Task 1. Namuhs

Հթրայե մոլորակից հումանոիդ այլմոլորակայինները (նրանց անվանում են նամուհս) իրենց զարգացման գագաթնակետում են և նրանք կարող են ստեղծել կենդանի էակներ: Մեր հայտնի տիեզերքում երկար ճամփորդությունից հետո հատուկ թիմը պատրաստել է մոլորակների ցուցակ սորտավորված ըստ Հթրայեյից հեռավորության: Թիմը ցուցակը տվել է նամուհսների «Բարձր կոնսուլին», ով վերլուծեց այս ինֆորմացիան մանրամասնորեն և որոշեց յուրաքանչյուր մոլորակի վերագրել մեկ ամբողջ թիվ, որը նկարագրում է նրա պոտենցիալը փորձի համար: «Բարձր կոնսուլի» ղեկավարը **Դ**ենին է և նա պետք է որոշի, թե որ մոլորակները պետք է մասնակցեն իր պլանում: Քանի որ հեռավորությունները տիեզերքում շատ մեծ են, պետք է ընտրել միայն մոլորակների մեկ բազմություն, որոնք ցուցակում իրար կից են, այսինքն նրանք կազմում են հատված: Մոլորակների բազմության ընդհանուր պոտենցիալը հավասար է նրանց պոտենցիալների գումարին: «Բարձր կոնսուլը» որ ընտրված բազմությունը պետք է ունենա ամենամեծ հնարավոր պոտենցիալը: **Դ**ենին հիշեց, որ նա գիտի մի մոլորակ ոչ այնքան հեռու՝ Երկիր մոլորակը, որի քաղաքակրթությունը (այսպես կոչված մարդիկ) այդքան էլ առաջադեմ չի, բայց կան արարածներ, ովքեր կարող են օգնել իրեն այս խնդրի համար ծրագիր գրել: Նամուհսը չի ցանկանում այս գաղտնի ինֆորմացիան տարածել, այնպես որ այդ տվյալները ստանալու միակ եղանակը մոլորակների երկու հատվածների գումարների համեմատության հարցումների միջոցով է:

Խնդիրը

Դուք պետք է իրականացնեք *find_max* ֆունկցիան, որը կկոմպիլացվի ժյուրիի (**Դ**ենիի) source ֆայլի հետ և պետք է վերադարձնի երկու թիվ՝ կից մոլորակների հատվածի վերաբերյալ, որոնց ընհդանուր պոտենցիալը հնարավոր մեծագույնն է: Այս ֆունկցիան ստանալու է մեկ թիվ՝ ***N*** մոլորակների քանակը: Ժյուրին ունի մոլորակների պոտենցիալների արժեքները: Ձեր նպատակն է կից մոլորակների հատվածների պոտենցիալների համեմատության հարցերի միջոցով գտնել մեծագույն հնարավոր պոտենցիալով հատվածը: Երաշխավորվում է, որ գոյություն ունի միայն մեկ պատասխան:

Իրականացման մանրամասներ

find_max ֆունկցիայի նախատիպը այսպիսին է.

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

Այն կանչվում է միայն մեկ անգամ ժյուրիի ծրագրի կողմից: *N* – ը ցուցակում մոլորակների քանակն է: Երբ ձեր ծրագիրը գտնի պատասխանը, դուք պետք է այն պահեք *left* և *right* պարամետրերում – հատվածի ձախ և աջ եզրերը համապատասխանաբար (համարակալումը սկսած 1-ից):

Ժյուրիի ծրագրի հետ հաղորդակցվելու համար դուք ունեք այս ֆունկցիան.

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

Այն հարցնում է արդյոք It asks a question whether the total potential of the planets with positions from *left1*-ից *right1* (ներառյալ) դիրքերով մոլորակների ընդհանուր պոտենցիալը ավելի մեծ է *left2*-ից *right2* (ներառյալ) դիրքերով մոլորակների ընդհանուր պոտենցիալից: Այստեղ հետևյալ պայմանները տեղի ունեն. $1 \le left1 \le right1 \le N$ և $1 \le left2 \le right2 \le N$: Նկատենք, որ *compare_segments* ֆունկցիայի աշխատանքի բարդությունը գծային է կախված մուտքային տվյալներից: Նրան պետք է (*right1*-*left1*+1)+(*right2*-*left2*+1) իտերացիաներ պատասխանը գտնելու համար:

Դուք պետք է submit անեք միայն **namuhs.cpp** ֆայլը, որում պետք է լինի *find_max ֆունկցիան*. Այս ֆայլում կարող են այլ անհրաժեշտ ֆունկցիաներ ևս լինել, բայց այնտեղ *main* ֆունկցիա չպիտի լինի: Ֆայլի սկզբում պետք է գրված լինի **#include "namuhs.h"**.

Constraints

$2 \le \boldsymbol{N} \le 10^5$

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

Ժյուրիի ծրագրի հետ հաղորդակցման օրինակ

Ենթարդենք մոլորակների պոտենցիալներն են 2, -3, 5, -2, 3: Այստեղ պատասխանը 3 դիրքից 5-րդ դիրքը ներառյալ մոլորակներով հատվածն է: Մոլորակների ընդհանուր քանակը 5 է: Ժյուրիի ծրագիրը ձեր ֆունկցիայի կանչը կանի հետևյալ կերպ.

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

Պատասխանը գտնելու համար հաղորդակցությունը կարող է լինել այսպիսին.

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	true	Here are needed 7 iterations for the function.

,2,5)		
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.