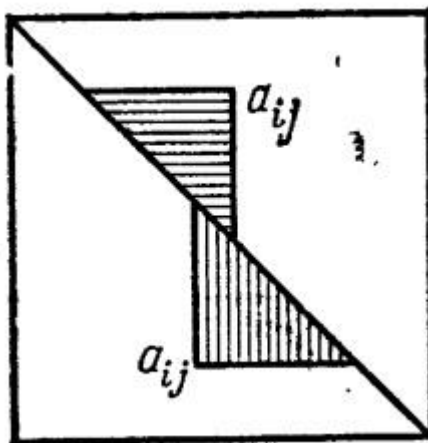


**Тема:** циклы и массивы

**Вариант: 1.1.3**

**Задача:** По заданной матрице  $A = \|a_{ij}\|$  построить матрицу  $B = \|b_{ij}\|$  того же размера, что и матрица  $A$ . Элементы  $b_{ij}$  равен минимальному элементу треугольника в  $A$ , определяемого элементом  $a_{ij}$  :



**Формат входных данных:**

В первой строке входного файла записаны значения  $W$  – количество столбцов и  $H$  – количество строк матрицы. Считать, что  $W$  и  $H$  ограничены значением 255.

В следующих  $H$  строках записано по  $W$  целых чисел, задающих исходную матрицу.

**Формат выходных данных:**

В первой строке записать количество совпадающих элементов матриц  $A$  и  $B$ .

В следующих  $H$  строках выходного файла записать получившуюся матрицу  $B$ .

**Пример входных и выходных данных:**

input.txt	output.txt
3 3 1 2 3 4 0 6 7 8 -1	3 1 0 -1 0 0 -1 -1 -1 -1
3 2 2 7 -13 10 8 7	4 2 2 -13 2 8 7

**Тема:** Строки

**Вариант: 2.1.2**

**Задача:** Написать программу, которая находит все слова в тексте, подходящие под заданный шаблон.

В шаблоне могут быть использованы следующие спецсимволы:

1. “\d” соответствует любому цифровому символу.
2. “\D” соответствует символу буквы английского алфавита.
3. Любой конкретный символ, обозначающий вхождение самого этого символа: ‘а’, ‘b’, ‘c’, и т.д.;  
В качестве конкретного символа не используются ‘\’, ‘~’, ‘[’, ‘]’, ‘(’, ‘)’, ‘\*’.

Кроме того, в шаблоне могут быть использованы следующие конструкции:

1. ~x, где x – это конкретный символ. Обозначает, что на данной позиции может стоять любой символ кроме x (снова запрещены символы ‘\’, ‘~’, ‘[’, ‘]’, ‘(’, ‘)’, ‘\*’).
2. [N\*(...)], где N – натуральное число, а в круглых скобках записано выражение по общим правилам. Означает, что вместо данного выражения в квадратных скобках должно стоять N штук выражений в круглых скобках.

Вложенные конструкции [] и ~ в шаблонах не используются

**Пример:**

Под шаблон “a[3\*(\d\D)]~b” подходят строки: “a1z3q4P9”, “a1p1p1pa”, “a9q8p7o1” и т.д.

**Формат входных данных:**

В первой строке входного файла записан шаблон по приведенным выше правилам.

Считать, что шаблон ограничен 64 символами.

Далее записано число K.

В конце файла идет K строк. Считать, что каждая строка ограничена 1024 символами.

**Формат выходных данных:**

В выходной файл через пробел записать номера строк, подходящих под указанный шаблон.

Нумерация строк идет с нуля. Указывать строки в том же порядке, в каком они были записаны во входных данных. Если никакая из строк не подошла, вывести “none”.

**Пример входных и выходных данных:**

input.txt	output.txt
a[3*(\d\D)]~b 5 a33b a1z3q4P9 b1z3q4Pa a9q8p7o1 a1A2B3Ca	1 3 4
123\Dabc 3 123abc 123Qabc 1239abc	1
[10*(\d)] 3 12345 Abc 12c	none

**Дополнительное задание:**

Добавить возможность использования конструкции <...>\* допускающей произвольное количество вхождений выражения в треугольных скобках подряд.

**Тема:** Рекурсия и сортировки

**Вариант:** 3.1.2

**Задача:**

Реализовать **пирамидальную** сортировку на множестве целочисленных квадратных матриц размерности. Считать, что одна матрица больше другой, если ее определитель больше, чем определитель другой. Вычисление определителя реализовать с помощью формулы разложения по первой строке.

**Формат входных данных:**

В первой строке входного файла записано число  $N$  – количество матриц.

Далее следуют  $N$  квадратных матриц, каждая из которых задана следующим образом:

- В первой строке записано число  $k$  – размерность матрицы
- В следующих  $k$  строках записаны по  $k$  элементов матрицы.

Считать, что  $N \leq 100$  и  $k \leq 64$

**Формат выходных данных:**

В выходном файле записать матрицы в порядке возрастания их определителей.

**Пример входных и выходных данных:**

input.txt	output.txt
3 2 1 2 3 4 1 5 3 1 0 0 0 1 0 0 0 1	1 2 3 4 1 0 0 0 1 0 0 0 1 5
5 1 -18 2 1 3 3 2 1 -1 3 -3 0 0 0 -3 0 0 0 -3 1 100	-3 0 0 0 -3 0 0 0 -3 -18 1 3 3 2 -1 100

**Дополнительное задание:**

Реализовать систему для сравнения временной эффективности реализаций алгоритмов сортировки на различных примерах.

Должна присутствовать следующая функциональность:

1. Многократный запуск алгоритма на одних и тех же данных, замеры времени выполнения алгоритма для каждого запуска, нахождение выборочного среднего и [стандартного отклонения](#).
2. На заданном наборе входных данных реализовать нахождение лучшего и худшего запуска алгоритма (сравнивать по среднему), печать соответствующих данных на экран.
3. Генерацию случайного набора входных данных достаточной длины и запуск выбранного алгоритма сортировки на нем.

Протестировать реализованный в задаче алгоритм в данной системе. Кроме того, реализовать простую сортировку (пузырьком или выбором), и повторить эксперимент с ней.

**Тема:** Работа с динамической памятью; Стеки, очереди и бинарные деревья.

**Вариант: 4.1.1**

**Задача:**

Реализовать программу для работы с бинарным деревом поиска, ключами в котором являются строки.

На вход программе подается текст на английском языке, список слов и значение  $k$ .

Необходимо:

- Построить бинарное дерево поиска, ключами которого являются слова из текста. Сравнивать слова лексикографически. Если слово уже было в дереве, ничего не делать. Слова в разном регистре считать разными
- Удалить из дерева слова, приведенные в списке.
- В качестве ответа вывести
  1. Количество элементов, оставшихся в дереве после удаления слов из списка.
  2. Слова, находящиеся в дереве на уровне  $k$ .

**Замечание:**

Для прохождения тестов необходимо добавлять левое поддереву к самому левому потомку правого поддерева при удалении элемента.

**Формат входных данных:**

Первой строкой входного файла всегда является строка «TEXT:»

Далее следует текст, словами из которого необходимо заполнить бинарное дерево поиска.

Размер текста заранее не известен.

Следующей строкой после текста является строка «DELETE:»

Далее следуют слова, которые необходимо удалить из дерева по одному в строке.

Следующей строкой после слова для удаления является строка «LEVEL:»

В последней строке записано число  $k$ .

**Формат выходных данных:**

В первую строку выходного файла записать количество элементов, которое осталось в дереве после удаления слов из списка.

Во вторую строку выходного файла записать слова, находящиеся в дереве на уровне  $k$ . Слова записывать от меньшего к большему в лексикографическом порядке.

**Пример входных и выходных данных:**

input.txt	output.txt
TEXT: you can't make an omelet without breaking a few eggs. DELETE: without you make LEVEL: 2	7 a breaking few
TEXT: 6 3 1 5 8 7 9 6 3 1 DELETE: 5 8 LEVEL: 2	5 1 7

**Дополнительное задание:** кроме бинарного дерева поиска реализовать и заиспользовать сбалансированное: либо красно-черное, либо AVL дерево.

Для проверки такой модификации задачи подготовить набор тестов, демонстрирующих корректность реализации сбалансированного дерева (тесты в системе тестирования при этом проходить перестанут, т.к. они ожидают именно BST).

**Тема:** Графы

**Вариант: 5.1.1**

**Задача:**

Во время очередных своих приключений Индиана Джонс нашел гробницу древних индейцев, в которой по легенде были спрятаны огромные сокровища. Гробница представляла собой сеть комнат и переходов, в каждом из которых, естественно, была смертоносная ловушка для защиты сокровищ. В некоторых комнатах находились сундуки с различными сокровищами, в других лишь мелкие побрякушки.

К счастью, Индиана Джонс выторговал у старого индейца в резервации подробную карту комнат и переходов гробницы. В карте указано, где и какие именно сокровища нужно искать, а также перечислены все ловушки, расположенные в переходах.

Индеец также подарил нашему герою магический амулет, который сможет защитить его от ловушек. У амулета есть заряд, после того, как он спасает от очередной ловушки, заряд уменьшается. Разные ловушки уменьшают заряд на разную величину, но в карте эти величины указаны.

Единственным условием, которое поставил индеец, было то, что Индиана Джонс заберет лишь одно сокровище. Как только он возьмет его в руки, все ловушки отключатся, и он сможет спокойно выйти из гробницы.

Естественно, Индиана Джонс был знаком с теорией графов. С ее помощью он без труда добрался до самого ценного (насколько позволял заряд амулета) сокровища и вынес его из гробницы.

Вам необходимо написать программу, которая по заданной карте и заряду амулета возвращает стоимость сокровища, которое вынес из гробницы Индиана Джонс.

Не забывайте, что Индиана Джонс всегда торопится, а переходы в подземелье существуют совсем не между каждой парой комнат. Стоит учитывать это для реализации наиболее эффективного алгоритма.

**Формат входных данных:**

В первой строке входного файла записаны три значения:

$P$   $N$   $M$  – заряд амулета; количество комнат; количество переходов;

В следующих  $M$  строках описаны переходы и ловушки в формате:

*from to value* – номер комнаты, откуда идет переход; номер комнаты, куда идет переход; значение, на которое ослабляет амулет ловушка из этого перехода;

В следующих  $N$  строках записаны стоимости сокровищ, находящихся в соответствующих комнатах. ( $k$ -ая строка соответствует  $k$ -ой комнате)



Считать, что начальная комната имеет номер 0.



**Формат выходных данных:**

В выходной файл необходимо записать стоимость сокровища, которое вынес из гробницы Индиана Джонс.

**Пример входных и выходных данных:**

input.txt	output.txt
	
15 4 4 0 1 5 0 2 10 1 3 12 2 3 5 5 10 20 21	21
17 8 12 0 6 12 0 2 15 2 6 5 2 5 1 6 5 3 5 3 1 6 7 15 7 3 2 3 4 88 5 4 5 5 1 11 1 4 2 5 99 10 38 102 30 1 999	38