# Задача D. Анти-QuickSort (2 балла)

| | |
|---|---|
| Имя входного файла: | `antiqs.in` |
| Имя выходного файла: | `antiqs.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Для сортировки последовательности чисел широко используется быстрая сортировка — QuickSort. Далее приведена программа, которая сортирует массив `a`, используя этот алгоритм.

```
var a : array [1..N] of integer;

procedure QSort(left, right : integer);
var i, j, key, buf : integer;
begin
    key := a[(left + right) div 2];
    i := left;
    j := right;
    repeat
        while a[i] < key do     {первый while}
            inc(i);
        while key < a[j] do     {второй while}
            dec(j);
        if i <= j then begin
            buf := a[i];
            a[i] := a[j];
            a[j] := buf;
            inc(i);
            dec(j);
        end;
    until i > j;
    if left < j then QSort(left, j);
    if i < right then QSort(i, right);
end;

begin
    ...
    QSort(1, N);
end.
```

Хотя QuickSort является самой быстрой сортировкой в среднем, существуют тесты, на которых она работает очень долго. Оценивать время работы алгоритма будем количеством сравнений с элементами массива (то есть суммарным количеством сравнений в первом и втором while). Требуется написать программу, генерирующую тест, на котором быстрая сортировка сделает наибольшее число таких сравнений.

## Формат входного файла

В первой строке находится единственное число $n$ ($1 \le n \le 70000$).

## Формат выходного файла

Вывести перестановку чисел от 1 до $n$, на которой быстрая сортировка выполнит максимальное число сравнений. Если таких перестановок несколько, вывести любую из них.

## Пример

| antiqs.in | antiqs.out |
|---|---|
| 3 | 1 3 2 |

# Задача E. K-ая порядковая статистика (2 балла)

| | |
|---|---|
| Имя входного файла: | `kth.in` |
| Имя выходного файла: | `kth.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Дан массив из $n$ элементов. Какое число $k$-ое в порядке возрастания в этом массиве?

## Формат входного файла

В первой строке входного файла содержатся два числа $n$ — размер массива и $k$.($1 \le k \le n \le 3 \cdot 10^7$). Во второй строке находятся числа $A$, $B$, $C$, $a_1$, $a_2$ по модулю не превосходящие $10^9$. Вы должны получить элементы массива начиная с третьего по формуле: $a_i = A * a_{i-2} + B * a_{i-1} + C$. Все вычисления должны производится в 32 битном знаковом типе, переполнения должны игнорироваться.

## Формат выходного файла

Выведите $k$-ое в порядке возрастания число в массиве $a$.

## Пример

| kth.in | kth.out |
|---|---|
| 5 3<br>2 3 5 1 2 | 13 |
| 5 3<br>200000 300000 5 1 2 | 2 |

Во втором примере элементы массива $a$ равны: (1, 2, 800005, −516268571, 1331571109).

## Примечание

Нельзя использовать встроенные сортировки и функции нахождения k-й порядковой статистики.

# Задача B. Пирамидальная сортировка (2 балла)

| | |
|---|---|
| Имя входного файла: | `sort.in` |
| Имя выходного файла: | `sort.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью пирамидальной сортировки (heap sort). За решения, основанные на любых других сортировках, баллы ставиться не будут.

## Формат входного файла

В первой строке входного файла содержится число $n$ ($1 \le n \le 100000$) — количество элементов в массиве. Во второй строке находятся $n$ целых чисел, по модулю не превосходящих $10^9$.

## Формат выходного файла

В выходной файл надо вывести этот же массив в порядке неубывания, между любыми двумя числами должен стоять ровно один пробел.

## Пример

| sort.in | sort.out |
|---|---|
| 10<br>1 8 2 1 4 7 3 2 3 6 | 1 1 2 2 3 3 4 6 7 8 |

## Примечание

Необходимо написать свою сортировку кучей.

# Задача C. Цифровая сортировка (2 балла)

| | |
|---|---|
| Имя входного файла: | `radixsort.in` |
| Имя выходного файла: | `radixsort.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Дано $n$ строк, выведите их порядок после $k$ фаз цифровой сортировки.

## Формат входного файла

В первой строке входного файла содержится число $n$ — количество строк, $m$ — их длина и $k$ – число фаз цифровой сортировки ($1 \le n \le 1000$, $1 \le k \le m \le 1000$). В следующих $n$ строках находятся сами строки.

## Формат выходного файла

Выведите строки в порядке в котором они будут после $k$ фаз цифровой сортировки.

## Пример

| radixsort.in | radixsort.out |
|---|---|
| 3 3 1<br>bbb<br>aba<br>baa | aba<br>baa<br>bbb |
| 3 3 2<br>bbb<br>aba<br>baa | baa<br>aba<br>bbb |
| 3 3 3<br>bbb<br>aba<br>baa | aba<br>baa<br>bbb |

## Примечание

Необходимо написать свою цифровую сортировку.

# Задача D. Приоритетная очередь (3 балла)

| | |
|---|---|
| Имя входного файла: | `priorityqueue.in` |
| Имя выходного файла: | `priorityqueue.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Реализуйте приоритетную очередь. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Все операции нумеруются по порядку, начиная с единицы. Гарантируется, что размер очереди в процессе выполнения команд не превысит $10^6$ элементов.

## Формат входного файла

Входной файл содержит описание операций с очередью. Операции могут быть следующими:

- `push` $x$ — требуется добавить элемент $x$ в очередь.
- `extract-min` — требуется удалить из очереди минимальный элемент и вывести его в выходной файл. Если очередь пуста, в выходной файл требуется вывести звездочку *.
- `decrease-key` $x$ $y$ — требуется заменить значение элемента, добавленного в очередь операцией `push` в строке входного файла номер $x$, на $y$. Гарантируется, что на строке $x$ действительно находится операция `push`, что этот элемент не был ранее удален операцией `extract-min`, и что $y$ меньше, чем предыдущее значение этого элемента.

В очередь помещаются и извлекаются только целые числа, не превышающие по модулю $10^9$.

## Формат выходного файла

Выведите последовательно результат выполнения всех операций `extract-min`, по одному в каждой строке выходного файла. Если перед очередной операцией `extract-min` очередь пуста, выведите вместо числа звездочку *.

## Пример

| priorityqueue.in | priorityqueue.out |
|---|---|
| push 3<br>push 4<br>push 2<br>extract-min<br>decrease-key 2 1<br>extract-min<br>extract-min<br>extract-min | 2<br>1<br>3<br>* |

## Примечание

Необходимо написать свою структуру данных.

# Задача B. Очередь (1 балл)

| | |
|---|---|
| Имя входного файла: | `queue.in` |
| Имя выходного файла: | `queue.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "`+ N`", либо "`-`". Команда "`+ N`" означает добавление в очередь числа $N$, по модулю не превышающего $10^9$. Команда "`-`" означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит $10^6$ элементов.

## Формат входного файла

В первой строке содержится количество команд — $M$ ($1 \leqslant M \leqslant 10^6$). В последующих строках содержатся команды, по одной в каждой строке.

## Формат выходного файла

Выведите числа, которые удаляются из очереди, по одному в каждой строке. Гарантируется, что извлечения из пустой очереди не производится.

## Пример

| queue.in | queue.out |
|---|---|
| 4<br>+ 1<br>+ 10<br>-<br>- | 1<br>10 |

## Примечание

Необходимо написать свою очередь.

# Задача C. Правильная скобочная последовательность (1 балл)

| | |
|---|---|
| Имя входного файла: | `brackets.in` |
| Имя выходного файла: | `brackets.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов '`(`', '`)`', '`[`' и '`]`'. Выясните, является ли она правильной скобочной последовательностью с двумя типами скобок.

Подсказка: используйте стек.

## Формат входного файла

Входной файл содержит $1 \le n \le 500$ строк, каждая из которых содержит скобочную последовательность длиной $1 \le l \le 10^4$.

## Формат выходного файла

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

## Пример

| brackets.in | brackets.out |
|---|---|
| `()()`<br>`([])`<br>`([)]`<br>`([]]`<br>`)(` | YES<br>YES<br>NO<br>NO<br>NO |

## Примечание

Необходимо написать свой стек.

# Задача D. Постфиксная запись (2 балла)

| | |
|---|---|
| Имя входного файла: | `postfix.in` |
| Имя выходного файла: | `postfix.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел $A$ и $B$ записывается как $A\ B\ +$. Запись $B\ C\ +\ D\ *$ обозначает привычное нам $(B+C)*D$, а запись $A\ B\ C\ +\ D\ *\ +$ означает $A+(B+C)*D$. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

Подсказка: используйте стек.

## Формат входного файла

В единственной строке записано выражение в постфиксной записи, содержащее однозначные числа и операции $+$, $-$, $*$. Строка содержит не более 100 чисел и операций.

## Формат выходного файла

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений по модулю меньше $2^{31}$.

## Пример

| postfix.in | postfix.out |
|---|---|
| 8 9 + 1 7 - * | -102 |

## Примечание

Необходимо написать свой стек.