Aranycipők

Bergengócia labdarúgó bajnokságában minden idény végén kiosztanak aranycipőt annak a játékosnak, aki a legtöbb gólt lőtte. Ha több játékos holtversenyben szerezte a legtöbb gólt, akkor mindegyikük kap egy-egy aranycipőt. A bajnokságban N játékos lőtt legalább egy gólt, és a Bergengóc Labdarúgó Szövetségnek (BLSZ) van egy listája az egyes játékosok által lőtt gólok számáról. A segítségedet kérik az aranycipős játékosok meghatározásában.

Írj programot, amely az egyes játékosok által lőtt gólok száma alapján meghatározza, hogy hány góllal lehetett aranycipőt szerezni, majd megadja az aranycipős játékosok számát, valamint ezen játékosok sorszámait is!

Bemenet

A `standard bemenet` első sorában a góllövő játékosok száma ($1 \leq N \leq 1000$) van. A következő N sor mindegyike egy-egy játékos által lőtt gólok számát tartalmazza ($1 \leq G_i \leq 100$).

Kimenet

A `standard kimenet` első sorában egyetlen egész szám legyen, a legtöbb gól, amit egy játékos szerzett! A második sorba az aranycipős játékosok K számát írd! A következő K sorban az aranycipős játékosok sorszámai legyenek, növekvő sorrendben, soronként egy-egy!

Példa

Bemenet	Kimenet
7 2 11 8 3 11 1 10	11 2 2 5 Magyarázat: 11 a legtöbb gól, amennyit lőttek, és 2 játékos lőtt ennyi gólt, a második és ötödik játékos.

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Pontozás

A pontok egyharmadát lehet szerezni, ha a kimenet első sora helyes, és további egyharmadát, ha a második sor helyes.

A lehető legkevesebb átszállás

Egy vasútvonal állomásai között személyvonatok közlekednek. A személyvonatok a két végállomásuk között minden közbülső állomáson megállnak. Az állomásokat a vasútvonal mentén növekvő sorszámokkal azonosítjuk.

Írj programot, amely megadja, hogy minimum hányszor kell átszállni, hogy eljuthassunk az első állomásról az utolsóra!

Bemenet

A `standard bemenet` első sorában a vonatok száma ($1 \le N \le 10\,000$) és az állomások száma ($2 \le M \le 100\,000$) van. A következő `N` sorban egy-egy vonat két végállomásának sorszáma található ($1 \le A_i < B_i \le M$), induló állomások szerint növekvő sorrendben.

Kimenet

A `standard kimenet` első sorába az átszállások minimális K számát kell írni, amellyel eljuthatunk az első állomásról az utolsóra! A második sor K+1 vonat sorszámát tartalmazza, növekvő sorrendben, amelyekkel K átszállással el lehet jutni az utolsó állomásra! Több megoldás esetén bármelyik kiírható.

Ha nem lehetséges eljutni az első állomásról az utolsóra, akkor egyetlen sorba `-1`-et kell kiírni!

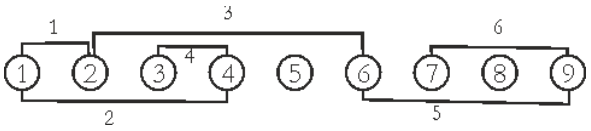
Példa

`Bemenet`

```
6 9
1 2
1 4
2 6
3 4
6 9
7 9
```

`Kimenet`

```
2
2 3 5
```



Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Pontozás

A pontok 30%-a szerezhető olyan tesztekre, ahol $N \le 1000$.

Elágazás nélküli úton levő települések

Településeket kétirányú közvetlen utak kötnek össze. Bármely két település között legfeljebb egy közvetlen út létezhet. *Zsákfalunak* hívjuk azokat a településeket, amelyek pontosan egy másik településhez kapcsolódnak közvetlen úttal.

Írj programot, amely megadja azokat a településeket, amelyek valamely zsákfaluból elérhetőek elágazás nélküli, egy vagy több közvetlen útból álló útvonalon!

Bemenet

A `standard bemenet` első sorában a települések száma ($2 \le N \le 10\,000$) és az utak száma ($1 \le M \le 100\,000$) van. A következő M sorban két-két település sorszáma található ($1 \le A_i \ne B_i \le N$), amelyeket közvetlen út köt össze.

Kimenet

A `standard kimenet` első sorába azon települések K számát kell írni, amelyek valamely zsákfaluból elágazás nélküli útvonalakon érhetők el! A második sor ezen K település sorszámait tartalmazza, növekvő sorrendben! Ha $K=0$, akkor a második sor legyen üres!

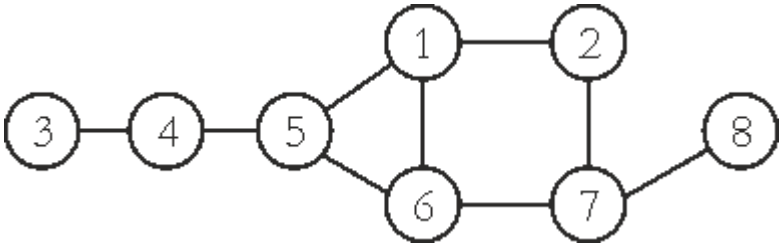
Példa

Bemenet

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

Kimenet

```
3
4 5 7
```



Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Negáló rendezés

A Mirákulum™ szoftvercég informatikusai napról napra azon fáradoznak, hogy az adatbázisaikban fellelhető, felfoghatatlan mennyiségű adatot minél hatékonyabban tudják kinyerni és rendezett formában továbbítani a felhasználók felé. Ehhez most kifejlesztettek egy új eljárást, a NegálóRendezés™-t.

Adott egy nemnegatív egészeket tartalmazó sorozat. Egy NegálóRendezés™ során:

1. Tetszőleges számú, szabadon kiválasztott tömbelemet a `-1`-szeresére cserélünk.
2. Rendezzük az elemeket nagyság szerint növekvő sorrendbe.
3. Minden elemet kicserélünk az abszolút értékére, azaz elhagyjuk az első lépésben megváltoztatott előjeleket.

Így végül az eredeti sorozat elemeinek egy átrendezését kapjuk eredményként.

Például az `[5,1,2,3,2]` sorozat esetén a rendezés egy lehetséges alkalmazása:

1. Változtassuk negatívra az első és a második elemet: `[-5,-1,2,3,2]`.
2. Rendezzük az elemeket nagyság szerint: `[-5,-1,2,2,3]`.
3. Állítsuk vissza az előjeleket: `[5,1,2,2,3]`.

Írj programot, amely meghatározza, hogy egy adott sorozatra egyszer alkalmazva a NegálóRendezés™-t hányféle különböző sorozatot kaphatunk! Mivel ez a szám viszonylag nagy is lehet, ezért a 10^9+7-tel vett osztási maradékát kell megadni! Két sorozat pontosan akkor különböző, ha legalább egy pozíción különböző értékű számot tartalmaznak.

Bemenet

A `standard bemenet` első sorában a sorozat elemeinek száma ($2 \le N \le 100\,000$) áll. A második sor tartalmazza a sorozat elemeit ($0 \le a_i \le 100\,000$).

Kimenet

A `standard kimenetre` egyetlen egész szám kerüljön, a rendezés eredményeként kapható különböző sorozatok darabszáma modulo 10^9+7!

Példa

Bemenet	Kimenet
5 5 1 2 3 2	12

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Pontozás

A pontszám 20%-a szerezhető olyan tesztekre, ahol $N \le 8$.

A pontszám további 20%-a szerezhető olyan tesztekre, ahol a sorozat elemei különbözők.

Sípálya

A Bájtra hegységben N hegycsúcs helyezkedik el egymás mellett. Egy olyan sípályát szeretnénk kialakítani, mely K darab egymás melletti csúcsból áll. Az *optimális lesiklási élmény* érdekében, ha a sípálya legelső csúcsa H magasságú, akkor a többi csúcs sorban H-1,H-2,…,H-K+1 magasságú kell, hogy legyen. A következő műveletet tudjuk akárhányszor elvégezni: egy petákért földet szállíttatunk tetszőleges hegycsúcs tetejére és ezzel megnöveljük a magasságát 1-gyel.

Írj programot, ami meghatározza, hogy minimálisan mennyi petákba kerül kialakítani egy K hosszú sípályát!

Bemenet

A standard bemenet első sorában a hegycsúcsok száma ($1 \le N \le 200\,000$) és a kialakítandó sípálya hossza ($1 \le K \le N$) található. A második sorban N pozitív egész szám van, a hegycsúcsok magasságai ($1 \le H_i \le 10^9$).

Kimenet

A standard kimenetre egyetlen egész szám kerüljön, a K hosszú sípálya kialakításának minimális költsége!

Példa

Bemenet	Kimenet
5 3 5 5 6 3 1	4

Korlátok

Időlimit: 0.4 mp.

Memórialimit: 64 MB

Pontozás

A pontszám 20%-a szerezhető olyan tesztekre, ahol $N \le 1000$.

A pontszám további 20%-a szerezhető olyan tesztekre, ahol $N \le 10\,000$.

Világnaptár

A világnaptár Elisabeth Achelis javaslatára a The World Calendar Association gondozásában jelent meg 1930-ban. A szervezet azóta több alkalommal kampányolt a bevezetéséért, sikertelenül.

December 30-a után minden év egy világnappal végződik, ez az év 365. napja. Ez a nap teljesen független, nem része az adott hétnek. Szökőévek esetén ugyanígy egy, az adott héttől független szökőnapot iktat be június 30-a után. A világnaptárban január, április, július és október 31 napos, a többi hónap pedig 30 napos.

Írj programot, amely egy szokásos naptár szerinti dátumot átalakít világnaptár szerinti dátummá!

Bemenet

A `standard bemenet` első sorában a szokásos naptár szerinti dátum van ($1901 \leq \text{Év} \leq 2099$, `Hó` és `Nap` helyes a naptár szerint). Ebben az időszakban pontosan a 4-gyel osztható évszámok a szökőévek.

Kimenet

A `standard kimenet` első sorába a világnaptár szerinti dátumot kell írni! Szökőnap esetén a napsorszám helyére az `SZN`, világnap estén pedig a `VN` szöveget kell kiírni!

Példa

Bemenet	Kimenet
2021 10 11	2021 10 11
Bemenet	Kimenet
2020 7 1	2020 6 SZN
Bemenet	Kimenet
2019 12 31	2019 12 VN

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Első negyedév		
Január	**Február**	**Március**
V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Második negyedév		
Április	**Május**	**Június**
V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 **SzN**
Harmadik negyedév		
Július	**Augusztus**	**Szeptember**
V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Negyedik negyedév		
Október	**November**	**December**
V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	V H K S C P S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 **VN**
SzN – Szökőnap (a hét rendszeren kívül) **VN** – Világnap (a hét rendszeren kívül)		