Task 1. Namuhs

Alienii umanoizi de pe planeta Htrae (numiți namuhs) sunt la vârf in dezvoltarea lor și pot crea ființe vii. După o lungă călătorie prin univers, o echipa specială a facut o listă a planetelor sortate după distanța de la Htrae. Echipa a dat lista "înaltului consiliu" al namuhilor, care a analizat detaliat aceste informații și a decis să atribuie fiecărei planete un număr întreg pentru a descrie potențialul acestora pentru experimente. Șeful "Înaltului Consiliu" este Deni și ea a decis care planete vor lua parte la plan. Deoarece distanțele în spațiu rămân foarte mari chiar și pentru această rasă avansată, trebuie ales doar un set de planete, care sunt adiacente în listă – de exemplu pot forma un segment. Potențialul total al unui set de planete este egal cu suma potențialelor tuturor planetelor din set.

"Înaltul Consiliu" a decis că setul selectat trebuie să fie unul cu cel mai înalt potențial total. **D**eni și-a amintit că ea cunoaște o planetă nu tare îndepărtată –planeta Pământ, unde civilizația (așa numiții oameni) nu este atât de avansată, dar sunt creaturi care o pot ajuta să realizeze un program pentru problema ei. Deni nu vrea să divulge informațiile secrete colectate, astfel singurul tău mod de acces la date, se va realiza prin întrebări despre compararea sumelor din două segmente de planete.

Task

Trebuie să implementezi o funcție *find_max* care va fi compilată cu fișierul sursă al comisiei (**D**eni) și va returna două numere pentru segmentul de planete adiacente care are cel mai înalt potențial posibil. Această funcție va primi un număr ***N*** – numărul de planete. Consiliul are valorile potențialelor planetelor în ordinea corespunzătoare. Misiunea ta este ca, punând întrebări despre compararea segmentelor de planete adiacente să determini segmentul cu cel mai înalt potențial posibil. Se garantează că răspunsul este unic!

Detailii de implementare

Funcția *find_max* are următorul prototip:

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

Funcția este apelată de programul juriului a singură dată cu argumentul *N* – numărul planetelor în listă. Atunci când programul tău găsește răspunsul, acesta trebuie salvat în parametrii *left* și *right* – extremitățile stângă și dreaptă ale segmentului respectiv (numerotarea începe de la 1).

Pentru comunicarea cu programul juriului vei avea funcția:

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

Ea pune o întrebare dacă potențialul total al planetelor cu pozițiile de la *left1* la *right1* (inclusiv) este mai mare sau egală decât potențialul total al planetelor cu pozițiile de la *left2* la *right2* (inclusiv). Următoarele condiții trebuie să fie respectate: $1 \le left1 \le right1 \le N$ și $1 \le left2 \le right2 \le N$. De remarcat că funcția *compare_segments* rulează în complexitate liniară pe datele de intrare adică sunt necesare $(right1-left1+1)+(right2-left2+1)$ iterații pentru a găăsi răspunsul!

Vei trimite în sistem doar fișierul **namuhs.cpp**, care conține funcția *find_max*. Acest fișier poate conține și alte linii de cod și funcții, necesare funcționării funcției *find_max*, dar nu va conține o funcție main - *main*. La începutul fișierului trebuie să scrii: **#include "namuhs.h"**.

Restricții

$2 \le N \le 10^5$

În 10% din teste: $N \le 10^2$
În alte 30% din teste: $N \le 10^3$

Exemplu de comunicare cu programul juriului

Fie lista cu potențialele planetelor este: 2, -3, 5, -2, 3. Aici răspunsul este segmentul cu planetele de la poziția 3 până la poziția 5 inclusiv. Numărul total al planetelor este 5, astfel programul juriului va apela funcția *find_max* după cum urmează:

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

Exemplul de comunicare pentru găsirea răspunsului poate fi următorul:

Apelarea funcției din programul juriului	**Rezultatul returnat**	**Explicație**
compare_segments(3,5,1,1)	True	Programul tău întrebă dacă potențialul total al planetelor din segmentul de la poziția 3 la poziția 5 este mai mare sau egal decât potențialul total al planetelor de la poziția 1 la poziția 1, adică dacă (5+(-2)+3) ≥ 2, ceea ce este adevărat și funcția va returna true.
compare_segments(1,1,3,5)	False	Aceasta înseamnă că potențialul total al planetelor din segmentul de la poziția 3 până la poziția 5 este strict mai mare decât potențialul de la 1 la 1.
compare_segments(3,5,2,2)	True	Aici sunt necesare 4 iterații pentru funcție.
compare_segments(3,5,3,3)	True	Aici sunt necesare 4 iterații pentru funcție.
compare_segments(3,5,4,4)	True	Aici sunt necesare 4 iterații pentru funcție.
compare_segments(3,5,5,5)	True	Aici sunt necesare 4 iterații pentru funcție.
compare_segments(3,5,1,2)	True	Aici sunt necesare 5 iterații pentru funcție.
compare_segments(3,5,2,3)	True	Aici sunt necesare 5 iterații pentru funcție.
compare_segments(3,5,3,4)	True	Aici sunt necesare 5 iterații pentru funcție.
compare_segments(3,5,4,5)	True	Aici sunt necesare 5 iterații pentru funcție.
compare_segments(3,5,1,3)	True	Aici sunt necesare 6 iterații pentru funcție.
compare_segments(3,5,2,4)	True	Aici sunt necesare 6 iterații pentru funcție.
compare_segments(3,5,1,4)	True	Aici sunt necesare 7 iterații pentru funcție.
compare_segments(3,5,2,5)	True	Aici sunt necesare 7 iterații pentru funcție.
compare_segments(3,5,1,5)	True	Am determinat deja că potențialul total al planetelor din segmentul de la 3 la 5 este mai mare sau egal decât potențialul fiecăruia din celelalte segmente. Prin urmare, acest segment ne dă răspunsul optimal. Acum programul setează valorile parametrilor funcției *find_max* după cum urmează: *left*=3 și *right*=5 și funcția își încheie lucrul aici.

Testarea locală (grader local)

Vei primi fișierele **Lgrader.cpp** și **namuhs.h** pentru testarea locală. Trebuie să le plasezi în același folder cu programul tău **namuhs.cpp** și trebuie să compilezi fișierul **Lgrader.cpp**. În acest mod vei crea un program pentru verificarea corectitudinii funcției tale. Acest program va necesita următoarele date din inputul standard:

- Pe prima linie: un număr pozitiv – numărul planetelor din listă.
- pe linia a doua: potențialele planetelor corespunzătoare.

Output-ul programului va fi răspunsul găsit de funcția ta.

Submisia testelor utilizatorului către sistem

Poți da teste sistemului. Formatul datelor de intrare trebuie să fie același ca și în grader-ul local. Rezultatul pe care îl vei primi va fi un număr mai mare decât 0, dacă răspunsul programului tău este corect, și 0, dacă răspunsul programului tău nu este corect.