9. feladat: Háromszög (50 pont)

Adott a síkon egy ***P*** ponthalmaz és két kitüntetett pontja; ***a*** és ***b***. Adott továbbá egy ***q*** pont. Kiszámítandó a ***P*** ponthalmaz egy olyan ***c*** pontja, hogy a ***q*** pont az ***▲(a,b,c)*** háromszög belsejében van (nem eshet az oldalára sem), és a ***P*** ponthalmaz egyetlen más pontja sem esik a háromszögbe (oldalára sem eshet).

Írj programot (**harom.pas, harom.c, harom.cpp**), amely kiszámítja a kívánt háromszög harmadik ***c*** csúcsát, ha létezik ilyen!

### *Bemenet*

A **harom.be** szöveges állomány első sora öt egész számot tartalmaz, sorrendben a pontok ***N*** (***2≤N≤100000***) számát, a ***q*** pont ***x***- és ***y***-koordinátáját, és a kitüntetett ***a*** és ***b*** pont sorszámát (***1≤a≠b≤N***). A pontokat az ***1,…,N*** számokkal azonosítjuk. A további ***N*** mindegyike két egész számot tartalmaz egy pont koordinátáit. Az ***i+1***-edik sor az ***i***-edik pont ***x*** és ***y*** (***30000≤x,y≤30000)*** koordinátáit tartalmazza.

### *Kimenet*

A **harom.ki** szöveges állomány első és egyetlen sorába egy egész számot kell írni, a kívánt háromszög harmadik csúcsának sorszámát! Ha nincs megoldás, akkor az első sorba az egyetlen ***0*** számot kell írni. Több megoldás esetén bármelyik kiírható.

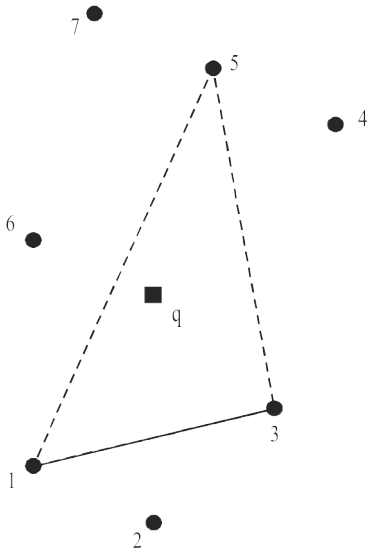
Példa bemenet és kimenet:

harom.be

```
7 4 5 1 3
2 2
4 1
6 3
7 8
5 9
2 6
3 10
```

harom.ki

```
5
```



### *Pascal programozóknak:*

```
Function ForgasIrany(x0,y0,x1,y1,x2,y2:integer):Integer;
 {Kimenet: +1 ha (x0,y0)->(x1,y1)->(x2,y2) balra fordul,
            0 ha egy egyenesre esnek
           -1 ha (x0,y0)->(x1,y1)->(x2,y2) jobbra fordul.}
Var
  KeresztSzorz:Int64;
Begin{ForgasIrany}
  KeresztSzorz:=(Int64(x1)-x0)*(Int64(y2)-y0)-(Int64(x2)-x0)*(Int64(y1)-y0);
  If (KeresztSzorz < 0) Then ForgasIrany:=-1
  Else If KeresztSzorz>0 Then ForgasIrany:=1
  Else ForgasIrany:=0;
End{ForgasIrany};
```

### *C++ programozóknak:*

```
int ForgasIrany(int x0,int y0, int x1,int y1, int x2, int y2) {
/*
Kimenet: +1 ha (x0,y0)->(x1,y1)->(x2,y2) balra fordul,
         0 ha egy egyenesre esnek
         -1 ha (x0,y0)->(x1,y1)->(x2,y2) jobbra fordul.
*/
  double p1xp2=((double)x1-x0)*((double)y2-y0)-((double)x2-x0)*((double)y1-y0);
  if (p1xp2==0.0) return 0;
  if (p1xp2>0.0) return 1;
  else return -1;
}
```

10. feladat: Hálózat (50 pont)

Egy számítógépes hálózat csomópontokat és bizonyos csomópont-párokat közvetlenül összekötő kétirányú adatátvitelt biztosító adatátviteli vonalakat tartalmaz. A hálózat tartalmaz egy kitüntetett csomópontot, a központi csomópontot. A hálózat üzemeltetői tudni akarják, hogy melyek azok a csomópontok, amelyek akkor is elérhetőek a központból, ha bármely egyedi közvetlen vonal meghibásodik.

Írj programot (**`halozat.pas, halozat.c, halozat.cpp`**), amely kiszámítja azokat a csomópontokat, amelyek akkor is elérhetők a központi csomópontból, ha bármely egyedi közvetlen vonal meghibásodik!

### *Bemenet*

A **`halozat.be`** szöveges állomány első sorában a csomópontok ***N*** (***1<N≤5000***) száma, a közvetlen vonalak ***M*** (***1<M≤20000***) száma, és a központi csomópont ***K*** sorszáma van. A csomópontokat az ***1,…,N*** számokkal azonosítjuk. A további ***M*** sor mindegyike egy ***u v*** (***1≤u,v≤N, u≠v***) számpárt tartalmaz, ami azt jelenti, hogy az ***u*** és ***v*** csomópontot közvetlen vonal köti össze, amin ***u***-ból ***v***-be és ***v***-ből ***u***-ba lehet adatot átvinni. Bármely két csomópont között legfeljebb egy közvetlen vonal van.

### *Kimenet*

A **`halozat.ki`** szöveges állomány első sorába azon csomópontok ***C*** számát kell írni, amelyek bármely egyedi vonal meghibásodása esetén is elérhetők a központi csomópontból! A második sor pontosan ***C*** egész számot tartalmazzon, az 1-meghibásodás esetén is elérhető csomópontok sorszámait, tetszőleges sorrendben.

Példa bemenet és kimenet:

`halozat.be`

```
11 12 3
3 2
2 4
4 3
1 3
2 6
6 10
6 11
2 11
5 7
5 8
5 4
7 8
```

`halozat.ki`

```
5
3 2 4 6 11
```