P-III-4 Výlet do Švýcarska

Program: `swiss.pas` / `swiss.c` / `swiss.cpp`
Vstup: `swiss.in`
Výstup: `swiss.out`

Tibor je vášnivý cyklista. Kromě cyklů v grafech má však také rád hory. Rozhodl se, že se o prázdninách vypraví s kamarády na kola do Švýcarska. Chtěli by podniknout čtyřdenní výlet a spát budou vždy v hotelu v některém ze švýcarských měst. Vlakem dojedou do jednoho z měst a odtud vyrazí do strmých kopců a hlubokých údolí. Na internetu si našli seznam jednodenních tras mezi švýcarskými městy, dokonce i s bohatým hodnocením od ostatních cyklistů. Stejně jako mnoho dalších věcí v Tiborově životě, i výlet musí tvořit cyklus. Po čtyřech dnech se tedy musí vrátit zase zpět do města, ze kterého vyrazili. Teď jen naplánovat tu nejlepší trasu. Tibor však zjistil, že to není tak jednoduché a rád by, abyste mu pomohli.

Soutěžní úloha:

Švýcarská města si pro jednoduchost očíslujeme čísly od 1 do N a počet tras nalezených na internetu si označíme M. Každá trasa vždy spojuje dvě různá města a žádné dvě trasy nespojují stejnou dvojici měst. Navíc má každá trasa ohodnocení, což je přirozené číslo menší nebo rovné 256. Všechny trasy lze projet oběma směry a ohodnocení trasy je pro oba směry stejné. Z každého města vede nejvýše 100 tras. Tibor přirozeně nechce jet žádnou z tras dvakrát. Vaším úkolem je najít čtyři na sebe navazující trasy takové, že první z nich začíná ve stejném městě, kde poslední končí, a tyto trasy mají největší součet ohodnocení. Pokud existuje více řešení, můžete vybrat libovolné z nich.

Formát vstupu:

První řádek vstupního souboru `swiss.in` obsahuje přirozená čísla N a M, počet měst $4 \leq N \leq 10\,000$ a počet tras $4 \leq M \leq 1\,000\,000$.

Na následujících M řádcích jsou popsány jednotlivé trasy. Na i-tém řádku jsou přirozená čísla x_i, y_i a h_i, kde $1 \leq x_i \leq N$ a $1 \leq y_i \leq N$ jsou čísla měst, které spojuje i-tá trasa, a $1 \leq h_i \leq 256$ je ohodnocení i-té trasy. Žádné město se nevyskytuje ve více než 100 trasách.

Formát výstupu:

Na první řádek výstupního souboru `swiss.out` vypište nejvyšší součet hodnocení čtyř tras, které splňují podmínky zadání. Na druhý řádek vypište pět čísel navštívených měst v pořadí, v jakém je cyklisté projedou. Dle zadání úlohy musí být první a poslední z těchto čísel stejná.

Pokud žádné takové čtyři trasy neexistují, výstup bude tvořen jedním řádkem se slovem „`NEEXISTUJE`“.

Příklady:

Vstup:

```
6 9
1 2 10
2 5 11
3 1 10
6 3 7
1 4 3
2 6 15
5 3 10
4 5 5
4 6 9
```

Výstup:

```
43
2 6 3 5 2
```

Vstup:

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

Výstup:

```
NEEXISTUJE
```

P-III-5 Záchranná akce

Program: `akce.pas` / `akce.c` / `akce.cpp`
Vstup: `akce.in`
Výstup: `akce.out`

Do lunární stanice na Měsíci narazil neznámý předmět a jedinou šancí na záchranu je vyslat teleportem robota s prosbou o pomoc do některé z dalších stanic na Měsíci. Nejbližší takovou stanicí je stanice Alfa, která je však kromě jednoho skladiště zcela odstíněna proti teleportaci.

Mapa skladiště je obdélníková mřížka čtvercových polí, kde je každé pole buď celé zabráno překážkou (a tedy je neprůchozí) nebo je pro robota celé průchozí. Robot je malý primitivní model, kterého lze ovládat jen tak, že se mu zadá posloupnost jednoduchých příkazů a on ji neustále opakuje. Příkazy jsou čtyři: pro otočení robota vlevo a vpravo (o 90°) a pro pohyb vpřed a vzad (na sousední pole).

Robot se po teleportaci může ocitnout na libovolném políčku na mapě skladiště, kde není překážka, a je natočen směrem na sever. Pokud se robot pokusí pohnout na místo s překážkou, zůstane na místě (zarazil se o ni) a pokračuje následujícím příkazem ze své posloupnosti. Za opuštění skladiště se považuje překročení libovolného okraje mapy.

Soutěžní úloha:

Vaším úkolem je napsat testovací software, který na základě příkazové sekvence pro robota a mapy skladu ve tvaru čtvercové sítě spočítá, z kolika políček lze sklad opustit prováděním zadané posloupnosti příkazů. Připomeňme, že po provedení posledního příkazu této posloupnosti robot začne znovu provádět příkazy posloupnosti od začátku. Navíc pro každé políčko, ze kterého robot skladiště opustí, program určí počet příkazů, které robot provede do opuštění skladiště (do tohoto počtu se započítávají i ty příkazy, které robot nemohl vykonat kvůli překážce v cestě).

Formát vstupu:

Na prvním řádku vstupního souboru `akce.in` je délka L ($1 \leq L \leq 500$) sekvence příkazů pro robota; samotná sekvence příkazů pro robota je pak uvedena na druhém řádku. Druhý řádek tedy obsahuje posloupnost čítající L následujících znaků:

- `L` = otočení robota o 90 stupňů doleva,
- `R` = otočení robota o 90 stupňů doprava,
- `+` = posun robota o jedno políčko vpřed vzhledem ke směru jeho natočení,
- `-` = posun robota o jedno políčko vzad vzhledem ke směru jeho natočení.

Na třetím řádku vstupu jsou pak čísla S a V, oddělená jednou mezerou. S ($1 \leq S \leq 500$) udává šířku mapy, tedy počet políček od západu k východu (vodorovně), a V ($1 \leq V \leq 500$) reprezentuje výšku mapy, tj. počet políček od severu k jihu (svisle).

Na dalších V řádcích následuje samotná mapa. Každý řádek obsahuje S znaků tečka (`.`) nebo mříž (`#`), kde tečky představují volná políčka a mříže překážky. První znak prvního řádku mapy odpovídá jejímu severozápadnímu rohu.

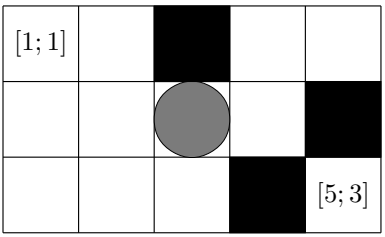
Formát výstupu:

Na první řádek výstupního souboru `akce.out` vypište počet políček, ze kterých robot skladiště opustí. Každý z následujících V řádků pak obsahuje S čísel: j-té číslo na i-tém řádku udává počet robotem vykonaných příkazů do okamžiku, kdy robot opustí skladiště. Pokud z odpovídajícího políčka nelze skladiště opustit nebo se na něm nachází překážka, je toto číslo rovno nule.

Příklad:

Vstup:	*Výstup:*
`8`	`9`
`+R++LL+R`	`1 1 0 1 1`
`5 3`	`9 9 0 4 0`
`..#..`	`17 0 0 0 3`
`....#`	
`...#.`	

Plán skladiště z příkladu:



Z políčka se šedou značkou robot neunikne, pokud dostane posloupnost příkazů uvedenou v příkladu.