# CS F211
# Data Structures and Algorithms
# Assignment - 8

## Stacks, Queues and Select

Allowed Language: **C**

February 26, 2024

## General Tips

- **Note:** For problems A-G, use stacks/queues/deques by creating necessary push and pop functions. You are free to use either linked list/ array for the inner implementation.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.
- Use `scanf` to read characters/strings from STDIN. Avoid using `getchar`, `getc` or `gets`. Try to read up about character suppression in `scanf` as it will be very helpful in some of the problems.
- Use `printf` instead of `putc`, `putchar` or `puts` to print character/string output on STDOUT.
- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, Use comments wherever necessary.
- Use a proper IDEs like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily.
- **Note:** Kindly try to do all of these questions by yourself at least once. Spend some time thinking about it, or trying to code it instead of directly asking help of your friends or searching it up online. This helps you understand the question, allowing you to solve further questions which are not in the scope of this Assignment yourself.

# A: Internship Drive

As you might have heard from your seniors, that Internship and Placements are very tough. They ask tough problems like those from Dynamic Programming, Disjoint Sets, Segment Trees, Tries and whatnot. So seeing the opportunity of sitting for startups in Launchpad, you decide to give your luck a try. Soon you realise that your seniors were just gaslighting you into doing tough problems. Your interview problem was as follows:

Given the characters "(", ")", "[", "]", "{" and "}", check whether the given string is a valid balanced parentheses string. In order to be balanced, each opening bracket should be closed with the same type and in correct order.

A valid parentheses can be defined as a sequence where the count of opening parentheses "(", "[", "{" is equal to the count of closing parentheses ")", "]", "}", and every closing parenthesis has a corresponding opening parenthesis that precedes it and is at the same nesting level.

## Input

The first line contains a string s ($1 \leq |s| \leq 10^6$).

## Output

"YES" if the string is a valid parentheses string, otherwise "NO" (without the quotes).

input

```
{[](})
```

output

```
NO
```

input

```
[([])[]]
```

output

```
YES
```

input

```
[][][]
```

output

```
YES
```

# B: Kira is Blind Now

Kira was solving too much CoderAt problems recently, which led to him having a blurry vision. Due to this his doctor gave him special specs to see arrays, but he could just see $k$ contiguous elements at once. Since the rich guy he is, he decided to take an array of money, and find the maximum money he could get from each $k$-sized window he could observe.

## Input

The first line contains two integers $n$ and $k$ ($1 \leq k \leq n \leq 10^6$) - the number of elements in the array and the size of window visible to Kira respectively.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($10^{-9} \leq a_i \leq 10^9$).

## Output

$n - k + 1$ space separated integers containing the maximum in each window.

---

```
input
8 3
1 3 -1 -3 5 3 6 7

output
3 3 5 5 6 7
```

---

```
input
1 1
1

output
1
```

---

```
input
10 5
29 27 83 10 13 56 23 10 23 90

output
83 83 83 56 56 90
```

---

# C: DSA IS NOT EASY!

Once there was this guy called Aljun Twiggy. He always met Nom after every lab and said that it was too easy. So one fine day, he locked Twiggy up in a 4 sq. ft. room and told him that he could only leave if he is able to solve the following problem standing. The problem is as follows:

Given an array of integers, find the sum of all $min(a)$ where $a$ is every possible contiguous subarray of the original array.

Since Twiggy cannot handle large numbers, help him find the answer modulo $10^9 + 7$.

## Input

The first line contains a single integer $n$ $(1 \leq n \leq 10^6)$ - the number of elements in the array.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$.

## Output

A single integer - sum of all $min(a)$.

---

```
input
12
776 181 356 708 184 699 409 603 669 119 430 926

output
18236
```

---

```
input
1
1

output
1
```

---

```
input
2
1 2

output
4
```

---

# D: Corporate Life is Boring Sometimes

After not getting IT Placements, you become a Data Analyst. Your Boss gives you 100's of Histograms a day. All of those Histograms have bars with a width of size 1 and no spaces in between. One day you record the heights in an array and decide to relive your coding days by trying to find out the maximum area of any possible rectangle inside of the Histogram.

## Input

The first line contains a single integer $n$ $(1 \leq n \leq 10^6)$ - number of bars in the Histogram.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^4)$ - heights of the bars.

## Output

A single integer - maximum area of all possible rectangles.

```
input
2
2 4

output
4
```

```
input
7
11 70 25 47 30 23 40

output
138
```

```
input
10
713 795 829 828 835 744 845 857 785 880

output
7130
```

# E: Bearish Kira

Kira loves to short stocks. One day he got a special offer in his dreams. He could sell 1 stock at spot price, remove $k$ digits from the spot price of the stock and buy the stock back at the new price. Find the total profit made by Kira.

**Note:** After each operation of removing a digit, all the leading zeroes should be removed as well, if any.

## Input

The first line contains two integers $n$ and $k$ ($1 \leq k \leq n \leq 10^6$ - number of digits of the spot price and the number of digits to remove respectively.
The second line contains a string $s$ of length $n$ having digits from 0-9 (The first digit is never 0).

## Output

A single integer giving the total profit of Kira.

```
input
7 3
1432219

output
1431000
```

```
input
18 3
107220268174334033

output
107200000000000000
```

```
input
6 4
345063

output
345060
```

# F: Peanut Butter And Nom Are Goals

Nom was on a Peanut Butter streak. So he decided to pre-order $n$ jars, 1 for each day. In order to make it more interesting, he spiced things up a bit. He asked the seller to send random weights. Now the seller sent him an array with all the jar weights.

Help Nom find another array where he notes down after how many days from the current day will he be able to have his beloved Peanut Butter whose weight is just greater than the current day's jar in the seller's array. If there is no such jar, assume 0 for that position in the array.

Similarly, Nom wants to find another array where he notes after how many days will he have a jar whose weight is just lesser than the current day's jar. In case he starts dieting of course. If there is no such jar, assume 0 for that position in the array.

## Input

The first line contains a single integer $n$ $(1 \leq n \leq 10^6)$ - the length of the seller's array.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^6)$ - weights of the Peanut Butter jars.

## Output

Two arrays of size $n$ in two different lines as described above.

```
input
4
30 40 50 60

output
1 1 1 0
0 0 0 0
```

```
input
7
12 192 102 128 294 293 239

output
1 3 1 1 0 0 0
0 1 0 0 1 1 0
```

# G: Trivially Easy Problem

Orakash is a very smart guy. Kira had very high expectations of Orakash, but Orakash shifted domains and Kira felt sad. Now Orakash wanted to start his comeback arc. So Kira felt happy and gave him a trivially easy problem to get him started. The problem is as follows:

Given an array of size $n$, find the number of all (non-empty) contiguous subarrays such that no two elements in the subarray have an absolute difference more than $k$.

Help Orakash in his comeback to become a Master-Candidate.

## Input

The first line contains two integers $n$ $k$ $(1 \leq n \leq 10^6, 1 \leq k \leq 10^9)$, as mentioned in the problem.. The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$.

## Output

A single integer - the number of (non-empty) subarrays which satisfy the given condition.

```
input
4 2
5 4 2 4

output
8

explanation
subarrays of size 1:  [5], [4], [2], [4].
subarrays of size 2:  [5,4], [4,2], [2,4].
subarrays of size 3:  [4,2,4].
There are no subarrys of size 4.
Total continuous subarrays = 4 + 3 + 1 = 8.
```

```
input
5 3
1 2 3 4 5

output
14
```

# H: SI COPE

Soomitra is the best developer. He was starting his SI cope arc. He felt the DSA labsheets are too easy for his level and approached Kira. Kira got very angry and told him to implement Min Stack and Avg Queue. Soomitra being too pro, wanted to implement both of them in the same code. Kira saw this and told him to use Heaps to implement these Datastructures. Will Soomitra be able to overcome this challenge.

Formally, implement a FILO stack which can push/pop/getmin in $O(1)$, and also implement a FIFO queue which can push/pop/getavg in $O(1)$ (min implies min of all elements currently in stack, similarly avg implies avg of all elements currently in queue). Also you need to use heaps to implement stack and queue (Give priority based on time of insertion).

## Input

The first line contains a single integer $q$ ($1 \leq q \leq 10^6$) - the number of queries.
The following $q$ lines are of the following format:

- 1 $x$ : push $x$ into both the queue and the stack.
- 2 : pop from both the queue and the stack.
- 3 : print the minimum element of the stack and the average of all elements in the queue.

## Output

For each query of form 3, print min and avg on a new line. (avg rounded down to nearest integer)

```
input
11
1 3
1 7
1 6
1 4
3
2
1 1
3
2
2
3
```

```
output
3 5
1 4
3 2
```

```
explanation
Stack      Queue
3          3
3 7        3 7
3 7 6      3 7 6
3 7 6 4     3 7 6 4
min = 3     avg = 5
3 7 6       7 6 4
3 7 6 1     7 6 4 1
min = 1     avg = 4
3 7 6       6 4 1
3 7         4 1
min = 3     avg = 2
  Max         Min
  Heap        Heap
```

# I: Select Algorithm

Select algorithm is the Median of Medians algorithm you learnt in class. It is used to find the $k$-th largest element in an array/list (unsorted) in $O(n)$ time complexity. Select is a very useful algorithm in the sense that it is used in a lot of proofs. But its $O(n)$ time complexity comes with a lot of constant factors so its not so good in practice.

Given an array of size $n$, we want you to implement Select Algorithm to find $k$-th largest number (No other algorithm will be tolerated).

## Input

The first line contains two integers $n$ and $k$ ($1 \leq k \leq n \leq 10^6$), as mentioned in the problem.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), the elements of the array.

## Output

The only line of output contains one integer - the $k$-th largest element. (Assume 1-based indexing)

```
input
6 2
3 2 1 5 6 4

output
5
```

```
input
9 4
3 2 3 1 2 4 5 5 6

output
4
```

```
input
11 5
1 2 3 4 5 6 7 8 9 10 11

output
7
```

# J: Quickselect Algorithm

Quickselect algorithm (also called RANDOMIZED-SELECT) is a spin off of select and quick sort. It is used to find the $k$-th largest element in an array/list (unsorted) in average case $O(n)$ time complexity. In Quickselect a random pivot (like in randomized quick sort) is chosen rather than MoM as in select. Quickselect is better than select in practice since its constant factors are smaller.

Given an array of size $n$, we want you to implement Quickselect Algorithm to find $k$-th largest number (No other algorithm will be tolerated).

## Input

The first line contains two integers $n$ and $k$ ($1 \leq k \leq n \leq 10^6$), as mentioned in the problem.
The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), the elements of the array.

## Output

The only line of output contains one integer - the $k$-th largest element. (Assume 1-based indexing)

---

```
input
6 2
3 2 1 5 6 4

output
5
```

---

```
input
9 4
3 2 3 1 2 4 5 5 6

output
4
```

---

```
input
11 5
1 2 3 4 5 6 7 8 9 10 11

output
7
```

---