Assalammualaikum

Mohammad Hanif Furqan Aufa Putra

5025221161

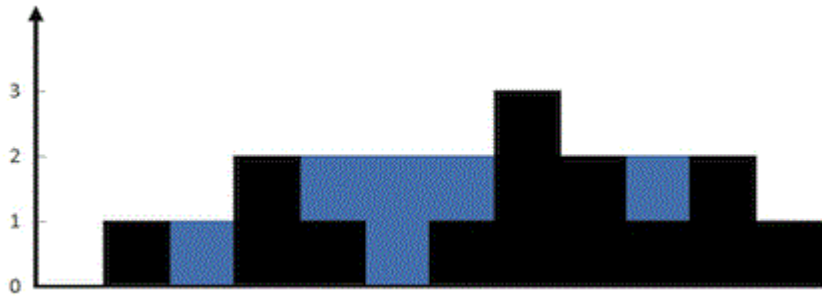
PERBAIKAN TUGAS 2 PBO C

Trapping Rain Water

Time limit 1 second

Memory limit 128 MiB

Given nn non-negative integers representing an elevation map where the width of each bar is 11.
Compute how much water it can trap after raining.



Input data

The first line contains the value of n (n≤105)n (n≤105).

The second line contains nn non-negative integers h1,h2,...,hn (hi≤105)h1,h2,...,hn (hi≤105).

Output data

Print how much water can be trapped after raining.

Examples

Input example #1

12
0 1 0 2 1 0 1 3 2 1 2 1
Output example #1

6
Input example #2

6
4 2 0 3 2 5
Output example #2

9

Soal ini meminta Anda untuk menghitung berapa banyak air yang dapat ditangkap oleh suatu konfigurasi topografi yang diwakili oleh deret angka. Setiap angka dalam deret tersebut mewakili tinggi suatu bar dalam peta elevasi. Air dapat tertangkap di antara dua batang jika ada bar yang lebih rendah di antara keduanya.

Contoh:

Jika Anda memiliki deret angka: `[0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]`, maka peta elevasi yang diwakili adalah sebagai berikut:

```
|                   |
|         #         |
|     #   ## #     |
|  #  ##  ######   |
```

Di sini, simbol '#' mewakili tinggi batang sesuai dengan nilai dalam deret angka.

Dengan kode ini, masalah ini dipecahkan dengan menggunakan pendekatan tumpukan (stack) untuk menghitung air yang terperangkap. Berikut adalah langkah-langkah utama dalam kode tersebut:

1. Input: Program membaca masukan, yaitu jumlah bar (n) dan tinggi setiap bar (h1, h2, ..., hn).

2. Inisialisasi variabel stack (st) untuk menyimpan indeks bar dalam bentuk stack dan variabel water untuk menghitung air yang terperangkap.

3. Iterasi melalui setiap bar dalam deret angka:

   - Saat mengiterasi, program membandingkan tinggi bar saat ini (height[i]) dengan tinggi bar di atas tumpukan (stack.top()).

   - Jika tinggi bar saat ini lebih tinggi dari tinggi bar di atas tumpukan, berarti ada potensi air yang terperangkap.

   - Program kemudian menghitung jarak antara dua bar (distance) dan tinggi air yang terperangkap (boundedHeight) di antara keduanya.

   - Jumlah air yang terperangkap (water) ditambahkan dengan hasil perkalian distance dan boundedHeight.

4. Jika tinggi bar saat ini tidak lebih tinggi dari tinggi bar di atas tumpukan, maka indeks bar saat ini dimasukkan ke dalam tumpukan (stack) untuk kemungkinan penggunaan di masa depan.

5. Setelah selesai iterasi melalui semua bar, jumlah total air yang terperangkap dihitung.

6. Hasilnya dicetak ke layar.

Pendekatan ini bekerja dengan cara menghitung air yang terperangkap antara setiap pasangan batang yang lebih tinggi dari batang di antara keduanya. Dengan demikian, kita dapat menghitung total air yang terperangkap di seluruh peta elevasi.

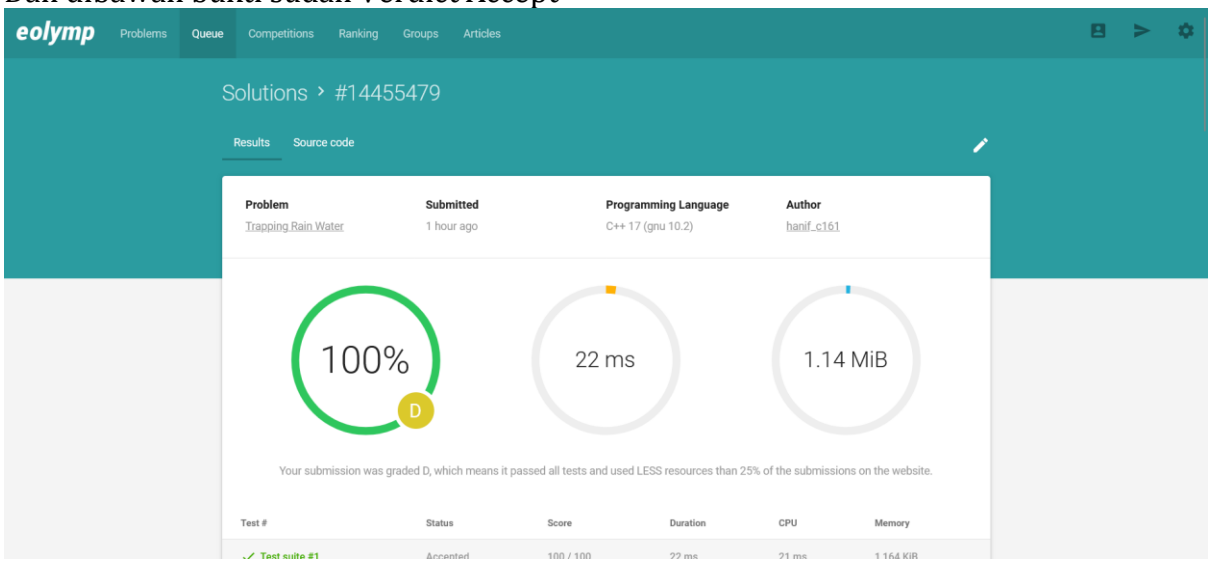
Dan dibawah ini merupakan keseluruhan code

```cpp
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

int trapRainWater(vector<int> &height)
{
    int n = height.size();
    if (n <= 2)
        return 0; // Tidak dapat menjebak air dengan kurang dari 3 bar

    stack<int> st; // Membuat tumpukan (stack) untuk menyimpan indeks bar
    int water = 0; // Variabel untuk menghitung air yang terperangkap

    for (int i = 0; i < n; ++i)
    {
        while (!st.empty() && height[i] > height[st.top()])
        {
            int topIdx = st.top();
            st.pop();

            if (st.empty())
                break; // Tidak ada batas kiri

            int distance = i - st.top() - 1; // Menghitung jarak antara dua bar
            int boundedHeight = min(height[i], height[st.top()]) - height[topIdx];
 // Menghitung tinggi air yang terperangkap
            water += distance * boundedHeight;
 // Menambahkan air yang terperangkap ke total air yang terperangkap
        }
        st.push(i); // Memasukkan indeks bar ke dalam stack
    }

    return water; // Mengembalikan jumlah total air yang terperangkap
}

int main()
{
    int n;
    cin >> n; // Membaca jumlah bar
    vector<int> elevationMap; // Vektor untuk menyimpan tinggi bar-bar

    for (int i = 0; i < n; ++i)
    {
        int height;
        cin >> height; // Membaca tinggi setiap bar dari input
        elevationMap.push_back(height);
    }

    int trappedWater = trapRainWater(elevationMap); // Menghitung air yang terperangkap
    cout << trappedWater << endl; // Menampilkan hasil jumlah air yang terperangkap ke layar

    return 0;
}

```

Dengan menjalan program tersebut diperoleh hasil seperti dibawah ini

```
                    > cd "f:\Coding\PBO-2023\Kuis\" ; if ($?) { g++ Trapping_Rain_Water.cpp -o Trapping_Rain_Water } ; if ($?) { .\Trapping_Rain_Water }
12
0 1 0 2 1 0 1 3 2 1 2 1
6
PS F:\Coding\PBO-2023\Kuis>
                    > cd "f:\Coding\PBO-2023\Kuis\" ; if ($?) { g++ Trapping_Rain_Water.cpp -o Trapping_Rain_Water } ; if ($?) { .\Trapping_Rain_Water }
6
4 2 0 3 2 5
9
PS F:\Coding\PBO-2023\Kuis>
```

Dan dibawah bukti sudah Verdict Accept



Arrange

Time limit **1** second

Memory limit **128** MiB

There are mm students in a classroom, out of which nn of them are girls. The teacher, Mr X wants to arrange all the students in a line. X thinks that his class girls are very talkative. So he doesn't want any two girls to be together. Mr X wants to know the number of ways he can arrange these mm students. Help him out.

Input data

The first line contains a single integer t (1≤t≤105)t (1≤t≤105), denoting the number of testcases. Each of the next tt lines contain two integers mm and n (1<n≤m≤106)n (1<n≤m≤106).

Output data

For each test case print the answer modulo 109+7109+7 in a single line.

Examples

Input example #1

```
2
3 2
4 2
```

Output example #1

```
2
12
```

Pada masalah ini, kita perlu mencari berapa banyak susunan barisan yang dapat dibuat untuk sekelompok siswa. Terdapat beberapa siswa perempuan dalam kelompok tersebut, dan aturan yang harus diikuti adalah tidak ada dua siswa perempuan yang boleh berdampingan dalam barisan. Soal ini akan memberikan beberapa testcase, masing-masing dengan jumlah total siswa dan jumlah siswa perempuan dalam kelompok tersebut. Tugas kita adalah menemukan berapa banyak susunan yang memenuhi aturan ini, dan mengembalikan hasil dalam bentuk modulo 10^9 + 7.

Pendekatan yang digunakan dalam kode adalah dengan memanfaatkan konsep faktorial dan permutasi. Pertama, faktorial dari 0 hingga 106 dihitung dan disimpan dalam array. Ini diperlukan untuk menghitung banyaknya susunan barisan. Selanjutnya, setiap testcase diambil, dan jika jumlah total siswa lebih dari atau sama dengan 2 * jumlah siswa perempuan - 1, maka perhitungan dimulai.

Untuk menghitung jumlah susunan yang memenuhi aturan, kita menggunakan rumus kombinasi nCr. Ini melibatkan menghitung faktorial dari jumlah siswa perempuan (untuk mewakili banyaknya susunan siswa perempuan) dan faktorial dari selisih antara jumlah total siswa dan jumlah siswa perempuan (untuk mewakili banyaknya susunan siswa laki-laki). Selanjutnya, rumus kombinasi nCr digunakan dengan bantuan invers modular untuk menghindari overflow. Ini dijelaskan dalam kode dengan implementasi fungsi inv dan powmod.

Contoh pengerjaan untuk sampel input:

Input:

2

3 2

4 2

Langkah-langkah:

1. Untuk testcase pertama (3 siswa, 2 siswa perempuan), kita memiliki 1 siswa laki-laki. Karena tidak ada dua siswa perempuan yang boleh berdampingan, satu susunan mungkin adalah LPL (L: laki-laki, P: perempuan). Jadi, hasilnya adalah 2 susunan yang memungkinkan.

2. Untuk testcase kedua (4 siswa, 2 siswa perempuan), kita memiliki 2 siswa laki-laki. Kita dapat membuat susunan seperti LPPL atau LPLP. Jadi, ada 2 susunan yang memungkinkan.

Kode ini menggunakan konsep faktorial, permutasi, dan rumus kombinasi dengan invers modular untuk menghitung jumlah susunan yang memenuhi aturan yang diberikan. Dengan pendekatan ini, kode dapat menyelesaikan masalah ini dengan efisien, bahkan untuk kasus dengan jumlah siswa yang besar, seperti yang ditunjukkan dalam sample input.

Dan dibawah ini merupakan keseluruhan code

```cpp
#include <stdio.h>
#include <algorithm>
#include <vector>
#define ll long long

using namespace std;
const int MOD = 1e9 + 7;


ll fast_pow(ll base, ll n, ll M)
{
    if (n == 0)
        return 1;
    if (n == 1)
        return base;
    long long halfn = fast_pow(base, n / 2, M);
    if (n % 2 == 0)
        return (halfn * halfn) % M;
    else
        return (((halfn * halfn) % M) * base) % M;
}

ll findMMI_fermat(ll n, ll M)
{
    return fast_pow(n, M - 2, M);
}

vector<ll> factorial(1000009);
int main(int argc, char const *argv[])
{
```

```
    factorial[0] = 1;
    for (ll i = 1; i < 1000009; i++)
    {
        factorial[i] = (factorial[i - 1] * i) % MOD;
    }

    int t;
    scanf("%d", &t);

    ll m, n;
    for (int i = 0; i < t; i++)
    {
        scanf("%lld %lld", &m, &n);

        ll man = m - n;

        if (man + 1 < n)
        {
            printf("0\n");
            continue;
        }

        ll first = factorial[man];

        ll y = findMMI_fermat(factorial[man + 1 - n], MOD);
        y = (factorial[man + 1] * y) % MOD;

        printf("%lld\n", (first * y) % MOD);
    }
    return 0;
}
```

Dengan menjalan program tersebut diperoleh hasil seperti dibawah ini

Success #stdin #stdout 0.01s 10212KB

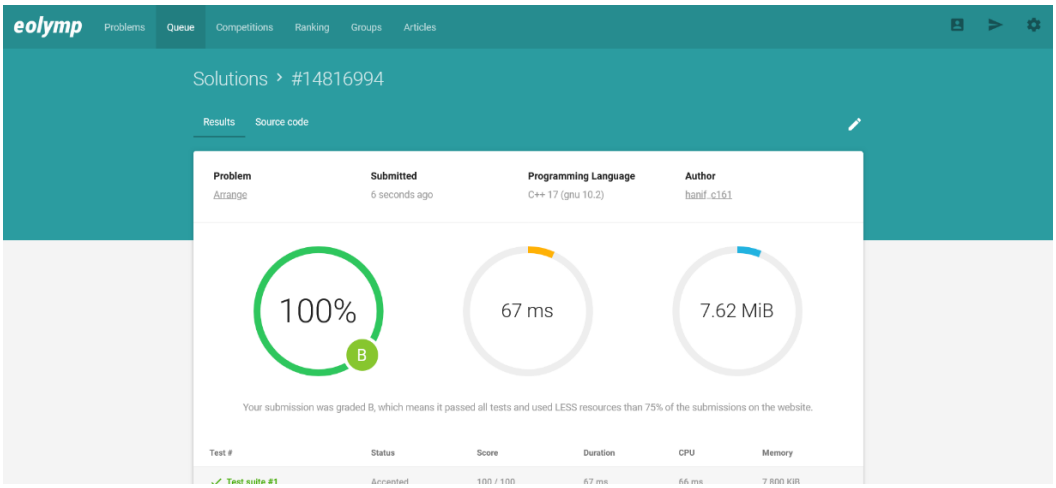
stdin

```
2
3 2
4 2
```

stdout

```
2
12
```

Dan dibawah bukti sudah Verdict Accept



LOTTERY- Tickets lottery

Byteland organizes this year's Soccer World Cup. Because of the mismanagement of the organizing team almost all tickets are sold out. But one radio station still has some tickets which will be raffled. Specifically, the radio station has announced a telephone game, where participants can choose a number between 1 and 1.000.000.000, and after each day the person who has chosen the k^{th} smallest number will win one ticket. A special rule is in place which disallows people to choose a number which was already chosen by someone else (in this case the person is asked to choose another number).

Martin, a fanatic soccer fan without tickets, bribed Robert H., an employee of the radio station, by promising small gifts for telling him a current winning number: "A fine basket with specialities from the black forest, including some really good sausages, ham and - hold on to your seat - a wonderful KuKuClock! And a beer mug, too! Do I leave you any choice???"

Now Robert is in trouble and asks you if you can write a program which will tell him the k^{th} smallest number at any time of the game.

Input

The first line of the input consists of the number of test cases to follow. Each test case starts with a line containing the number of telephone calls **c** ($1 \le c \le 500000$) followed by the number **k** ($1 \le k \le \min(c,100000)$). The following **c** lines specify the chosen numbers in chronological order of the phone calls. You can assume that all chosen numbers are unique, except the number 0 which indicates that Martin called and asked for the current k^{th} smallest number.

Output

For each line in the input with a zero print a line with the current k^{th} smallest number (or -1 if there are currently less than k chosen numbers).

Example

Input:

```
2
2 1
1337
0
7 2
4711
0
4
0
210706
3
0
```

Output:
1337
-1
4711
4

Langkah-langkah untuk "LOTTERY":

1. Observasi:

- Soal ini menggambarkan sebuah kasus di mana sebuah stasiun radio memiliki tiket yang akan diberikan dalam permainan telepon. Setiap peserta dapat memilih nomor dari 1 hingga 1.000.000.000, dan setiap hari orang yang memilih nomor ke-k akan memenangkan satu tiket. Nomor yang dipilih harus unik, kecuali nomor 0 yang digunakan oleh Martin untuk menanyakan nomor ke-k saat ini.

2. Pemilihan Class STL:

- Dalam kode ini, digunakan struktur data `priority_queue` dari C++ STL untuk menyimpan nomor-nomor yang dipilih dan menjaga agar selalu dalam urutan menurun (dari yang terbesar ke yang terkecil). Hal ini memungkinkan kita untuk dengan cepat mengambil nomor ke-k terkecil.

3. Abstract Data Type (ADT):

- Dalam konteks ini, ADT yang paling utama adalah `priority_queue`, yang digunakan untuk menyimpan nomor-nomor yang telah dipilih dalam urutan menurun.

4. Cara Kode Bekerja untuk Menyelesaikan Masalah:

- Dengan Fungsi `getNum()` membaca angka dari input, mengabaikan spasi, dan menghandle angka negatif jika ada, lalu mengembalikan angka yang dibaca.

- Kode ini memulai dengan membaca jumlah kasus uji (test case) yang akan dijalankan.

- Untuk setiap kasus uji, kode ini membaca jumlah panggilan telepon dan nomor ke-k.

- Selama setiap panggilan telepon, kode ini membaca nomor yang dipilih.

- Jika nomor adalah 0, maka kode ini memeriksa apakah ada cukup banyak nomor yang dipilih untuk mengambil nomor ke-k terkecil. Jika tidak, maka mengembalikan -1. Jika ya, maka kode ini mengambil nomor ke-k terkecil dari `priority_queue` dan mencetaknya.

- Jika nomor bukan 0, kode ini memeriksa apakah `priority_queue` sudah berisi kurang dari k nomor. Jika ya, maka nomor tersebut dimasukkan ke dalam `priority_queue`. Jika tidak, kode ini memeriksa apakah nomor yang baru dipilih lebih kecil dari nomor terbesar dalam `priority_queue`. Jika ya, maka nomor terbesar dihapus dari `priority_queue` dan nomor baru

dimasukkan. Jikat tidak maka nomor baru tidak akan dimasukkan dan lanjut ke nomor selanjutnya.

5. Kesimpulan:

- Kode ini adalah implementasi dari solusi untuk masalah "LOTTERY" yang menggunakan `priority_queue` untuk menjaga nomor-nomor yang telah dipilih dalam urutan menurun. Kode ini membaca input, memproses panggilan telepon, dan mengembalikan nomor ke-k terkecil ketika diminta.

Dan dibawah ini merupakan keseluruhan code

```cpp
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;

int getNum()
{
    int res = 0;
    char c;
    int b = 0;
    while (1)
    {
        c = getchar_unlocked();
        if (c == '-')
            b = 1;
        if (c == ' ' || c == '\n')
            continue;
        else
            break;
    }
    if (c != '-')
        res = c - '0';
    while (1)
    {
        c = getchar_unlocked();
        if (c >= '0' && c <= '9')
            res = 10 * res + c - '0';
        else
            break;
    }
    if (b == 1)
        res *= -1;
    return res;
}
```

```cpp

int main()
{
    int
testcase, phoneCalls, number, result, kth;
    testcase = getNum();
    for (int i = 0; i < testcase; i++)
    {
        priority_queue<int> numList;
        phoneCalls = getNum();
        kth = getNum();
        for (int j = 0; j < phoneCalls; j++
)
        {
            number = getNum();
            if (number == 0)
            {
                if (kth > numList.size())
                    result = -1;
                else
                    result = numList.top();
                printf("%d\n", result);
            }
            else
            {
                if (numList.size() < kth)
                    numList.push(number);
                else if (numList.size() ==
kth && number < numList.top())
                {
                    numList.pop();
                    numList.push(number);
                }
            }
        }
    }
    return 0;
}
```

Dengan menjalan program tersebut diperoleh hasil seperti dibawah ini

Success #stdin #stdout 0.01s 5536KB

stdin

```
2
2 1
1337
0
7 2
4711
0
4
0
210706
3
0
```

stdout

```
1337
-1
4711
4
```

Dan dibawah bukti sudah Verdict Accept

: submissions

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
31966150	☐	2023-10-06 05:30:18	Tickets lottery	**accepted** edit ideone it	0.05	5.4M	CPP14

Terima Kasih telah membaca, Waalaikumsalam wr wb.