Task 1. Namuhs

The humanoid aliens from the planet Htrae (they are called namuhs) are at the peak of their development and they can create living creatures. After a big journey around our known universe a special team has made a list of planets sorted by their distance from the Htrae. The team has given the list to "The high council" of the namuhs who analyzed this information in details and decided to assign of every planet an integer number to describe its potential for the experiment. The chief of "The high council" is **D**eni and she has to decide which planets will take part in the plan. Because the distances in space are very big even for this advanced race, it has to be chosen only one set of planets, which are adjacent in the list – i.e. they should form a segment. The total potential of one set of planets is equal to the sum of the potentials of every planet in it. "The high council" has decided that the chosen set has to be one with the highest possible total potential. **D**eni remembered that she knows one planet not far away – the planet Earth, where the civilization (of the so called humans) isn't so advanced but there are creatures which can help her to make a program for this problem. Namuhs don't want to give away this secret information so the only access to that data you have, is through asking questions about comparing the sums of two segments of planets.

Task

You have to implement a function *find_max* which will be compiled with the source file of the jury (of **D**eni) and has to return two numbers for the segment of adjacent planets which have the highest possible total potential. This function will receive one number ***N*** – the number of planets. The jury has the values of planets potentials in appropriate order. Your goal is through asking questions for comparing segments of adjacent planets to find the segment with the highest possible potential. It is guaranteed that there is only one answer!

Details for the implementation

The function *find_max* has this prototype:

```
void find_max (int N, int& left, int& right);
```

It is called once by the program of the jury with argument *N* – the number of the planets in the list. When your program finds the answer, you have to save it in the parameters *left* and *right* – the left and right end of the segment respectively (the numbering starts from 1).

For communication with the program of the jury you have this function:

```
bool compare_segments (int left1, int right1, int left2, int right2);
```

It asks a question whether the total potential of the planets with positions from *left1* to *right1* (inclusive) is higher than or equal to the total potential of the planets with positions from *left2* to *right2* (inclusive). Here the following conditions have to be met: $1 \le left1 \le right1 \le N$ and $1 \le left2 \le right2 \le N$. Note, that the function *compare_segments* works in linear complexity of the input data i.e. it needs (*right1*-*left1*+1)+(*right2*-*left2*+1) iterations to find the answer!

You have to submit to the system only the file **namuhs.cpp**, which contains the function *find_max*. This file can contain also other lines of code and functions, needed for the work of *find_max*, but it shouldn't have a main function - *main*. At the beginning of your file you have to write: **#include "namuhs.h"**.

Constraints

$2 \le N \le 10^5$

In 10% of tests: $N \le 10^2$
In other 30% of tests: $N \le 10^3$

Sample communication with the program of the jury

Let the list with the potential of the planets is: 2, -3, 5, -2, 3. Here the answer is the segment with planets from position 3 to position 5 inclusive. The total number of planets is 5 so the program of the jury will call your function *find_max* as follows:

```
find_max(5,left,right);
```

Sample communication for finding the answer can be the following:

Calling function from the jury program	**Return result**	**Explanation**
compare_segments(3,5,1,1)	true	Your program asks whether the total potential of the planets in the segment from 3 to 5 is higher than or equal to the total potential of the planets from position 1 to position 1, i.e. if (5+(-2)+3) ≥ 2, which is true and the function returns true.
compare_segments(1,1,3,5)	false	This means that the total potential of the planets in the segment from 3 to 5 is strictly higher than the potential from 1 to 1.
compare_segments(3,5,2,2)	true	Here are needed 4 iterations for the function.
compare_segments(3,5,3,3)	true	Here are needed 4 iterations for the function.
compare_segments(3,5,4,4)	true	Here are needed 4 iterations for the function.
compare_segments(3,5,5,5)	true	Here are needed 4 iterations for the function.
compare_segments(3,5,1,2)	true	Here are needed 5 iterations for the function.
compare_segments(3,5,2,3)	true	Here are needed 5 iterations for the function.
compare_segments(3,5,3,4)	true	Here are needed 5 iterations for the function.
compare_segments(3,5,4,5)	true	Here are needed 5 iterations for the function.
compare_segments(3,5,1,3)	true	Here are needed 6 iterations for the function.
compare_segments(3,5,2,4)	true	Here are needed 6 iterations for the function.
compare_segments(3,5,1,4)	true	Here are needed 7 iterations for the function.
compare_segments(3,5,2,5)	true	Here are needed 7 iterations for the function.
compare_segments(3,5,1,5)	true	We have already checked that the total potential of the planets in the segment from 3 to 5 is higher than or equal to the potential of each of the other segments. So this segment gives us the optimal answer. Now the program set the parameters of the function *find_max* the values: *left*=3 and *right*=5 and the function has to stop its work.

Local testing

You are given the files **Lgrader.cpp** and **namuhs.h** for local testing. You have to place them in the same folder as your program **namuhs.cpp** and you have to compile the file **Lgrader.cpp**. In such a way, you will make a program to check the correctness of your function. This program will require the following data from the standard input:

- on the first line: one positive number – the number of the planets in the list.
- on the second line: the potentials of the corresponding planets.

The output of the program will be the answer that your function found.

Submitting user tests to the system

You can give tests to the system. The format of the input data has to be the same as in the local grader. The output you will receive is a number greater than 0, if the answer of your program is correct, and 0, if the output of your program is not correct.