Velmi dlouhá police

task: `bookshelf`	points: 100

V údolí tak úzkém, že by se dalo překročit, a tak dlouhém, že i kdyby bylo o polovinu kratší, tak by pořád bylo dlouhé, ležela malá víska Vyšná Boca. A protože i chvíle tam je obvykle dlouhá, napadlo místní vesničany, že by si ji mohli zkrátit čtením kvalitní a hodnotné literatury.

Protože je brzy přestalo bavit číst kolem dokola stále ty samé tři knihy, rozhodli se, že si postaví knihovnu. Aby se vešla do údolí musela být úzká. Hodně úzká. Tak úzká, že se do ní vešla jen jedna dlouhatánská polička plná knih. Čas od času přijde nějaký vesničan s dvoukolákem, vytáhne si několik po sobě jdoucích knih a odveze si je domů. Jindy zase přijde někdo s plným trakařem knih, zvolí si místo v poličce, knihy roztáhne a do vzniklé mezery vloží ty, které přivezl.

Zadání úlohy

Knihy jsou očíslované kladnými celými čísly menšími než $2 \cdot 10^9$. V knihovně může být více kopií téže knihy. *Oblíbené* knihy mají čísla menší nebo rovna 100. Pozice, na kterých knihy leží, jsou očíslovány zleva doprava, nejlevější políčko má index 0. Úplně na začátku je knihovna prázdná.

Vaším úkolem je napsat modul, který bude simulovat chod knihovny. Soubor, který odevzdáte, musí obsahovat implementace následujících funkcí.

- `void init()`
  `procedure init`
  Tato funkce se zavolá pouze jednou na začátku.
- `int examine_book(int p)`
  `function examine_book(p : longint) : longint`
  Tato funkce by měla vrátit knihu, která je nyní na pozici `p`. (Kniha na polici zůstane.)
- `void add_books(int k, int n, int *b)`
  `procedure add_books(k, n : longint; b : array of longint)`
  Tato funkce by měla vzít `n` knih z pole `b` a vložit je do poličky od pozice `k`.
- `void remove_books(int k, int n)`
  `procedure remove_books(k, n : longint)`
  Tato funkce by měla z police odstranit `n` po sobě jdoucích knih, počínaje knihou na pozici `k`.
- `int best_books(int begin, int end)`
  `function best_books(begin,end : longint) : longint`
  Tato funkce by měla vrátit počet oblíbených knih, které jsou na pozicích $begin, begin + 1, \ldots, end$. (Knihy zůstávají v polici.)

Vyhodnocování

Váš modul bude testovaný na mnoha různých sadách vstupních dat. Test proběhne úspěšně jen tehdy, skončíli volání všech vašich funkcí korektně a v časovém limitu a všechna volání `examine_book` a `best_books` vrátí správné knihy nebo správné číslo.

Každá sada testovacích dat může být popsána třemi čísly: celkový počet volání funkcí C, které udělá náš vyhodnocovací systém, maximální počet knih B v poličce v libovolném okamžiku, celkový počet přidaných nebo odebraných knih N a číslo Q – součet délek úseků, na které se ptáme funkcí `best_books`. Navíc některé sady nebudou obsahovat žádné volaní `remove_books`.

Limity pro různé testovací sady vypadají následovně:

body	10	10	10	10	10	10	10	10	10	10
C	1000	2000	250000	250000	250000	500000	700000	500000	500000	500000
B	1500	2500	125000	125000	250000	700000	750000	750000	800000	10^6
N	1500	4000	250000	125000	250000	700000	750000	750000	$5 \cdot 10^6$	$5 \cdot 10^6$
Q	10000	10000	500000	10^9	10^6	10^{10}	10^7	10^7	10^{10}	10^9
remove?	ne	ano	ano	ne	ne	ne	ne	ne	ano	ano

Příklad

input

```
init()
add_books(0, 5, [1, 2, 3, 4, 5])
add_books(2, 2, [1000, 1002])
examine_book(3)
remove_books(3, 3)
best_books(1, 3)
```

output

```
examine_book(3): 1002
best_books(1, 3): 2
```

Slovenský dvojkríž

task: `doublecross`	points: `100`

Jedním z národních symbolů Slovenska je Slovenský znak, zobrazený na obrázku vpravo. Nejdůležitější částí znaku je stříbrný dvojkříž. Tento symbol můžete nalézt i na některých slovenských euromincích a samozřejmě i v této úloze.

Dvojkříž snadno nakreslíme jako bitmapu, zda jsou dva příklady:

```
..........
....1.....
..11111...
....1.....
111111111.
....1.....
....1.....
```

```
..1..
.111.
..1..
11111
..1..
```

Formálně: *dvojkříž* se skládá z jedné svislé a ze dvou vodorovných částí, které jsou tvořeny jedničkami. Navíc musí být splněné následující podmínky:

- Vodorovné části nesmí být v sousedních řádcích.
- Svislá část musí začínat nad vodorovnými částmi a končit pod nimi.
- Svislá část musí rozdělovat vodorovné části na dvě stejně velké části.
- Vrchní vodorovná část musí být kratší než spodní.

Všimněte si, že bitmapa vpravo obsahuje nejmenší možný dvojkříž.

Zadání úlohy

Máte danou matici `0` a `1`. Vypište hodnotu (D modulo $10^9 + 9$), kde D je počet výskytů dvojkříže v matici.

Jakmile jsou podmínky dvojkříže splněné, nemusíte se starat o obsah okolních políček. (V uvedených příkladech mohou pozice s tečkami obsahovat jak `0`, tak `1`.)

Pokud se stejný kříž vyskytne na různých místech, nebo existuje více překrývajících se křížů, započítejte je všechny.

Formát vstupu

První řádek obsahuje dvě čísla R a S: počet řádků a počet sloupců matice. Řádky jsou číslované od 1 po R shora dolů, sloupce od 1 do S zleva doprava.

Druhý řádek obsahuje jedno číslo N – počet políček, které obsahují `0`.

Následuje N řádků. Každý řádek obsahuje dvě čísla r_i a s_i – souřadnice (řádek a sloupec) políčka, které obsahuje `0`. Žádné dvě z těchto N políček nejsou shodné.

Omezení

Testovací vstupy jsou rozdělené do několika skupin. Pro každou skupinu jsou dány maximální hodnoty R, S, N a počet bodů, které za ní můžete získat.

body	10	5	5	15	15	20	5	5	5	15
R	10	10	100	100	2500	4000	400	400	400	100
S	10	10	100	6000	50	30	3000	3000	3000	10000
N	100	5	15	1000	200	1000	1	5	15	10000

Navíc první skupina je značně usnadněná tím, že v každém vstupu této skupiny všechny `1` tvoří dvojkříž.

Formát výstupu

Vypište jeden řádek obsahující číslo (D modulo $10^9 + 9$), kde D je počet způsobů, jak může být dvojkříž umístěný.

Příklad

input

```
6 8
12
1 2
1 3
1 4
1 6
2 2
3 2
3 3
3 4
3 7
6 4
6 6
4 8
```

output

```
5
```

Takhle vypadá odpovídající vstupní matice:

```
10001011
10111111
10001101
11111110
11111111
11101011
```

A obsahuje následujících pět dvojkříží:

```
....1...   ....1...   ....1...   ....1...   ....1...
...111..   ...111..   ...111..   ...111..   ..11111.
....1...   ....1...   ....1...   ....1...   ....1...
..11111.   ..11111.   ....1...   ....1...   ....1...
....1...   ....1...   ..11111.   .1111111   .1111111
........   ....1...   ....1...   ....1...   ....1...
```

Poskakující posloupnost

task: `updown`	points: `100`

Charakteristikou posloupnosti celých čísel $a_1, \ldots, a_n$ je řetězec S délky $n-1$ definovaný následovně:

$$S[i] = \begin{cases} \texttt{I} & \leftarrow a_i < a_{i+1} \text{ (posloupnost roste)} \\ \texttt{C} & \leftarrow a_i = a_{i+1} \text{ (posloupnost je konstantní)} \\ \texttt{D} & \leftarrow a_i > a_{i+1} \text{ (posloupnost klesá)} \end{cases}$$

Například posloupnost $(3, 1, 1, 5, 6, 2)$ má charakteristiku `DCIID`.

Některé posloupnosti jsou nudné. Například $(1, 5, 2, 7, 3, 6, 2, 8)$. Nuda, že? Jen skáče nahoru a dolu. Toto je pěkně vidět, pokud se podíváme na její charakteristiku: `IDIDIDI`.

Nyní si *stupeň nudnosti (SN)* definujme formálně: Je to největší celé číslo k takové, že její charakteristika obsahuje dva stejné podřetězce délky k. (Tyto podřetězce se smí i překrývat.)

Například výše uvedená posloupnost $(1, 5, 2, 7, 3, 6, 2, 8)$ má SN rovný pěti.

Zadání úlohy

Pro dané k sestrojte *nejdelší možnou* posloupnost celých čísel, jejíž SN je ostře menší než k.
Navíc by vaše posloupnost měla obsahovat *nejmenší možný počet* navzájem různých prvků. (Nějaké body ale dostanete i tehdy, pokud splníte pouze první požadavek a nikoliv druhý.)
Velikost programu, který odevzdáte, nesmí překročit 10 000 znaků

Formát vstupu

Na jediném řádku vstupu je jedno celé číslo k ($1 \leq k \leq 14$).
Všechny možné hodnoty k budou použité jako vstupy. Za velká k bude víc bodů než za malá.

Formát výstupu

Vypište svoji posloupnost, do každého řádku jeden prvek. Každý prvek se musí vejít do 32b celočíselné proměnné se znaménkem.

Příklad

input	output

`2`	`1`
	`2`
	`3`
	`3`
	`3`
	`2`
	`47`
	`2`
	`1`

Výstupní posloupnost má charakteristiku `IICCDIDD`.
V této posloupnosti se neopakuje žádný podřetězec délky 2, jde tedy skutečně o posloupnost se SN=1.
Posloupnost v příkladu však nemá optimální délku, takže by za ní žádné body nebyly.

Nudnost knih

task: `borringnes`	points: 100

Už víme, co znamená stupeň nudnosti v případě posloupností čísel. Pokud se ale nad jeho definicí zamyslíme hlouběji, zjistíme, že podobným způsobem lze definovat nudnost i jiných věcí. Knih, filmů, hudby atd.

Např. stupeň nudnosti knihy lze definovat jako číslo k takové, že její text obsahuje dva stejné podřetězce délky k. Tato definice se ukázala být takřka dokonalá, neboť není nic nudnějšího, než číst stejný text dvakrát za sebou.

Nejedna knihovna (včetně té ve Vyšné Boce) se rozhodla zahrnout do svých katalogů i nudnost jednotlivých knih. Bohužel ale zjistily, že stupeň nudnosti není vůbec jednoduché rychle spočítat, a tak požádali o pomoc vás.

Zadání úlohy

Napište program, který dostane vstupní text a co nejrychleji určí stupeň nudnosti tohoto textu. Můžete pro zjednodušení předpokládat, že text se skládá pouze z malých znaků anglické abecedy.

Formát vstupu

Na jediném řádku vstupu je vstupní text. Můžete předpokládat, že délka textu bude vždy nižší než 2^{20}.

Formát výstupu

Na jediném řádku výstupu bude celé nezáporné číslo k, které odpovídá stupni nudnosti zadaného textu, tj. největší číslo takové, že se v textu na různých pozicích nacházejí dva stejné podřetězce délky k. (Tyto podřetězce se mohou i překrývat.)

Příklad

input	output
`mississippi`	`4`

V textu 'mississippi' se dvakrát vyskytuje podřetězec 'issi', který má délku 4 znaky. Žádné delší shodné podřetězce se v něm už nevyskytují.

Pomsta

task: `revenge`	points: 100

Představte si, že za váma příjde dívka vašich snů a poprosí vás, abyste ji napsali řešení domácího úkolu z programování – konkrétně nalezení minimální kostry grafu. Vy nesměle přikývnete a pípnete, že moc rádi. Ona vám dá velkou pusu na čelo a řekne, že jste ten nejúžasnější kluk na celém světě. A pak tu potvoru ještě téhož dne zahlédnete v kavárně s jiným...

Teď jistě cítíte, jak jste naštvaní a plánujete strašlivou pomstu. Že nevíte jakou? To nevadí, my vám rádi poradíme. Napište program, který místo nejmenší kostry v grafu najde kostru, která je druhá nejmenší, ale ostře větší než ta minimální. Toho si jistě nevšimne a vaše pomsta bude dokonána.

Zadání úlohy

Napište program, který na vstupu dostane popis souvislého grafu s ohodnocenými hranami a nalezne kostru, jejíž hodnota je ostře větší než hodnota minimální kostry, ale nejmenší možná. Kostrou grafu G rozumíme podmnožinu hran takovou, že pokud bychom ponechali v grafu jen tyto hrany, zůstal by graf souvislý. Kdybychom ale odebrali libovolnou další, tak by se rozpadl na 2 části. Všimněte si, že kostra bude mít právě $N - 1$ hran, kde N je počet vrcholů. Hodnotou kostry se pak rozumí součet ohodnocení jejích hran.

Formát vstupu

Na prvním řádku budou dvě celá čísla N a M ($1 \leq N \leq 20\,000, 1 \leq M \leq 100\,000$), kde N značí počet vrcholů a M počet hran grafu G. Následuje M řádků, které obsahují vždy trojici čísel a, b a ℓ ($0 \leq a, b < N - 1$, $0 \leq \ell \leq 100\,000$) popisující právě jednu hranu z vrcholu a do vrcholu b délky ℓ. Máte zaručeno, že graf je souvislý.

Máte zaručeno, že pro 80% vstupů bude $N \leq 2\,000$

Formát výstupu

Na jediném řádku výstupu bude hodnota kostry, jejíž velikost je ostře větší než velikost minimální kostry grafu a zároveň je co nejmenší. Máte zaručeno, že taková kostra existuje.

Příklad

input

```
9 13
1 7 3
2 7 2
7 8 5
7 5 2
6 7 1
5 6 3
4 6 2
3 4 7
3 6 5
7 3 4
8 6 4
0 2 1
0 3 8
```

output

```
20
```

Trojúhelníky

task: `triangles`	points: 100

Od dávných dob se koná světoznámý závod plachetnic nazývaný Sejly. Každoročně se jej účastní mnoho zkušených námořníků. Letos se koná výroční stý závod a pořadatelé se rozhodli že chtějí přilákat rekordní počet účastníků. Proto se rozhodli, že po světě vylepí krásné billboardy s cílovou fotografií z konce loňského závodu. Plachetnice jsou takové, že při pohledu na ně vidíte převážně plachtu. Soutěž omezuje barvu plachty jen na zelenou, a tak jí pořadatelé budou při tisku potřebovat opravdu mnoho. Bohužel, její cena v poslední době značně vzrostla a tak by pořadatelé chtěli vědět, kolik billboardů si mohou dovolit. A s tím se obracejí na Vás – mladé a (doufejme) nadějné programátory. Protože jsou pravidla soutěže značně omezující, budete mít úlohu mírně zjednodušenou – všechny plachty jsou rovnoramenné pravoúhlé trojúhelníky, souřadnice jejich vrcholů jsou celočíselné a odvěsny jsou rovnoběžné se souřadnicovými osami.

Zadání úlohy

Máte množinu pravoúhlých rovnoramenných trojúhelníků s celočíselnými souřadnicemi, osami rovnoběžnými se souřadnicovými osami a přeponou jdoucí „zleva shora doprava dolů“. Vaším cílem je spočítat obsah jejich sjednocení.

Formát vstupu

První řádek obsahuje číslo n ($1 \leq n \leq 10\,000$) – počet trojúhelníků. Následujících n řádků obsahuje trojice x, y, d (čísla jsou oddělená mezerou) popisující jednotlivé trojúhelníky. Trojice x, y, d odpovídá trojúhelníku ABC s následujícími souřadnicemi: $A = [x, y]$, $B = [x + d, y]$, $C = [x, y + d]$. Můžete předpokládat, že $0 \leq x, y, d \leq 1\,000\,000$.

Formát výstupu

Bude sestávat z jediného čísla u – obsahu sjednocení daných trojúhelníků, vypsaného s jednou číslicí za desetinnou tečkou. Můžete předpokládat, že $u < 2^{31}$.

Příklad

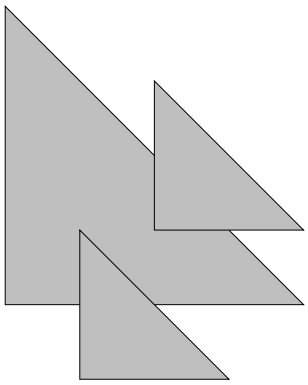
input

```
3
1 1 4
2 0 2
3 2 2
```

output

```
11.0
```

Odpovídající obrázek je zobrazen níže.



Zaskalkulátor

task: `calculator`	points: `100`

Honzík má podivuhodnou a ryze praktickou kalkulačku – takzvaný *zaskalkulátor* (lidově zásobníkový kalkulátor). Obsahuje K paměťových buněk (na začátku prázdných), které jsou číslované 1 až K. Kromě toho umí následující čtyři operace:

„`1`“ Vložit číslo 1 do první prázdné buňky. Pokud žádná taková buňka neexistuje, tak nastane chyba.

„`+`“ Sečíst poslední a předposlední neprázdnou buňku, vyprázdnit je a výsledek uložit do první prázdné buňky. Pokud neexistovali dvě neprázdné buňky, nastane chyba.

„`-`“ Odečíst poslední neprázdnou buňku od předposlední neprázdné, obě vyprázdnit a výsledek uložit do první prázdné buňky. Pokud neexistovali dvě neprázdné buňky, nastane chyba. Také pokud by mělo být výsledkem záporné číslo, nastane chyba.

„`*`“ Vynásobit poslední a předposlední neprázdnou buňku, vyprázdnit je a výsledek uložit do první prázdné buňky. Pokud neexistovali dvě neprázdné buňky, nastane chyba.

Na displeji kalkulátoru je vždy zobrazena hodnota poslední neprázdné buňky. Pokud žádná neprázná buňka neexistuje, je displej prázdný. Každá operace trvá přesně jednu sekundu.

Zadání úlohy

Napište program, který co nejrychleji zjistí, jak nejrychleji lze na displeji kalkulačky s prázdnou pamětí zobrazit zadané číslo N.

Formát vstupu

První řádek vstupu obsahuje dvě celá čísla N a K ($1 \leq N \leq 10^9$, $2 \leq K \leq 100$).

Formát výstupu

První a jediný řádek obsahuje celé nezáporné číslo T, což je minimální doba potřebná k zobrazení čísla N na displeji.

Příklad

input	output
`6 3`	`9`

Čísla 6 lze nejrychleji dosáhnout sledem operací `111++11+*`.

input	output
`11 4`	`15`

Čísla 11 lze nejrychleji dosáhnout sledem operací `1111++11+*11+*1-`.

Sprejzeď

task: `fence`	points: `100`

V Bajtozemi stojí dlouhá zeď, skládající se z N panelů očíslovaných od 1 do N. Bajtozemní vláda by ji ráda ozdobila, a tak najala M sprejerů.

Na začátku se postaví každý sprejer k jednomu panelu a tento panel může buď nastříkat (což trvá právě b sekund) nebo se může přesunout k sousednímu panelu (což trvá přesně a sekund). Případně může zůstat stát a čekat, dokud ho nepolíbí múza.

Vláda však má několik (v našich končinách poněkud nezvyklých) požadavků. Chtějí spotřebovat jen tolik barvy kolik je nezbytně nutné, tedy natřít každou desku jen jednou. Zároveň chtějí mít celou zeď natřenou co nejrychleji.

Zadání úlohy

Pro zadanou délku zdi, počáteční pozice sprejerů, doby za kterou se uskuteční přesun sprejera na sousední panel a doby, ze kterou se natře jeden panel, vypočítejte minimální čas potřebný k natření celé zdi.

Formát vstupu

Na prvním řádku vstupu jsou dvě celá čísla N a M ($1 \leq N, M \leq 100\,000$). Na dalším řádku jsou další dvě celá čísla a a b ($1 \leq a, b \leq 10^6$). Na posledním řádku je M celých čísel $p_1, p_2, \ldots, p_M$, která určují počáteční pozice sprejerů ($1 \leq p_i \leq N$).

Navíc ve 40% vstupů bude $M \leq 2$.

Formát výstupu

Na výstup vypište jedno celé číslo – minimální čas potřebný k natření celé zdi.

Příklad

input

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

output

```
5
```

Zeď lze natřít za 5 minut pokud sprejer 1 natře desku 2, sprejer 2 desku 1 a sprejer 3 desku 3. Sprejer 4 nebude dělat nic.

Permník

task: garden	points: 100

Permský skleník (lidově permník) je obdélníková budova, ve které se pěstují rostliny z dob Permu. Skleník je rozdělen cestičkami na čtverce. V prostředku každého čtverce se nachází jedna rostlina. Nikde jinde se rostliny nenachází. Velikost čtverce závisí na kořenovém systému květiny.

V průběhu roku však cestičky zarůstají travou, což činí pohyb po skleníku velmi nepohodlným. Aby nedocházelo při zahradnických pracích k poškození kořínků, je u každé rostliny cedulka s rozměry jejích kořínků.

Definujme systém Kartézských souřadnic s počátkem v levém dolním rohu skleníku. Osa OX probíhá kolem dolního okraje skleníku a osa OY probíhá levým okrajem skleníku. Na začátku jsou cesty vytyčeny rovnoběžně s osami systému. Jednotky souřadného systému byly zvoleny tak, aby souřadnice čtverců byly celočíselné.

Zadání úlohy

Napište program, který pro zadanou velikost skleníku a souřadnice rostlin určí čtverce, které jim odpovídají.

Formát vstupu

Na první řádce vstupu jsou celá čísla W – délka skleníku, H – šířka skleníku a N – počet rostlin ve skleníku ($1 \leq W, H \leq 10^{12}$, $1 \leq N \leq 2 \cdot 10^5$). Na následujících N řádcích jsou souřadnice rostlin – dvě čísla x_i, y_i ($0 < x_i < W$, $0 < y_i < H$).

Formát výstupu

Vypište N celých čísel – délky stran čtverců odpovídajících rostlinám ze vstupu.

Příklad

input

```
4 6 3
1 1
3 1
2 4
```

output

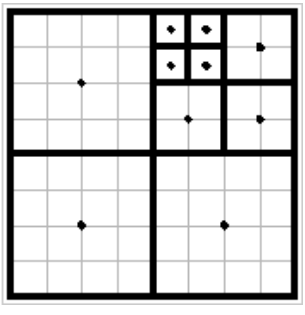
```
2
2
4
```

input

```
8 8 10
4.5 7.5
5.5 7.5
2 6
4.5 6.5
7 7
5 5
6 2
7 5
2 2
5.5 6.5
```

output

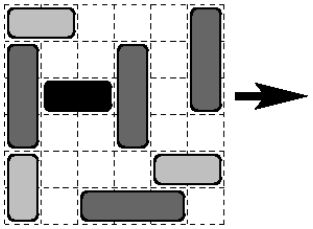
```
1
1
4
1
2
2
4
2
4
1
```



Obrázek ilustrující tento vstup je zobrazen napravo od něj.

Parkování

task: `cars`	points: `100`



Dopravní zácpa je noční můrou všech řidičů. Nikdo nerad trčí v ulicích plných aut, která se pohybují jen velmi pomalu, pokud vůbec nějak. Řidiči z povolání se s dopravní zácpou setkávají velmi často. Poradíte jim, jak se dostat ze zácpy ven?

Malou (i když dost komplikovanou) dopravní zácpu budeme modelovat na hrací desce velké 6×6 políček. Vozidla (osobní a nákladní) jsou rozmístěna na hrací desce na celočíselných souřadnicích, jak je nakresleno na obrázku níže. Oba dva typy vozidel jsou jedno políčko široké. Osobní auta jsou dlouhá dvě a nákladní tři políčka. Každé vozidlo je orientováno buď vodorovně (východo-západně) nebo svisle (severo-jižně).

Vozidla nemohou projíždět skrz sebe, nemohou se otáčet a nemohou ani vyjet z hrací desky. Mohou se pohybovat pouze ve svém směru (vodorovně orientovaná vozidla se nemohou pohybovat svisle a naopak), pokud jim v tom nebrání jiné vozidlo nebo by nevyjela z hrací desky. V jednom tahu se může pohybovat pouze jedno vozidlo, ale může se pohnout o libovolný počet políček (za předpokladu, že má dost místa).

Naším cílem je pohybovat vozidly tak, aby nakonec mohlo označené vodorovně orientované vozidlo (vaše vlastní – na obrázku je označeno černě) vyjet přes pravý (východní) okraj hrací desky a tím se dostat ze zácpy.

Zadání úlohy

Napište program, který nalezne řešení s minimálním možným počtem tahů.

Formát vstupu

Na první řádce je jedno celé číslo n ($1 \leq n \leq 10$), udávající počet vozidel v zácpě. Pak následuje n řádek popisujících jednotlivá vozidla. První znak popisu vozidla je buď znak `h` (vozidlo je orientované vodorovně) nebo znak `v` (vozidlo je orientované svisle). Pak následuje mezera a dvě celá čísla r, c ($1 \leq r, c \leq 6$) oddělená mezerou. Tato čísla udávají souřadnice levého horního políčka zabraného vozidlem. Pak následuje mezera a za ní buď znak `c` nebo `t` určující, zda je vozidlo osobní nebo nákladní. První popsané vozidlo je to, které má vyjet ven ze zácpy, a můžete o něm předpokládat, že je osobní a že je orientováno vodorovně.

Formát výstupu

Na první a jedinou řádku vypište minimální počet tahů potřebných k vyjetí prvního auta z hrací desky přes její pravý okraj. Pokud není možné s prvním autem vyjet, zapište do souboru řetězec „`The car is trapped.`“.

Příklady

input

```
8
h 3 2 c
h 1 1 c
h 5 5 c
h 6 3 t
v 2 1 t
v 5 1 c
v 2 4 t
v 1 6 t
```

output

```
8
```

Vstup odpovídá uvedenému obrázku.

input

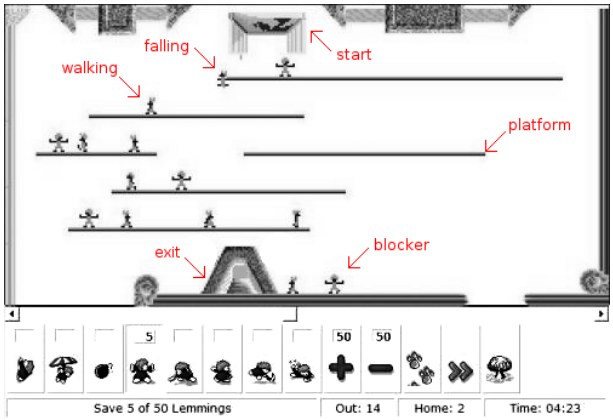
```
3
h 1 1 c
v 1 6 t
v 4 6 t
```

output

```
The car is trapped.
```

Lemmings

task: `lemmings`	points: 100



Lemmings je jednou z nejslavnějších her v historii. Protože už však od jejího vydání nějaký ten pátek uběhl, připomenu pravidla. Vaším úkolem je vodit skupinku lumíků se sebevražednými sklony a zelenými vlasy. Z toho důvodu můžete přinutit některé z nich ke speciálním činnostem, například kopání, zablokování cesty, postavení schodů atd. Řešení takové úlohy je dokazatelně těžký problém a pravděpodobně neexistuje polynomiální řešení. Pro naše účely budeme uvažovat zjednodušenou verzi hry s pouze dvěma činnostmi.

Na začátku hry (v čase $t = 0$) se otevřou vrátka a L lumíků mířících doprava začne vypadávat dolů rychlostí jeden lumík za sekundu. Vaším úkolem bude nasměrovat co největší množství lumíků do cílové pozice. Hra končí ve chvíli, kdy už nezbývají žádní živí lumíci (lumíci jsou buďto mrtví nebo bezpečně dovedeni do cíle).

Každé kolo naší zjednodušené verze lumíků se skládá z

- startovních vrátek, z nichž vypadávají lumíci
- několik horizontálních plošin
- nekonečně široká vodní hladina ve výšce $y = 0$
- cílová pozice, do které byste měli navádět lumíky

Pro naše potřeby budeme předpokládat, že startovní vrátka a cílová pozice jsou body. Lumík dosáhne cíle v okamžiku, kdy se dostane do nějaké malé vzdálenosti ε od něj.

Lumík je na plošině právě tehdy, když pro jeho pozici (x, y) existuje nějaká plošina p taková, že $y = p_y$ a $x \in [p_{start}, p_{end}]$. Živý a pohybující se lumík je v každém okamžiku buďto *pohybující se* nebo *padající* – pokud stojí na plošině, tak se pohybuje, pokud ne tak padá. Oba pohyby se konají konstantní rychlostí 1 jednotka za sekundu.

Hráč může kdykoliv změnit libovolného pohybujícího se lumíka na *blokujícího*. Lumík se v tu chvíli zastaví tam, kde zrovna stál. Blokujícím pak už zůstává navždy a tento stav nelze nijak změnit. Pokud jiný lumík dosáhne pozice blokujícího lumíka, tak se otočí.

Lumík muže zemřít jedním z těchto tří způsobů:

- Jestliže lumík padá vzdálenost větší než H, kde H je maximální bezpečná výška pádu, tak jakmile se dopadne na zem, rozmázne se.
- Pokud lumík spadne do vody, okamžitě se utopí. Lumík však může stát na plošině ve výšce $y = 0$ aniž by se utopil.
- Kdykoliv v průběhu hry může hráč aktivovat nálože, které má u sebe každý lumík. Po deseti sekundách od aktivace všechny bomby explodují. Pokud lumík během těchto deseti vteřin dorazí do cíle, je mu bomba deaktivována a to i v případě, že dorazí přesně v čase exploze. Povšimněte se, že tím hru neměníme, protože aktivaci jsme mohli provést o libovolně malé ε později. V opačném případě po deseti vteřinách exploduje (a to i v případě, že ještě nebyl vpuštěn do hry).

  Odpálení lumíků je obzvláště užitečné pokud máte nějaké blokující lumíky a hra by jinak neskončila.

Vaším úkolem je nalézt maximální počet lumíků, kteří mohou být bezpečně dovedeni do cíle a celkový čas potřebný k dokončení kola.

Vstup

První řádek vstupu obsahuje dvě celá čísla L, H ($1 \leq L \leq 10^9, 1 \leq H \leq 10^9$) – počet lumíků, kteří vypadnou, a maximální výška z níž mohou spadnout. Druhá a třetí řádka obsahuje 2 celá čísla x ($-10^9 \leq x \leq 10^9$) a y ($0 \leq y \leq 10^9$) – pozici startu a cíle.

Na čtvrtém řádku je celé číslo N ($1 \leq N \leq 100\,000$) – počet plošin. Dalších N řádek popisuje plošiny. Na $i+4$. řádku budou 3 celá čísla $xstart_i, xend_i, y_i$ ($0 \leq y_i \leq 10^9$ a $-10^9 \leq xstart_i < xend_i \leq 10^9$).

Můžete předpokládat, že

- žádné dvě plošiny se neprotínají v žádném bodě
- start není na žádné plošině
- lumík padající ze startu nepřistane na okraji plošiny
- cíl je na nějaké plošině
- cíl není na okraji plošiny

Výstup

Výstup se skládá z jedné řádky se dvěma celými čísly L' a T – počet lumíků, kteří mohou být zachráněni a čas potřebný k dokončení kola. Váš výstup by měl být T pokud čas potřebný k dokončení je v intervalu $(T-\varepsilon, T+\varepsilon)$ pro ε libovolně malé. Pokud existuje více možností, zvolte takové, aby L' bylo maximální. Pokud je stále více možností zvolte tu, která minimalizuje T.

Příklad

input

```
1 1000
10 10
25 6
3
0 20 8
0 20 7
20 30 6
```

output

```
1 19
```

Jediný lumík bude padat 2 sekundy, potom půjde vpravo po $10+\varepsilon$ sekund. Poté začne padat, mine druhou plošinu a přistane na třetí plošině.
Čas potřebný pro dokončení kola (zanedbáme-li ε sekund) je $2+10+2+5=19$ sekund.

input

```
3 1
10 9
35 5
4
0 20 8
0 20 6
20 30 7
20 40 5
```

output

```
1 31
```

Bez blokujících lumíků by se lumíci dlouhým pádem rozmázli pádem z bodu $(30+\varepsilon, 7)$. Abychom je zachránili, je třeba použít dva blokující lumíky. Optimální je použít je v $(20+\varepsilon, 7)$ a $(20-\varepsilon, 6)$. Třetí lumík tak může bezpečně dorazit do cíle.

Pošta

task: `post`	points: `100`

Ve městě Bajtník nad Bečvou plánují vylepšení svého poštovního systému. Místo velkého množství starých a rozpadajících se poštovních aut chtějí použít jeden Bajtmobil, který se bude pohybovat cyklicky mezi všemi poštovními pobočkami a rozvážet dopisy k doručení.

Kromě toho je Bajtník známý tím, že všechny ulice v něm jsou jednosměrné, a tím, že se na nich mění maximální povolená rychlost v závislosti na denním období.

Ještě jeden problém zbývá vyřešit – ve které pobočce udělat distribuční centrum, tj. kam se mají vždy dovézt všechny dopisy před tím, než je Bajtmobil začne rozvážet do ostatních poboček. Pro takové distribučním centrum by mělo platit, že Bajtmobil stihne objet všechny pobočky a vrátit se zpět do centra v co nejkratším čase. Můžete předpokládat, že vyložení Bajtmobilu trvá zanedbatelně dlouho.

Zadání úlohy

Napište program, který dostane trasu Bajtmobilu, seznam rychlostních limitů na jednotlivých úsecích jeho trasy a určí pobočku, která se má stát distribučním centrem, a zároveň nejkratší možnou dobu, kterou Bajtmobil potřebuje na projetí trasy a návratu zpět. Platí, že Bajtmobil z distribučního centra vyjíždí vždy v čase nula.

Formát vstupu

Na prvním řádku vstupu je přirozené číslo N ($1 \leq N \leq 100\,000$), což je počet poštovních poboček v Bajtníku. Ty jsou očíslované od jedné do N podle toho, v jakém pořadí jimi projíždí Bajtmobil.

Na následujících $3 \times N$ řádcích se nacházejí popisy jednotlivých úseků trasy. Každý úsek je popsán třemi řádky s následujícím formátem:

- Na první řádku je přirozené číslo d_i, které udává délku cesty ($1 \leq d_i \leq 10^9$), a číslo E_i, což je počet intervalů, na kterých je maximální povolená rychlost konstantní.
- Na druhém řádku jsou celá čísla $t_{i,j}$ ($1 \leq j < E_i$, $0 < t_{i,j} \leq 10^9$), která určují časy, kdy se mění maximální povolená rychlost.
- Na třetím řádku jsou přirozená čísla v_j, které určují maximální povolenou rychlost na úseku v čase $[t_{i,j-1}, t_{i,j})$, kde $1 \leq j \leq E_i$. Předpokládejte, že $t_{i,0} = 0$ a $t_{i,E_i} = \infty$.

Můžete předpokládat, že $\sum E_i \leq 100\,000$.

Formát výstupu

Na prvním řádku výstupu je číslo pošty, která by se měla stát distribučním centrem, a čas, za který Bajtmobil projede celou svojí trasu a vrátí se zpět do centra. Toto číslo musí být uvedeno s přesností 10^{-5}.

Příklady

input

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

output

```
2 2.833333
```

input

```
2
2 1

2
2 1

2
```

output

```
1 2.000000
```