BOI 2002

Primul concurs internațional important al anului la care au participat elevi români a fost Balcaniada. Vă prezentăm acum enunțurile problemelor propuse spre rezolvare la această a zecea ediție a competiției.

P060201: Robot

În viitorul apropiat, roboții ar trebui să transporte gustările pe care le vor servi participanții la *Olimpiada Balcanică de Informatică* în timpul concursului, folosind o singură tavă de formă pătratică. Din nefericire, este foarte probabil ca drumul dintre bucătărie și sala de concurs să fie "presărat" cu tot felul de obstacole, deci un robot nu va putea să transporte tăvi oricât de mari. Sarcina voastră este să determinați care este dimensiunea maximă a unei tăvi care poate fi transportată.

Drumul pe care ar trebui să îl parcurgă robotul este printr-un coridor cu pereții paraleli și care nu poate avea decât cotituri cu 90°. Coridorul începe în direcția axei *Ox* pozitive. Obstacolele constau în stâlpi cu dimensiuni neglijabile și sunt reprezentate prin puncte. Pentru ca robotul să poată ajunge în sala de concurs, tava pe care o poartă nu poate să se lovească de pereți sau de stâlpi, dar poate să îi atingă cu marginile. Robotul și tava sa se pot deplasa doar paralel cu axele de coordonate. Dimensiunile robotului sunt întotdeauna mai mici decât cele ale tăvii, iar robotul este în întregime acoperit de tava pe care o poartă.

Date de intrare

Prima linie a fișierului **ZAD1.DAT** conține un număr întreg m ($1 \le m \le 30$) care reprezintă numărul segmentelor liniare de pereți. Următoarele $m + 1$ linii ale fișierului de intrare conțin coordonatele x și y ale tuturor punctelor de cotitură (inclusiv ale punctelor terminale) ale peretelui care inițial, are coordonata y mai mare. Similar, următoarele $m + 1$ linii conțin coordonatele x și y ale tuturor punctelor de cotitură (inclusiv ale punctelor terminale) ale peretelui care inițial, are coordonata y mai mică. Următoarea linie conține un număr întreg n ($0 \le n \le 100$) care reprezintă numărul obstacolelor. Următoarele n linii conțin coordonatele x și y ale obstacolelor. Toate coordonatele sunt numere întregi a căror valoare absolută este mai mică decât 32001.

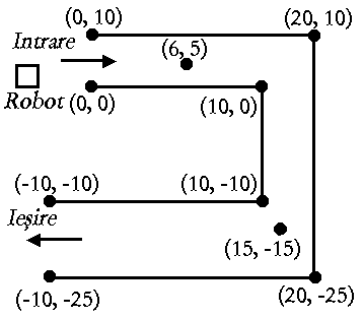
Date de ieșire

Fișierul de ieșire **ZAD1.RES** va conține o singură linie pe care se va afla un întreg care va reprezenta lungimea maximă a laturii tăvii.

Exemplu

ZAD1.DAT
```
3
0 10
20 10
20 -25
-10 -25
0 0
10 0
10 -10
-10 -10
2
15 -15
6 5
```

ZAD1.RES
```
5
```



P060202: Secvențe

Se consideră n secvențe ($2 \le n \le 10$), fiecare fiind formată din numere întregi cuprinse în intervalul $[0, 2^{31} - 1]$ dispuse în ordine crescătoare. Toate elementele din toate secvențele sunt diferite și, în total, nu sunt mai mult de 50000 elemente.

Elementele din secvențe nu sunt cunoscute, dar există posibilitatea apelării unei funcții pentru a obține valoarea unui element aflat într-o poziție a unei secvențe.

Se cere să se determine, utilizând un număr minim de apeluri ale funcției care furnizează valoarea unui element, cel de-al k-lea element din secvența rezultată în urma interclasării celor n secvențe inițiale.

Cel de-al k-lea element va fi scris pe prima linie a fișierului **ZAD2.RES**.

Instrucțiuni pentru programatorii în C

În codul sursă trebuie inserată linia:

```
#include "mergem.h"
```

Fișierul sursă trebuie linkeditat împreună cu fișierul **mergem.o**.

Interfața *C* vă pune la dispoziție următoarele funcții:

- **void** `Init()`: este folosită pentru inițializare și va fi apelată, obligatoriu, la începutul programului, înaintea apelării oricărei alte funcții a interfeței;
- **long** `GetN()`: returnează numărul de secvențe;

- **long** `GetK()`: returnează numărul de ordine al elementului care trebuie determinat, elementele fiind numerotate începând cu 1;
- **long** `GetLen(long ii)`: returnează numărul de elemente din secvența `ii`, secvențele fiind numerotate începând cu 1;
- **long** `GetA(long ii,long jj)`: returnează elementul cu numărul de ordine `jj` din secvența `ii`, elementele dintr-o secvență fiind numerotate începând cu 1;
- **void** `Finish()`: funcția va fi apelată obligatoriu înaintea terminării execuției programului.

Instrucțiuni pentru programatorii în Pascal

În codul sursă trebuie inserată linia:

```
uses mergem;
```

Interfața *Pascal* vă pune la dispoziție următoarele funcții și proceduri:

- **procedure** `init`: este folosită pentru inițializare și va fi apelată, obligatoriu, la începutul programului, înaintea apelării oricărei alte funcții a interfeței;
- **function** `GetN:Longint`: returnează numărul de secvențe;
- **function** `GetK:Longint`: returnează numărul de ordine al elementului care trebuie determinat, elementele fiind numerotate începând cu 1;
- **function** `GetLen(ii:Longint):Longint`: returnează numărul de elemente din secvența `ii`, secvențele fiind numerotate începând cu 1;
- **function** `GetA(ii:Longint; jj:Longint):Longint`: returnează elementul cu numărul de ordine `jj` din secvența `ii`, elementele dintr-o secvență fiind numerotate începând cu 1;
- **procedure** `Finish`: procedura va fi apelată obligatoriu înaintea terminării execuției programului.

Exemplu

Comunicare cu modulul		**ZAD2.OUT**
apel	*valoare returnată*	40
`GetN`	3	
`GetK`	3	
`GetLen(1)`	3	
`GetLen(2)`	4	
`GetLen(3)`	4	
`GetA(1,1)`	10	
`GetA(2,1)`	35	
`GetA(3,1)`	210	
`GetA(1,2)`	40	
`GetA(2,2)`	150	

P060203: Tur

Agenția de turism *Programatorii* trebuie să organizeze turul mănăstirilor din munții *Fruška Gora*. În acești munți există n ($3 \le n \le 1000$) mănăstiri, identificate prin numere întregi cuprinse între 1 și n. Din cauza timpului scurt nu se pot vizita toate mănăstirile, ci doar k ($3 \le k \le n$) dintre ele. Există drum direct între fiecare două mănăstiri din *Fruška Gora*, dar cum acestea sunt drumuri de munte, sunt foarte înguste, și au un singur sens de mers.

Trebuie să scrieți un program care să determine un tur care să treacă pe la exact k mănăstiri, turul trebuie să înceapă și să se termine la aceeași mănăstire, iar celelalte mănăstiri din tur să nu fie vizitate de două ori.

Date de intrare

Pe prima linie a fișierului **ZAD3.DAT** se află numerele întregi n și k, separate printr-un spațiu. Fiecare dintre următoarele $n \cdot (n - 1) / 2$ linii conține două numere întregi a și b ($1 \le a, b \le n$) care indică faptul că există drum de la mănăstirea a la mănăstirea b.

Date de ieșire

În cazul în care există un tur care respectă condițiile date, în fișierul **ZAD3.RES** se vor scrie k întregi care reprezintă mănăstirile din tur în ordinea vizitării lor. Dacă există mai multe astfel de tururi, se va scrie unul din ele. Dacă nu există soluție, în fișier se va scrie doar valoarea 0.

Exemplu

ZAD3.DAT	**ZAD3.RES**
6 3	1 3 4
3 4	
1 3	
3 5	
3 6	
3 2	
4 1	
4 5	
4 6	
4 2	
1 5	
1 6	
1 2	
5 6	
2 5	
6 2	

P060204: Extratereștrii

Extratereștrii ne vizitează planeta cu intenția evidentă de a găsi noi specii pentru grădina lor zoologică. După intrarea pe orbita Pământului ei s-au poziționat deasupra orașului *Belgrad*, unde au detectat o anumită formă de activitate. La apropierea de suprafața Pământului ei au văzut un grup de ființe semiinteligente. Aceste creaturi sunt actualii concurenți de la *BOI* 2002 care participă la excursia de după intensul concurs. Extratereștrii doresc să captureze toți cei n ($2 \le n \le 100000$) concurenți.

Extratereștrii folosesc o undă tractoare pentru a-și captura prada. Cu ajutorul undei tractoare pot fi capturate toate ființele aflate în interiorul sau pe circumferința unei suprafețe circulare. Cu cât raza undei tractoare crește (raza cercului la sol), cu atât consumul de energie este mai mare. Fiind extrem de inteligenți, extratereștrii sunt mai avansați

în domeniul științelor sociale decât în domeniul programării. Din această cauză, ei vă cer ajutorul pentru a determina poziția navei lor spațiale astfel încât energia necesară pentru capturarea celor *n* concurenți să fie minimă.

Ajutați-i pe frații extratereștri! Scrieți un program care să găsească rază minimă a zonei circulare care cuprinde toți cei *n* concurenți și poziția optimă a navei spațiale (centrul zonei circulare de la sol).

Date de intrare

Pe prima linie a fișierului **ZAD4.DAT** se află un număr întreg *n* care reprezintă numărul participanților la *BOI* 2002. Fiecare din următoarele *n* linii conține două numere reale x_i și y_i (-10000.0 ≤ x_i, y_i ≤ 10000.0), reprezentând coordonatele celui de-al *i*-lea concurent.

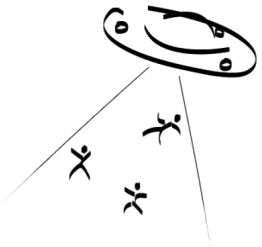
Date de ieșire

Fișierul de ieșire **ZAD4.RES** va conține două linii. Pe prima se află un singur număr real care reprezintă raza undei tractoare. A doua linie conține două numere reale, *x* și *y*, reprezentând coordonatele navei spațiale. Numerele din fișierul de ieșire vor fi scrise folosind două zecimale exacte.

Exemplu

ZAD4.DAT
```
6
8.0 9.0
4.0 7.5
1.0 2.0
5.1 8.7
9.0 2.0
4.5 1.0
```

ZAD4.RES
```
5.00
5.00 5.00
```



P060205: Arbore

O enumerare a nodurilor unui arbore este orice secvență ale cărei elemente sunt nodurile unui arbore iar fiecare nod apare exact o dată. Dacă oricare două elemente consecutive dintr-o secvență sunt noduri pentru care distanța în arbore nu este mai mare decât *d*, atunci respectiva enumerare este denumită *d*-enumerare.

Se poate demonstra că pentru orice arbore există cel puțin o 3-enumerare. Scrieți un program care să determine o 3-enumerare pentru un arbore dat.

Date de intrare

Pe prima linie a fișierului text **ZAD5.DAT** se află numărul *n* (1 ≤ *n* ≤ 30000), reprezentând numărul de noduri ale arborelui. Nodurile sunt numerotate de la 1 la *n*. Pe fiecare dintre următoarele *n* - 1 linii se află câte două numere întregi care reprezintă extremitățile uneia dintre muchiile arborelui.

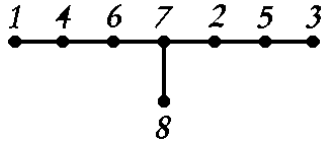
Date de ieșire

Fișierul text **ZAD5.RES** va conține o 3-enumerare a arborelui dat; fiecare linie a fișierului va conține câte un element al acestei 3-enumerări.

Exemplu

ZAD5.DAT
```
8
1 4
4 6
6 7
3 5
7 2
2 5
8 7
```

ZAD5.RES
```
3
2
6
1
4
8
7
5
```



P060206: Hartă veche

În biblioteca unei mănăstiri a fost găsită o carte veche cu hărți. Una dintre acestea conține doar castele și drumuri. Drumurile leagă castelele și nu se intersectează. Se poate călători între oricare două castele utilizând aceste drumuri. Pe hartă apar proprietăți care sunt delimitate de drumuri. Fiecare proprietate este delimitată de o linie poligonală simplă închisă formată din drumuri.

Taxa pentru o proprietate este proporțională cu numărul de castele aflate în interiorul proprietății. Cea mai mică taxă este pentru acea proprietate care nu are nici un castel sau drum în interiorul său.

Trebuie să scrieți un program care determină numărul de proprietăți de pe o hartă dată, care sunt delimitate de exact *k* drumuri și pentru care taxa este minimă.

Date de intrare

Pe prima linie a fișierului text **ZAD6.DAT** se află un număr întreg *n* (3 ≤ *n* ≤ 1000), reprezentând numărul de castele (acestea sunt identificate prin numere cuprinse între 1 și *n*). Pe fiecare dintre următoarele *n* linii se găsesc coordonatele întregi *x* și *y* (0 ≤ *x*, *y* ≤ 10000) ale unui castel, numărul *d* al castelelor vecine (cele cu care este legat direct) și cele *d* numere de ordine ale castelelor vecine. Pe ultima linie a fișierului se află numărul întreg *k* (3 ≤ *k* ≤ 1000).

Date de ieșire

Fișierul de ieșire **ZAD6.RES** va conține o singură linie pe care se va afla un număr întreg, reprezentând numărul de proprietăți pentru care taxa este minimă și care sunt delimitate de *k* drumuri.

Exemplu

ZAD6.DAT
```
17
1 5 2 2 5
4 5 3 1 3 5
6 1 3 2 4 8
7 5 4 3 5 6 11
5 8 5 13 16 1 2 4
8 4 2 4 7
8 3 1 6
10 2 4 9 10 11 3
12 2 2 10 8
12 4 4 12 11 8 9
10 5 6 14 15 4 8 10 13
11 6 2 10 13
10 9 3 5 11 12
9 7 2 15 11
7 7 2 14 11
3 10 2 5 17
2 9 1 16
4
```

ZAD6.RES
```
2
```