждая пара соседних ячеек (в том числе, первая и последняя) имеют общую сторону. Простым циклом называется цикл, в который ни одна ячейка не входит дважды. Два цикла пересекаются, если они оба содержат одну и ту же ячейку. На грядке можно выделить некоторое количество непересекающихся простых циклов, каждый из которых проходит по ячейкам, в которых находится один вид овощей.

Вася не знает, сколько таких циклов он должен выделить, и сколько вариантов ему придется перебрать чтобы выбрать оптимальный и прополоть грядку.

Василию необходимо определить *максимальное количество* непересекающихся простых циклов для каждой грядки, а также *количество различных способов* выделить максимальное число таких циклов. Либо сообщить, что количество способов превышает 10^{18}. Два способа являются различными, если в одном из них две ячейки принадлежат одному циклу, а в другом — нет.

Формат ввода

В первой строке входных данных вводится единственное целое число k — количество грядок ($1 \le k \le 20$).

Далее вводится описание k грядок.

Описание каждой грядки начинается со строки, в которой содержится два целых числа n_i и m_i — размеры i-го поля ($1 \le n_i \cdot m_i \le 160$).

В следующих n_i строках содержится по m_i строчных латинских букв — расположение овощей на i-ой грядке.

Формат вывода

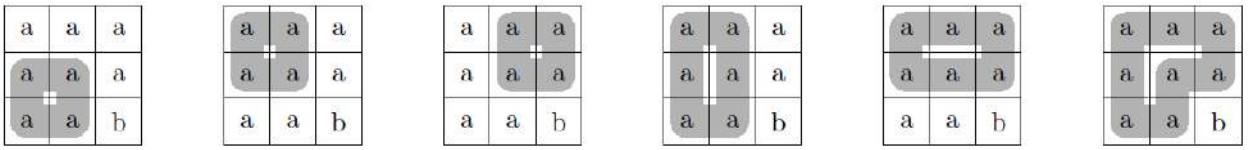
Для каждоой грядки выводится в отдельной строке два целых числа — максимальное количество простых непересекающихся циклов, проходящих по ячейкам с одинаковой буквой, которые можно выделить на поле i-ой грядки, и количество способов это сделать. Если количество способов строго больше 10^{18}, выводится вместо этого -1.

Пример

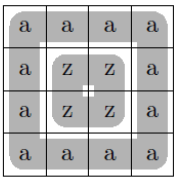
Ввод	**Вывод**
3 3 3 aaa aaa aab 4 4 aaaa azza azza aaaa 5 24 nsssnseeeeswsssssswttttt nnssnsessssswssssswssstss nsnsnseessswsswsswssstss nssnnsessssswswswssstss nsssnseeeessswswssssstss	1 6 2 1 11 486

Примечания

Все варианты выделения одного цикла на первой грядке:



Единственный способ выделить два цикла на второй грядке:



Решение:

```
#ifdef LOCAL
#  define _GLIBCXX_DEBUG
#else
#endif
#include <bits/stdc++.h>
#define ff first
#define ss second
#define szof(x) ((int)x.size())


using namespace std;
typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef unsigned long long ull;
int const INF = (int)1e9 + 1e3;
ll const INFL = (ll)1e18 + 1e6;
mt19937 tw(9450189);
uniform_int_distribution<ll> ll_distr;
ll rnd(ll a, ll b) { return ll_distr(tw) % (b - a + 1) + a; }

const int MAXN = 12;

vector<vector<int>> states[MAXN];
map<ull, int> ind_state[MAXN];
vector<ull> hashes[MAXN];

vector<vector<vector<vector<tuple<int, int, int, int>>>>> go[MAXN];

void solve() {
	int n, m;
	cin >> n >> m;
	vector<vector<char>> field(n, vector<char>(m));
	for (int i = 0; i < n; ++i) {
		string s;
		cin >> s;
		for (int j = 0; j < m; ++j) {
			field[i][j] = s[j];
```

```
		}
	}

	if (n > m) {
		vector<vector<char>> field2(m, vector<char>(n));
		for (int i = 0; i < n; ++i) {
			for (int j = 0; j < m; ++j) {
				field2[j][i] = field[i][j];
			}
		}
		swap(field, field2);
		swap(n, m);
	}

	vector<vector<pair<int, ll>>> d(2, vector<pair<int, ll>>(szof(states[n - 1]), {-INF, 0}));
	vector<vector<pair<int, ll>>> next(2, vector<pair<int, ll>>(szof(states[n - 1])));
	d[0][0] = {0, 1};

	for (int i = 0; i < m; ++i) {
		for (int j = 0; j < n; ++j) {
			fill(next[0].begin(), next[0].end(), make_pair(-INF, 0));
			fill(next[1].begin(), next[1].end(), make_pair(-INF, 0));
			for (int last = 0; last < 2; ++last) {
				for (int k = 0; k < szof(states[n - 1]); ++k) {
					for (auto [mask, st, add, nlast] : go[n - 1][last][j][k]) {
						if (mask & 1) {
							if (field[j - 1][i] != field[j][i]) {
								continue;
							}
						}
						if (mask & 2) {
							if (!i || field[j][i - 1] != field[j][i]) {
								continue;
							}
						}
						if (next[nlast][st].ff < d[last][k].ff + add) {
							next[nlast][st] = {d[last][k].ff + add,
d[last][k].ss};
						} else if (next[nlast][st].ff == d[last][k].ff + add) {
							next[nlast][st].ss += d[last][k].ss;
							if (next[nlast][st].ss > INFL) {
								next[nlast][st].ss = INFL;
							}
						}
					}
				}
			}
			swap(next, d);
		}
	}

	const ll threshold = 1000000000000000000ll;
```

```
	int sz = d[0][0].ff;
	ll sum = d[0][0].ss;
	if (d[1][0].ff > sz) {
		sz = d[1][0].ff;
		sum = d[1][0].ss;
	} else if (d[1][0].ff == sz) {
		sum += d[1][0].ss;
	}

	cout << sz << " ";
	if (sum > threshold) {
		cout << "-1\n";
	} else {
		cout << sum << "\n";
	}
}

int main() {
#ifdef LOCAL
	auto start_time = clock();
	cerr << setprecision(3) << fixed;
#endif
	cout << setprecision(15) << fixed;
	ios::sync_with_stdio(false);
	cin.tie(nullptr);

	for (int n = 1; n <= MAXN; ++n) {
		vector<int> cur(n, -1);
		vector<bool> used(n);

		function<void(int)> rec = [&](int pos) {
			if (pos == n) {
				states[n - 1].push_back(cur);
				ull h = 0;
				for (int p : cur) {
					h = h * 23 + p + 1;
				}
				hashes[n - 1].push_back(h);
				ind_state[n - 1][h] = szof(states[n - 1]) - 1;

		return;
			}
			rec(pos + 1);
			if (!used[pos]) {
				for (int i = pos + 1; i < n; ++i) {
					if (used[i]) {
						break;
					}
					cur[pos] = i;
					cur[i] = pos;
					used[i] = true;
```

```
                                        rec(pos + 1);
                                        used[i] = false;
                                        cur[pos] = cur[i] = -1;
                                }
                        }
                };

                vector<ull> pow23 = {1};
                for (int i = 0; i < n; ++i) {
                        pow23.push_back(pow23.back() * 23);
                }

                function<ull(int, int, int)> change = [&](int pos, int val1, int val2) {
                        return pow23[n - pos - 1] * (val2 - val1);
                };

                rec(0);

                go[n - 1] = vector<vector<vector<vector<tuple<int, int, int, int>>>>>(2,
vector<vector<vector<tuple<int, int, int, int>>>>(n, vector<vector<tuple<int, int, int,
int>>>(szof(states[n - 1]))));

                for (int i = 0; i < n; ++i) {
                        for (int j = 0; j < szof(states[n - 1]); ++j) {
                                for (int last = 0; last < 2; ++last) {
                                        if (last && i > 0 && states[n - 1][j][i - 1] != -1) {
                                                continue;
                                        }
                                        int mask = 0;
                                        if (states[n - 1][j][i] != -1) {
                                                mask |= 2;
                                        }
                                        go[n - 1][last][i][j].push_back({mask, j, 0, 0});
                                        if (i) {
                                                if (states[n - 1][j][i] != -1 && states[n - 1][j][i - 1] != -
1) {
                                                        if (states[n - 1][j][i] == i - 1 && states[n -
1][j][i - 1] == i) {
                                                                ull nh = hashes[n - 1][j] + change(i,
states[n - 1][j][i], -1) + change(i - 1, states[n - 1][j][i - 1], -1);
                                                                go[n - 1][last][i][j].push_back({mask |
1, ind_state[n - 1][nh], 1, 1});
                                                        } else {
                                                                int u1 = states[n - 1][j][i];
                                                                int u2 = states[n - 1][j][i - 1];
                                                                ull nh = hashes[n - 1][j] + change(u1, i,
u2) + change(i, u1, -1) + change(u2, i - 1, u1) + change(i - 1, u2, -1);
                                                                go[n - 1][last][i][j].push_back({mask |
1, ind_state[n - 1][nh], 0, 1});
                                                        }
                                                } else if (states[n - 1][j][i] == -1 && states[n - 1][j][i -
1] == -1) {
```

```
                                                if (!last) {
                                                        ull nh = hashes[n - 1][j] + change(i, -1, i
- 1) + change(i - 1, -1, i);
                                                        go[n - 1][last][i][j].push_back({mask |
1, ind_state[n - 1][nh], 0, 0});
                                                }
                                        } else if (states[n - 1][j][i] == -1) {
                                                int e = states[n - 1][j][i - 1];
                                                ull nh = hashes[n - 1][j] + change(i, -1, e) +
change(i - 1, e, -1) + change(e, i - 1, i);
                                                go[n - 1][last][i][j].push_back({mask | 1,
ind_state[n - 1][nh], 0, 0});
                                        } else {
                                                if (!last) {
                                                        int e = states[n - 1][j][i];
                                                        ull nh = hashes[n - 1][j] + change(i, e, -
1) + change(i - 1, -1, e) + change(e, i, i - 1);
                                                        go[n - 1][last][i][j].push_back({mask |
1, ind_state[n - 1][nh], 0, 1});
                                                }
                                        }
                                }
                        }
                }
        }

        cerr << "here " << clock() * 1000 / CLOCKS_PER_SEC << endl;

        int test_count = 1;
        cin >> test_count;
        for (int test = 1; test <= test_count; ++test) {
                solve();
        }

#ifdef LOCAL
        auto end_time = clock();
        cerr << "Execution time: " << (end_time - start_time) * (int)1e3 / CLOCKS_PER_SEC << "
ms\n";
#endif
}
```

Председатель организационного комитета
Открытой олимпиады СКФУ среди учащихся
образовательных организаций
«45 параллель» **_____Иванов В. А.**

Критерии оценивания письменных олимпиадных работ

1. Задание считаются решенными, если получен верный результат, выполнены необходимые действия и их обоснование, ведущие к этому результату. Максимальная сумма баллов за работу – 100.

2. Любое полностью правильное выполнение заданий оценивается в наибольшее количество баллов.

3. Правильный ответ, приведенный без обоснования или полученный из неправильных рассуждений, не учитывается.

4. Полный балл выставляется при правильном и полном выполнении задания.

5. Если задание не выполнено или при выполнении допущена принципиальная ошибка, тозадание оценивается в «0» баллов.

6. Если задание выполнено, но: – допущена грубая ошибка – снимается 50% от числа баллов, которыми оценено данное задание; – допущена негрубая ошибка – снимается 30% от числа баллов, которыми оценено данное задание; – допущены грамматические ошибки, небрежности – снимается 10% за каждую грамматическую ошибку или небрежность, но не более 15 баллов со всей работы.

7. К грубым ошибкам относятся ошибки, которые обнаруживают незнание учащимися формул, правил, основных свойств, теорем и неумение их применять; незнание приемов решения задач, а также вычислительные ошибки, если они не являются опиской.

8. К недочетам относятся: описки, недостаточность или отсутствие пояснений, обоснований в решениях.

9. Все ошибки, выявленные в ходе проверки олимпиадных работ, отмечаются красной пастой.

10. Количество победителей и призеров олимпиады не должно превышать 45 % от общего количества участников соответствующего этапа олимпиады.

Председатель организационного комитета
Открытой олимпиады СКФУ среди учащихся
образовательных организаций
«45 параллель» **_________________Иванов В. А.**