Málna

Két medvebocs, Artúr és Bendegúz a málnásba mennek lakmározni. A málnásban N bokor van egy sorban, és már megszámolták, hogy az egyes bokrokon hány málna van. Néhány szomszédos bokorról akarják az összes málnát megenni, természetesen úgy, hogy ugyanannyit egyenek mindketten, tehát a málnák összdarabszáma páros legyen. Most azon gondolkodnak, hogy hányféleképpen választhatják ki ezt a néhány szomszédos bokrot.

Készíts programot, amely kiszámítja, hogy hányféleképpen lehet kiválasztani néhány szomszédos málnabokrot úgy, hogy a rajtuk lévő összes málna száma páros legyen!

Bemenet

A *standard bemenet* első sorában a bokrok száma van ($1 \le N \le 100\,000$). A második sorban az egyes bokrokon lévő málnák darabszáma ($1 \le M_i \le 1000$) van felsorolva.

Kimenet

A *standard kimenet* első sorába a két medvebocs választási lehetőségeinek számát kell írni, vagyis az olyan különböző `i,j` ($1 \le i \le j \le N$) indexpárok számát, amire $M_i+M_{i+1}+\ldots+M_j$ páros!

Példa

Bemenet	Kimenet
5 3 1 4 5 2	6

Magyarázat: a 6 féle lehetőség: `3+1, 3+1+4, 1+4+5, 1+4+5+2, 4, 2.`

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Pontozás

A pontok 20%-a szerezhető olyan tesztekre, ahol $N \le 100$.

A pontok további 40%-a szerezhető olyan tesztekre, ahol $N \le 5000$.

Posta

Egy postán K-féle ügyintézési lehetőség van pontosan K ügyintézővel, típusonként eltérő ügyintézési időkkel. Most a járvány miatt a sor kint áll a posta előtt, egy alkalmazott megnézi, hogy az első ember milyen ügyet akar intézni, és ha annál az ügyintézőnél nincs senki, akkor beengedi, majd nézi a következő embert. Ha a kinti sor elsőjének várnia kell, akkor az összes többi mögötte állónak is, hiába lenne szabad az ügyintézője. Egyszerre többen is jöhetnek és a sor elejéről egyszerre többen is mehetnek az ügyintézőjükhöz, ha az szabad. Ha egy ember az E. időpontban kezdi az ügye intézését és az U időegységig tart, akkor az adott ügyintézőhöz leghamarabb az E+U. időpontban jöhet az újabb ügyfél. A sorban várakozók számát mindig az adott pillanatban történő beengedések és érkezések után tekintjük, közben nem.

Készíts programot, amely megadja, hogy a legtovább várakozó ember mennyi ideig várakozott, mielőtt az ügyét elkezdték volna intézni, valamint mekkora volt a posta előtti sor leghosszabb hossza!

Bemenet

A *standard bemenet* első sorában az emberek száma ($1 \le N \le 100\,000$), valamint az ügyintézési lehetőségek száma ($1 \le K \le 100$) van. A második sor az egyes ügyek intézési idejét tartalmazza ($1 \le U_i \le 100$). A következő N sor mindegyikében egy ember érkezési ideje ($1 \le E_i \le 1\,000\,000$), növekvő sorrendben, valamint az ügyének a sorszáma ($1 \le S_i \le N$) szerepel.

Kimenet

A *standard kimenet* első sorába a legtovább várakozó ember várakozási idejét kell írni! A második sorba a posta előtt várakozók maximális számát kell írni!

Példa

Bemenet

```
6 3
5 10 15
1 1
10 2
11 2
12 1
13 1
16 3
```

Kimenet

```
12
4
```

Az egyes emberek ügyintézési időadatai és a sorban várakozási ideje: 1-5-(0), 10-19-(0), 20-29-(9), 20-24-(8), 25-29-(12), 25-40-(9). Az első és a második nem várt semmit, a harmadik 9 percet, a negyedik 8 percet várt, az ötödik várta a legtöbbet, 12 percet (a 13. percben jött, a 25.-ben kezdte az ügyét intézni).

A harmadik embertől kezdve mindenki vár a második ügyének befejezésére, azaz 4 várakozó is volt egyszerre, 16-tól 19-ig.

Korlátok

Időlimit: 0.1 mp.

Memórialimit: 32 MB

Pontozás

A pontok 48%-a szerezhető olyan tesztekre, ahol $N \le 1000$.

Székek

Egy tanteremben eggyel több szék található, mint ahány diáknak helyet kellene adni. A terembe érkezéskor mindenki leült valamelyik székre, de csak utólag tudták meg, hogy melyik székre kellett volna leülniük – utólag sorszámozzák be a székeket és az i.-nek érkező diáknak az i sorszámú széken kellene ülne. A várt elhelyezkedéshez egyetlen típusú lépést tehetnek: valamelyik diák feláll a székéről és átül az üres székre.

Készíts programot, amely megadja, hogy minimum hány átüléssel érhető el, hogy mindenki a neki szánt székre kerüljön!

Bemenet

A *standard bemenet* első sorában a diákok száma van ($1 \le N \le 100\,000$). A második sor i. száma annak a széknek a sorszáma, ahova az i. diák ült ($1 \le S_i \le N+1$).

Kimenet

A *standard kimenet* első sorába a minimális átülés számot kell írni, amellyel elérthető, hogy mindenki a neki szánt székre kerüljön!

Példa

Bemenet

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

Kimenet

```
6
```

Magyarázat: először a második széken ülő diák átül a hatodik székre, utána a nyolcadik széken ülő diák átül a második székre, ...

Magyarázat – a zöldek a saját helyükön ülnek (az első átülés előtt a 3-as és a 7-es):

1.	*2.*	*3.*	*4.*	*5.*	*6.*	*7.*	*8.*
5	6	**3**	1	4		**7**	2
5		**3**	1	4	**6**	**7**	2
5	**2**	**3**	1	4	**6**	**7**	
5	**2**	**3**	1		**6**	**7**	4
	2	**3**	1	**5**	**6**	**7**	4
1	**2**	**3**		**5**	**6**	**7**	4
1	**2**	**3**	**4**	**5**	**6**	**7**	

Korlátok

Időlimit: 0.1 mp.

Memórialimit: 32 MB

Pontozás

A pontok 70%-a szerezhető olyan tesztekre, ahol $N < 5000$.

További 20% szerezhető olyan tesztekre, ahol $N \le 20\,000$.

Versenyeredmények

Egy programozási versenyen a résztvevők H karakteres azonosítót kaptak (az angol ábécé kisbetűiből). Szeretnénk közzétenni az eredményeket, de nem akarjuk felfedni a teljes azonosítót, csak az elejüket, amennyire muszáj ahhoz, hogy a listában mindenki beazonosíthassa a saját eredményét.

Készíts programot, amely kipontozza a lehető legtöbb karaktert az azonosítók végéről, majd kiírja a verseny végeredményét!

Bemenet

A *standard bemenet* első sorában a versenyzők száma ($1 \leq N \leq 200\,000$), valamint az azonosítók hossza ($1 \leq H \leq 100$) van. A következő N sorban az egyes versenyzők azonosítója és az általuk szerzett pontszám szerepel ($1 \leq P_i \leq 1000$) szerepel.

Kimenet

A *standard kimenet* N sorába bemenet szerinti sorrendben kell írni az eredményeket, kipontozva a nem közzéteendő részüket!

Példa

Bemenet	Kimenet
4 5 acadv 194 bacdv 123 aabcd 117 aacdv 76	ac... 194 b.... 123 aab.. 117 aac.. 76

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 64 MB

Pontozás

A pontok 40%-a szerezhető olyan tesztekre, ahol N<1000.

Zászló

Egy N milliméteres korlátot a magyar zászló színeire, piros-fehér-zöld színűre festettek, nem tudjuk azonban, hogy ezen belül milyen színű rész milyen hosszú.

Készíts programot a piros, fehér és zöld rész hosszának meghatározására.

Könyvtár

A megoldáshoz a `zaszlo` könyvtár műveleteit kell használni! Programod nem olvashat és nem írhat semmilyen fájlt, beleértve a standard bemenetet és kimenetet.

`#include "zaszlo.h"`	A függvény osztály használatba vétele.
`int kezdet();`	Megadja a korlát hosszát ($20 \le N < 1\,000\,000$). A programod elején, egyszer kell hívni!
`int kerdes(int a);`	A függvény értéke 1, ha a korlát `a`. milliméteres szakasza piros színű; 2, ha fehér és 3, ha zöld.
`void eredmeny(int p,int f, int z);`	Rendre a piros, fehér és zöld színű szakaszok hosszát kell a paraméterekben megadni! A programod végén, egyszer kell hívni! Végrehajtásával a program terminál.

Használat

Teszteléshez letölthető a `zaszlo` könyvtár C++ programja (nem feltétlenül azonos az értékelő rendszerben levővel).

A `kezdet` a standard bemenet első sorából beolvassa a korlát hosszát, a második sorból pedig a piros, fehér és zöld szakasz hosszát (mindhárom nagyobb 0-nál, összegük a korlát hossza).

Az `eredmeny` a standard kimenet első sorába a TRUE vagy FALSE szót írja annak megfelelően, hogy a helyes-e a három hossz.

Példa a használathoz

bemenet	kimenet
`kezdet();`	`100`
`kerdes(50);`	`2`
`kerdes(80);`	`3`
`eredmeny(30,40,30);`	`// TRUE`

Korlátok

Időlimit: 0.1 mp.

Memórialimit: 32 MB

Pontozás

Helyes megoldás esetén a pontszám 20%-át kapod, ha a `kerdes` hívások száma több mint `min(n/2,100)`.

Matekbolha

Matekbolha egy algebrai kifejezésen ugrál. A kifejezésben csak a négy alapművelet, kerek zárójelek, és változók lehetnek, amelyeket egy kisbetű jelöl. A bolha mindig a kifejezés egyik karakterén tartózkodik, ami lehet műveleti jel, változó, vagy zárójel is. Egy lépésben ugorhat balra vagy jobbra a szomszédos karakterre, vagy ha éppen egy zárójelen tartózkodik, akkor annak a párjára. Például, ha az y+x*(z+y) kifejezésben a (karakteren áll, akkor ugorhat a *-ra, a z-re, vagy a)-re is. Természetesen lehetnek a kifejezésben egymásba ágyazott zárójelek is. Egy zárójel párján – szokásos módon – azt az egyértelműen meghatározott másik zárójelt értjük, ahol a megfelelő számítási műveletnek a másik vége van. Tehát például ha Matekbolha az x*(y+u+z/(x-y)+w) kifejezésben az utolsó) karakteren áll, akkor ugorhat az első (karakterre, valamint a w-re. Az algebrai kifejezésben az x változó pontosan kétszer szerepel, és Matekbolha az első x-ről szeretne eljutni a második x-re úgy, hogy egyik karakterre sem ugrik kétszer.

Készíts programot, amely kiszámítja, hogy legalább, illetve legfeljebb hány ugrással tud eljutni a bolha a kifejezésben az első x-ről a másodikra!

Bemenet

A *standard bemenet* első sorában az algebrai kifejezés hossza van ($1 \le N \le 100\,000$). A második sorban található maga a kifejezés, amely csak az `a..z`,`+`,`-`,`*`,`/`,`(`,`)` karaktereket tartalmazhatja, és pontosan két `x` van benne.

Kimenet

A *standard kimenet* első sorába az ugrások minimális számát, a második sorába pedig az ugrások maximális számát kell írni, amivel a bolha el tud jutni az első x-ről a másodikra! A két részfeladat egymástól teljesen független.

Példa

Bemenet	Kimenet
16 a/(z*(x+y)*(-x))	6 12

Magyarázat: a bolha egy lehetséges útja az első esetben (a karakterek sorszámaival, 1-től indexelve): 7→6→10→11→12→13→14.
A második esetben: 7→8→9→10→6→5→4→3→16→15→12→13→14.

Korlátok

Időlimit: 0.2 mp.

Memórialimit: 32 MB

Pontozás

Minden tesztben a pontok fele jár külön-külön az első és a második részfeladatra.

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

A pontok további 20%-a szerezhető olyan tesztekre, ahol $N \le 500$.

A pontok további 20%-a szerezhető olyan tesztekre, ahol a kifejezésben csak egy zárójelpár van.