COMP2119 Introduction to Data Structures and Algorithms
Programming Assignment 2

Due Date: **7pm, Dec 3, 2024**

This problem set contains 3 tasks.

Assignment Completion

- You must use `C++` to complete the assignment.
- We are unable to assist with debugging your source code due to resource limitations.
- All the source code should be written by you. Using others' code will result in plagiarism.

Judging

- Your program will be judged by a computer automatically. A `C++` compiler (of `ISO/IEC 14882` standard, i.e., `C++14`) will be used.
- There will be 10 test cases for each task.
  - Each test case is worth 3 marks for task A.
  - Each test case is worth 4 marks for task B.
  - Each test case is worth 3 marks for task C.
- For each test case, you will receive full marks if your program:
  - compiles successfully within a reasonable time;
  - reads from standard input;
  - outputs to standard output;
  - terminates within the time limit (1 second for each test case) without any errors occurring;
  - has the correct output (including the correct format as specified, with no additional or mising characters).

  If any of above fails, you will receive 0 marks for that test case.
- The output of your program will be judged after your program terminates. For problems that require one answer for each query, it is allowed that your program outputs answers after all the input are processed, as long as your output follows the specified format in the correct order.
- You may `#include` headers supported by the `C++` standard library, which includes `<cstdio>`, `<cmath>`, `<iostream>`, `<algorithm>`, `<vector>`, `<map>`, `<set>`, etc.

  Usage of any library apart from standard library will result in 0 marks. Please note that Policy-Based Data Structures (under `__gnu_pbds::`) are not part of the standard library.
- Please do not involve randomness and undefined / unspecified / implementation-defined behaviors in your program. Such behaviors include but are not limited to integer overflow, division by zero, and usage of uninitialized variables.

- If your program trys to modify the grading computer or intervene in the grading process, 0 marks will be given, and your case will be reported immediately. Such behaviors include but are not limited to opening a file, reading from a file (other than standard input), writing to a file (other than standard output), and usage of `fork()`, `exec()`, `system()`.
- Parallel Computing is not allowed.

Suggestions and Good Practice

- To read from standard input, you may use: `getchar()`, `scanf()`, `fgets()`, `fread()`, `std::cin`, `std::getline()`. (`getchar()` is suggested when dealing with a large number of integers input.)
- To write to standard output, you may use: `putchar()`, `printf()`, `fputs()`, `fwrite()`, `std::cout`. (`putchar()` is suggested when dealing with a large number of integers output.)
- **[Self Testing.]** Test your program by yourself using the provided sample input/output files. The sample input/output is **different** from the ones used to judge your program, and it is designed for you to test your program on your own.
  - To test your program on Linux (this is the preferred option) or Mac OS X:
    1. Put your source code "`main.cpp`" and sample input file "`input.txt`" in the same directory.
    2. Open a terminal and navigate to that directory.
    3. Compile your program using "`g++ -std=c++14 main.cpp -o main`".
    4. Run your program using the given input file by entering "`./main < input.txt > myoutput.txt`".
    5. Compare "`myoutput.txt`" with sample output file.
    6. Your output in "`myoutput.txt`" needs to be **exactly** the same as the sample output. Otherwise, it will be judged as incorrect.
    7. You may find the `diff` command useful to help you compare two files, like "`diff -w myoutput.txt sampleoutput.txt`". The `-w` option ignores all blanks (SPACE and TAB characters).
  - To test your program on Windows (it's fine but may be complicated to install the compiler):
    1. Compile your program and obtain the executable file, "`main.exe`".
    2. Place the sample input file, "`input.txt`", in the same directory as "`main.exe`".
    3. Open the command line and navigate to the directory that contains "`main.exe`" and "`input.txt`".
    4. Run "`main.exe < input.txt > myoutput.txt`".
    5. Compare "`myoutput.txt`" with the sample output file. You may find the "`fc.exe`" tool useful.
    6. Your output in "`myoutput.txt`" needs to be **exactly** the same as the sample output. Otherwise, it will be judged as incorrect.
  - If these two options do not work, you can use an online compiler as a last resort: `https://www.onlinegdb.com/online_c++_compiler`

Submission

- Please submit your source files via HKU moodle. You should submit three source files (without compression) only, one for each task.
- Please name your files in format `UniversityNumber-X.cpp` for `X`$\in$`{A,B,C}`, e.g., "`1234554321-A.cpp`".
- If your submissions are not named correctly, you will receive 0 marks for that task.

Task A [30%]

In the lecture on *Binary Search Tree*, you have learned the following pseudocode about inserting a node into a binary search tree:

```
Tree_insert(T, z){
    y = NULL;
    x = T;
    while(x != NULL){
        y = x;
        if(z->key < x->key) x = x->left;
        else x = x->right;
    }
    z->parent = y; z->left = z->right = NULL;
    if(y==NULL) T = z;
    else if(z->key < y->key) y->left = z;
    else y->right = z;
}
```

Given a binary search tree and a node x from the tree, we define

$$\mathrm{dh}_x = |\mathrm{height}(L_x) - \mathrm{height}(R_x)|,$$

where L_x and R_x are the left subtree and right subtree rooted at node x. Without loss of generality, we assume the height of a null tree is -1.

Let T be an empty binary search tree and $a_1, a_2, \ldots, a_n$ be n distinct integers; one can insert the integers into T by calling `Tree_insert` on $a_1, a_2, \ldots, a_n$ sequentially.

In this problem, you are required to find the maximum dh_x among all nodes x from T after all the insertions.

Input

The first line contains an integer n.

In the second line, there are n distinct integers $a_1, a_2, \ldots, a_n$ separated by a space.

Test Cases No.	Specifications	
1,2,3,4,5,6,7,8,9,10	$1 \le n \le 10^3$	$1 \le a_i \le n$ for $1 \le i \le n$; $a_i \ne a_j$ for $1 \le i, j \le n$ and $i \ne j$.

Output

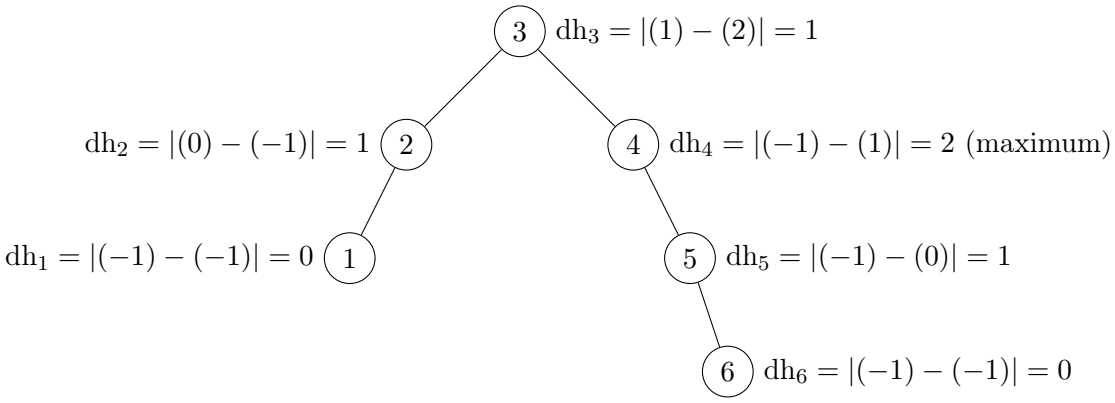
Output one integer representing the maximum dh_x in the binary search tree after inserting all the integers.

Examples

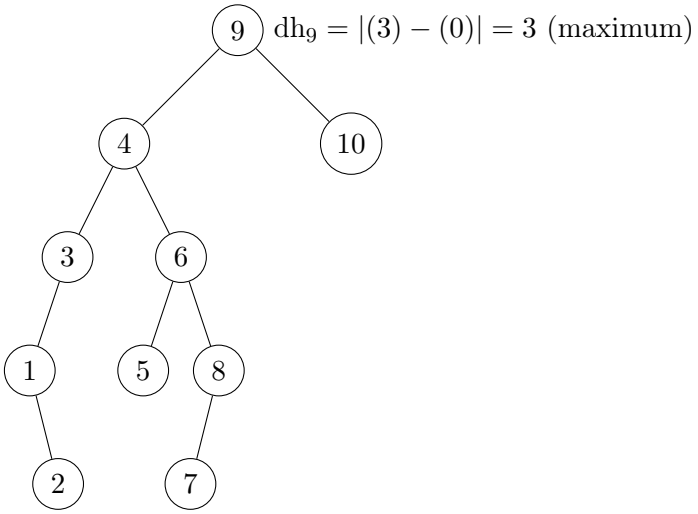
standard input	standard output
6 3 2 1 4 5 6	2
10 9 4 6 10 3 1 8 2 7 5	3
- See sampleA3.in -	- See sampleA3.ans -

Note

The binary search tree of sample 1 and calculation of dh for each node are as follows:



The binary search tree of sample 2 is as follows:



Sample 3 satisfies $n = 1,000$.

Submission

Your source code should be named `UniversityNumber-A.cpp`, where `UniversityNumber` should be replaced by your University Number, and be submitted via HKU Moodle.

For example, if your University Number is `1234554321`, your submission should be named "`1234554321-A.cpp`".

Hint

- A reasonable solution takes $O(n^2)$ time.

Task B [40%]

In this problem, you are required to maintain a collection S of integers (S is initially empty) and to support the following operations on S:

1. `Insert`(x): insert an integer x into S;
2. `Remove`(x): remove the integer x from S;
3. `FindKth`(k): find the k-th smallest element in S;
4. `Index`(x): find the integer k such that x is the k-th smallest element in S;
5. `Sum`(k): calculate the sum of k smallest elements.

Input

The first line contains an integer n, the number of operations. In the following n lines, each line starts with an integer `op` $\in \{1, 2, 3, 4, 5\}$ and it is followed by one integer in the same line separated by a space:

- if `op` $= 1$, it is followed by an integer x, which denotes the operation `Insert`(x);
- if `op` $= 2$, it is followed by an integer x, which denotes the operation `Remove`(x);
- if `op` $= 3$, it is followed by an integer k, which denotes the operation `FindKth`(k);
- if `op` $= 4$, it is followed by an integer x, which denotes the operation `Index`(x);
- if `op` $= 5$, it is followed by an integer k, which denotes the operation `Sum`(k).

You may assume that after each operation, S has no duplicate elements.

You may assume that all the operations are valid:

- for `Remove`(x) and `Index`(x), x is in the current S;
- for `FindKth`(k) and `Sum`(k), $1 \leq k \leq l$, where l is the size of current S.

Test Cases No.	Specifications	
1,2,3,4,5,6,7	`op` $\in \{1, 2, 3, 4\}$	$1 \leq n \leq 10^5$; $1 \leq x \leq 10^{12}$.
8,9,10	`op` $\in \{1, 2, 3, 4, 5\}$	

Output

For each input line with `op` $\in \{3, 4, 5\}$, output one line that contains an integer denoting the answer.

Examples

standard input	standard output
17 1 2 1 4 1 6 1 8 3 3 4 6 2 2 3 3 4 6 5 2 2 6 5 2 2 4 2 8 1 999999999999 1 1000000000000 5 2	6 3 8 2 10 12 1999999999999
- See sampleB2.in -	- See sampleB2.ans -
- See sampleB3.in -	- See sampleB3.ans -

Note

Sample 2 satisfies $n = 100,000$ and `op` $\in \{1, 2, 3, 4\}$.

Sample 3 satisfies $n = 100,000$ and `op` $\in \{1, 2, 3, 4, 5\}$.

Submission

Your source code should be named `UniversityNumber-B.cpp`, where `UniversityNumber` should be replaced by your University Number, and be submitted via HKU Moodle.

For example, if your University Number is `1234554321`, your submission should be named "`1234554321-B.cpp`".

Hints

- A reasonable solution takes $O(n \log n)$ time.
- Use AVL Tree.
- Use `long long int` to store the element.
- Use a variable `sum` in a node to record the sum of all the elements in its subtree and maintain it using the same structure of maintaining `size` of each subtree. When the whole subtree is involved in answering the query of `Sum`(k), you can use this variable to save the time needed to enter the subtree.

Task C [30%]

In the lecture on *Data Structures*, you learnt about the *list ADT*:

> A *list* is an ordered set of elements. E.g., $\{a_1, a_2, a_3, \ldots, a_n\}$ is a list of size n.
> For the list:
> - a_{i+1} follows (or succeeds) a_i.
> - a_{i-1} precedes a_i.
> - The position of element a_i is i.

In this problem, you are required to maintain a list L of integers and to support the following operations on L:

1. `Insert`(k, x): insert an integer x into L at position k
   (e.g., if $L = \{10, 9, 8\}$, after `Insert`$(1, 5)$, $L = \{5, 10, 9, 8\}$);
2. `RemoveKth`(k): remove the integer at position k from L
   (e.g., if $L = \{5, 10, 9, 8\}$, after `RemoveKth`(4), $L = \{5, 10, 9\}$);
3. `FindKth`(k): find the element at position k in L
   (e.g., if $L = \{5, 10, 9\}$, `FindKth`(1) gives 5);
4. `Index`(x): find the position of x in L
   (e.g., if $L = \{5, 10, 9\}$, `Index`(10) gives 2);
5. `Sum`(k_1, k_2): calculate the sum of elements at position $k_1, k_1 + 1, \ldots, k_2$
   (e.g., if $L = \{5, 10, 9\}$, `Sum`$(2, 3)$ gives 19);
6. `MaxElement`(k_1, k_2): find the maximum element among elements at position $k_1, k_1 + 1, \ldots, k_2$
   (e.g., if $L = \{5, 10, 9\}$, `MaxElement`$(1, 3)$ gives 10).

Input

The first line contains two integers n, m, separated by a space, where n refers to the number of elements in L at the beginning and m is the number of operations.

The second line contains n integers, each separated by a space, which represents the initial list L.

In the following m lines, each line starts with an integer `op` $\in \{1, 2, 3, 4, 5, 6\}$ and it is followed by one or two integers in the same line each separated by a space:

- if `op` $= 1$, it is followed by two integers k, x, which denotes the operation `Insert`(k, x);
- if `op` $= 2$, it is followed by one integer k, which denotes the operation `RemoveKth`(k);
- if `op` $= 3$, it is followed by one integer k, which denotes the operation `FindKth`(k);
- if `op` $= 4$, it is followed by one integer x, which denotes the operation `Index`(x);
- if `op` $= 5$, it is followed by two integers k_1, k_2, which denotes the operation `Sum`(k_1, k_2);
- if `op` $= 6$, it is followed by two integers k_1, k_2, which denotes the operation `MaxElement`(k_1, k_2).

You may assume that for the initial L and after each operation, L has no duplicate elements.

You may assume that all the operations are valid:

- for `Insert`(k, x): $1 \le k \le l + 1$, where l is the number of elements in current L;
- for `RemoveKth`(k) and `FindKth`(k): $1 \le k \le l$, where l is the number of elements in current L;
- for `Index`(x): x is in the current L;
- for `Sum`(k_1, k_2) and `MaxElement`(k_1, k_2): $1 \le k_1 \le k_2 \le l$, where l is the number of elements in current L.

Test Cases No.	Specifications	
1,2,3,4,5,6,7	`op` $\in \{1, 2, 3, 4\}$	$1 \le n, m \le 10^5; 1 \le x \le 10^{12}$.
8,9,10	`op` $\in \{1, 2, 3, 4, 5, 6\}$	

Output

For each input line with `op` $\in \{3, 4, 5, 6\}$, output one line which contains an integer denoting the answer.

Examples

standard input	standard output
3 11 10 9 8 1 1 5 2 4 3 1 4 10 5 2 3 6 1 3 2 1 2 1 2 1 1 1 1000000000000 5 1 1	5 2 19 10 1000000000000
- See sampleC2.in -	- See sampleC2.ans -
- See sampleC3.in -	- See sampleC3.ans -

Note

Sample 2 satisfies $n = m = 100,000$ and `op` $\in \{1, 2, 3, 4\}$.

Sample 3 satisfies $n = m = 100,000$ and `op` $\in \{1, 2, 3, 4, 5, 6\}$.

Submission

Your source code should be named `UniversityNumber-C.cpp`, where `UniversityNumber` should be replaced by your University Number, and be submitted via HKU Moodle.

For example, if your University Number is `1234554321`, your submission should be named "`1234554321-C.cpp`".

Hints

- A reasonable solution takes $O((n+m)\log(n+m))$ time. The array or linked list implementation is not sufficient to pass the test cases.
- Maintain a binary tree whose inorder sequence is the same as L.
- Use `long long int` to store the element.
- When insertion happens, find the $(k-1)$-th node and k-th node first and insert the new node to be a child of one of them.
- You may need another efficient data structure (e.g., hash table, binary search tree) to determine the corresponding node of a certain key in the binary tree.
- Use a divide-and-conquer approach on the tree to answer query `Sum`(k_1, k_2) and `MaxElement`(k_1, k_2). Maintain the answer of each subtree beforehand to reduce the number of nodes you have to visit.

- This is the end of the problem set. -