Bursele Agora

Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la etapele #09-#12 ale ediției 2002/2003 a concursului de programare Bursele Agora.

P010304: Piramidă

Se consideră o piramidă triunghiulară (determinată de patru puncte necoplanare din spațiu) și alte N puncte din spațiu. Să se determine câte dintre aceste N puncte se află în interiorul piramidei. Un punct este considerat a fi în interiorul piramidei chiar dacă se află pe una dintre cele patru fețe, pe una dintre cele șase muchii sau este unul dintre cele patru vârfuri ale piramidei.

Date de intrare

Primele patru linii ale fișierului de intrare **PYRAMID.IN** conțin câte trei numere întregi reprezentând coordonatele vârfurilor piramidei. Următoarea linie conține numărul N al celorlalte puncte. Fiecare dintre următoarele N linii va conține câte trei numere întregi reprezentând coordonatele unui punct.

Date de ieșire

Fișierul de ieșire **PYRAMID.OUT** trebuie să conțină o singură linie pe care se va afla numărul punctelor din interiorul piramidei.

Restricții și precizări

- $1 \le N \le 100000$;
- coordonatele punctelor sunt numere naturale cuprinse între 0 și 255;
- coordonatele tuturor punctelor sunt date în ordinea: coordonata pe axa Ox, coordonata pe axa Oy, coordonata pe axa Oz;

Exemplu

PYRAMID.IN

```
0 0 0
0 0 100
0 100 0
100 0 0
2
100 100 100
1 1 1
```

PYRAMID.OUT

```
1
```

Timp de execuție: 1 secundă/test

P010305: Poarta Stelară

Apophis dorește să viziteze toate cele N planete pe care le stăpânește. Oricare două dintre acestea sunt conectate prin intermediul unor porți stelare. El cunoaște care este cantitatea de energie necesară deplasării de la o planetă la alta și dorește să viziteze toate planetele consumând cât mai puțină energie.

Planetele aflate sub dominația lui *Apophis* sunt numerotate folosind numere întregi cuprinse între 1 și N. Inițial el se află pe planeta sa de reședință (identificată prin 1) și, după vizitarea tuturor planetelor el trebuie să se întoarcă aici.

Pentru ca locuitorii nici uneia dintre planete să nu se simtă favorizați, *Apophis* a decis că va vizita fiecare planetă (cu excepția celei de reședință) exact o dată.

Date de intrare

Prima linie a fișierului de intrare **STARGATE.IN** conține numărul N al planetelor care îl recunosc pe *Apophis* ca zeu.

Pe următoarele linii, până la sfârșitul fișierului, se află câte trei numere întregi x, y și c cu semnificația: *pentru deplasarea între planetele x și y este necesară o cantitate de energie c.*

Date de ieșire

Prima linie a fișierului de ieșire **STARGATE.OUT** va conține cantitatea totală de energie de care are nevoie *Apophis* pentru a vizita (pornind de pe planeta de reședință) toate planetele pe care le stăpânește și pentru a reveni pe planeta de reședință.

Pe următoarea linie se vor afla numerele de ordine ale planetelor (separate prin spații), în ordinea în care vor fi vizitate.

Restricții

- $3 \le N \le 500$;
- cantitatea de energie necesară trecerii de la o planetă la alta este un număr întreg cuprins între 1 și 255;
- cantitatea de energie necesară trecerii de la planeta x la planeta y este egală cu cea necesară trecerii de la planeta y la planeta x.

Exemplu

STARGATE.IN

```
4
1 2 2
1 3 3
1 4 4
2 3 6
2 4 8
3 4 12
```

STARGATE.OUT

```
21
1 3 2 4 1
```

Modalitatea de acordare a punctajului

Vom considera că pentru fiecare test se vor putea obține cel mult *X* puncte. Concurenții care vor obține cea mică valoare *CMin* pentru cantitatea totală de energie necesară vor primi *X* puncte pentru testul respectiv.

Ceilalți concurenți, care au furnizat un traseu valid pentru care este necesară o cantitate de energie *C*, vor obține $X \cdot CMin / C$ puncte pentru testul respectiv. Această valoare va fi aproximată cu două zecimale exacte.

Punctajul final va fi obținut prin adunarea punctajelor de la fiecare test și rotunjirea acestuia la cel mai apropiat număr întreg.

Dacă traseul nu este valid, concurenții nu vor primi nici un punct pentru testul respectiv.

De exemplu, dacă pentru un test se pot obține cel mult 5 puncte, cel mai bun rezultat obținut de un concurent constă într-un traseu pentru care cantitatea necesară de energie este 89, iar un alt concurent obține o soluție pentru care cantitatea necesară de energie este 145, atunci, pentru testul respectiv, punctajul concurentului va fi de $5 \cdot 89 / 145 = 3{,}06$ puncte.

Timp de execuție: 1 secundă/test

P010306: Ordine

Se consideră un șir de numere naturale cuprinse între 0 și 1000000. Va trebui să determinați lungimea celui mai lung subșir strict crescător al șirului dat.

Date de intrare

Aceasta este o problemă cu fișierele de intrare deschise. La adresa http://www.ginfo.ro/concurs/runda11/index.shtml se află un *link* spre o arhivă care conține cele zece fișiere de intrare. Numele acestora au forma **ORDERXX.IN**, unde **XX** ia valori între **01** și **10** și reprezintă numărul testului.

Fiecare dintre cele zece fișiere de intrare va conține o singură linie pe care se vor afla elementele șirului, separate prin spații.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele zece fișiere de ieșire corecte. Acestea vor fi denumite **ORDERXX.OUT**, unde **XX** ia valori între **01** și **10** și reprezintă numărul testului.

Prima linie a fișierului de ieșire va conține textul `ORDER - TEST #XX`, unde `XX` reprezintă numărul testului. Cea de-a doua linie va conține un singur număr care reprezintă lungimea celui mai lung subșir strict crecător.

Exemplu

Vom considera că acest exemplu reprezintă testul **00**.

ORDER00.IN

```
1 3 2 5 4 8 8 9
```

ORDER00.OUT

```
ORDER - TEST #00
5
```

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține 10 puncte. Punctajul maxim care poate fi obținut (dacă toate cele 10 fișiere de ieșire sunt corecte) este de 100 de punte.

Trimiterea fișierelor de ieșire

Arhiva pe care o veți trimite se va numi **YYYYYR11.ZIP**, **YYYYYR11.RAR** sau **YYYYYR11.ACE** (în funcție de programul de arhivare pe care îl folosiți), unde **YYYYY** este codul dumneavoastră de identificare (ID).

Ea nu va conține fișiere *batch* (**YYYYYR11.BAT**) sau surse (**YYYYYR11.PAS**, **YYYYYR11.CPP** sau **YYYYYR11.C**). În arhivă se vor afla cele zece fișiere de ieșire corecte, denumirea lor fiind **ORDERXX.OUT**, unde **XX** reprezintă numărul testului (cuprins între **01** și **10**).

P080207: Sortare

Se consideră un șir *a* cu *N* elemente ale căror valori sunt numere întregi disticte care nu sunt cunoscute.

Va trebui să determinați o permutare *p* a mulțimii {1, 2, ..., *N*} astfel încât șirul *b* ale cărui elemente sunt date de formula $b[i] = a[p[i]]$ ($i = 1, ..., N$) să fie sortat în ordine crescătoare.

Pentru determinarea permutării *p* puteți efectua un număr limitat de comparări între două elementele ale șirului *a*.

Pentru aceasta aveți la dispoziție o bibliotecă externă care vă pune la dispoziție rutine pentru:

- inițializare;
- determinarea numărului de elemente ale șirului;
- determinarea numărului maxim de comparări pe care le aveți la dispoziție;
- compararea a două elemente ale șirului *a*;
- furnizarea ca rezultat a unui element al permutării *p*.

Bibliotecile Pascal și C/C++

Pentru a putea folosi biblioteca va trebui să includeți în programul dumneavoastră istrucțiunea **uses** `sort`; pentru limbajul *Pascal* și `#include "sort.h"` pentru limbajul *C/C++*.

În continuare sunt prezentate sintaxele funcțiilor și procedurilor incluse în biblioteci:

```
procedure Init;
void Init()
```
- trebuie apelată la începutul programului și este folosită de către bibliotecă pentru inițializare;
- întrerupe execuția programului dacă este apelată a doua oară.

```
function GetN:Integer;
int GetN()
```
- returnează numărul de elemente (*N*) ale șirului *a* (și ale permutării *p* care trebuie determinată).

```
function GetNoComp:Integer;
int GetNoComp()
```
- returnează numărul de comparări (apeluri ale funcției `IsGreater`) pe care le mai aveți la dispoziție (în momentul apelului funcției `GetNoComp`) pentru determinarea permutării *p*.

```
function IsGreater(i,j:Integer):Boolean;
int IsGreater(int i,int j)
```
- returnează `true` (valoare nenulă) dacă *a*[*i*] > a[*j*] și `false` (0) în caz contrar;
- întrerupe execuția programului dacă cel puțin una dintre valorile `i` și `j` nu este un număr întreg cuprins între 1 și *N* sau dacă numărul de apeluri ale acestei funcții depășește numărul maxim permis.

```
procedure Result(i,pi:Integer);
void Result(int i,int pi)
```
- acceptă ca rezultat faptul că al `i`-lea element al permutării *p* este `pi`;
- întrerupe execuția programului dacă:
  - valoarea `i` nu reprezintă un număr întreg cuprins între 1 și *N*;
  - valoarea `pi` nu reprezintă un număr întreg cuprins între 1 și *N*;
  - valoarea celui de-al `i`-lea element al permutării *p* este furnizată a doua oară;
  - valoarea corectă a celui de-al `i`-lea element al permutării *p* nu este `pi`;
  - au fost furnizate valorile tuturor celor *N* elemente ale permutării (*Acesta este singurul caz în care sunt acordate puncte!*).

Toate funcțiile și procedurile (excepție: `Init`) întrerup execuția programului dacă sunt apelate înaintea inițializării.

Restricții și precizări

- Programul vostru nu va citi date din nici un fișier de intrare și nu va scrie date în nici un fișier de ieșire.
- $2 \le N \le 255$.

Exemple de utilizare a bibliotecii

Subprogram apelat	*Valoare returnată*
`Init`	-
`GetN`	3
`GetMax`	3
`IsGreater(1,2)`	true (1)
`IsGreater(1,3)`	false (0)
`IsGreater(2,3)`	false (0)
`Result(1,2)`	-
`Result(2,1)`	-
`Result(3,3)`	-

Subprogram apelat	*Valoare returnată*
`Init`	-
`GetN`	5
`GetMax`	10
`IsGreater(1,2)`	true (1)
`IsGreater(1,3)`	true (1)
`IsGreater(1,4)`	true (1)
`IsGreater(1,5)`	true (1)
`IsGreater(2,3)`	true (1)
`IsGreater(2,4)`	true (1)
`IsGreater(2,5)`	true (1)
`IsGreater(3,4)`	true (1)
`IsGreater(3,5)`	true (1)
`IsGreater(4,5)`	true (1)
`Result(1,10)`	-
`Result(2,9)`	-
`Result(3,8)`	-
`Result(4,7)`	-
`Result(5,6)`	-
`Result(6,5)`	-
`Result(7,4)`	-
`Result(8,3)`	-
`Result(9,2)`	-
`Result(10,1)`	-

Subprogram apelat	*Valoare returnată*
`Init`	-
`GetN`	5
`GetMax`	8
`IsGreater(1,2)`	false (0)
`IsGreater(3,4)`	false (0)
`IsGreater(4,5)`	false (0)
`IsGreater(3,5)`	false (0)
`IsGreater(1,3)`	false (0)
`IsGreater(2,3)`	true (1)
`IsGreater(2,4)`	true (1)
`IsGreater(2,5)`	false (0)
`Result(1,1)`	-
`Result(2,3)`	-
`Result(3,4)`	-
`Result(4,2)`	-
`Result(5,5)`	-

Timp de execuție: 1 secundă/test