

## Zadatak 1. Monopol

Deni uživa u igranju monopola. Ali igra traje predugo – od 5 do 7 sati. Zbog toga, Deni razmišlja da promeni pravila igre. Igrači se u monopolu uglavnom kreću u jednom smeru sa jedne pozicije na drugu, posle toga se vrate na početnu poziciju i počinju novi krug. U novoj verziji, kretanje će se ponovo odvijati sa jedne pozicije na narednu, ali će biti više različitih poteza koje je moguće napraviti na nekoj poziciji. Deni želi da izabere neke poteze, tako da se igrač nikada ne može vratiti na poziciju na kojoj je već bio. Na taj način ona namerava da skрати trajanje igre.

Ona je već počela da pravi novu tablu – izabrala je broj pozicija  $N$  (pozicije su označene od 1 do  $N$ ) i napravila je listu sa  $M$  poteza (svaki potez ima smer i ne postoji potez koji spaja poziciju sa njom samom). Ako je postoji potez koji vodi od pozicije  $i$  do pozicije  $j$ , onda ne postoji potez u obrnutom smeru, tj. ne postoji potez od pozicije  $j$  do pozicije  $i$ , I takođe, ne postoji više poteza koji vode od pozicije  $i$  do pozicije  $j$ . Deni je pomislila da je spremna da počne igru na novoj tabli, ali iznenada je shvatila da uslov koji je želela (da se nije moguće vratiti na poziciju na kojoj ste počeli, ako povlačite poteze po pravilima) nije ispunjen za njenu listu poteza. Ona je prvo htela da izbacila neke od poteza, ali je shvatila da bi to trajalo predugo. Zato je Deni odlučila da samo obrne smerove neke od poteza.

## Zadatak

Vi redovno igrate monopol sa Deni. Zato vi želite da joj pomognete tako što ćete napisati program **monopoly**, koji treba da joj kaže koje poteze da obrne, tako da navedeni uslov važi. Program mora da sadrži funkciju `find_reverse` koja će biti kompajlirana sa komisijskim programom.

## Detalji implementacije

Funkcija `find_reverse` ima sledeći prototip:

```
std::string find_reverse (int N, int M, int connections[][2]);
```

Poziva se samo jednom sa komisijskim programom i tri parametra:  $N$  – broj polja na novoj tabli,  $M$  – broj poteza u Deninoj listi i matrica `connections` koja sadrži  $M$  redova, svaki sadrži dva broja  $x$  i  $y$  – početnu i krajnju poziciju (usmerenog) poteza. Ova funkcija bi trebala da vrati binarnu nisku dužine  $M$  – u istom poretku kao u `connections`, za svaki potez bi trebalo da stavite '1' u nisku, ako potez treba da bude obrnut i '0', u suprotnom. Ako postoji više rešenja, možete da vratite bilo koje.

Sistemu treba poslati fajl **monopoly.cpp**, koji sadrži implementaciju funkcije `find_reverse`. Fajl sme da sadrži neke druge funkcije i bilo kakav kod neophodan za Vaš program, ali **ne sme da sadrži** main funkciju – `main`. Takođe, Vaš program ne bi smeo da čita iz standardnog ulaza, niti da ispisuje na standardni izlaz!

## Ograničenja

- ♣  $3 \leq N \leq 5 \times 10^5$
- ♣  $3 \leq M \leq 1.5 \times 10^6$

### Podzadaci

Podzadatak	Poeni	$N$	$M$	Ostala ograničenja
1	0	-	-	Primeri
2	15	$\leq 7$	$\leq 21$	-
3	40	$\leq 10^3$	$\leq 5 \times 10^3$	-
4	25	$\leq 10^5$	$\leq 5 \times 10^5$	-
5	20	$\leq 5 \times 10^5$	$\leq 1.5 \times 10^6$	-

Da bi dobili poene za neki podzadatak, vaše rešenje mora da prođe sve testove za neki podzadatak.

### Lokalno testiranje

Za lokalno testiranje, dat Vam je fajl **Lgrader.cpp**. We treba da ga smestite u isti folder u kojem se nalazi i Vaš program **monopoly.cpp** i treba da kompajlirate zajedno fajlove **Lgrader.cpp** i **monopoly.cpp**. Na taj način, napravićete program koji proverava korektnost Vaše funkcije. Taj program učitava sa standardnog ulaza:

- u prvoj liniji: dva cela broja – broj pozicija  $N$  i broj poteza  $M$  nove table.
- u poslednjih  $M$  linija: u svakoj liniji se nalaze dva broja  $x$  i  $y$  – početna i kranja pozicija svakog poteza.

Izlaz programa je binarni string koji ste pronašli.

### Primeri lokalnog testiranja

Ulaz	Izlaz	Objašnjenje
6 12 1 2 1 5 2 3 2 5 3 4 3 6 4 1 4 5 5 3 5 6 6 1 6 4	100000101 011	<p>Grane koje su obrnute su: (1,2), (4, 1), (5, 3), (6, 1), (6, 4).</p> <p>Gore navedena slika prikazuje pozicije sa usmerenim vezama <b>sa</b> Deniliste liste, a druga slika, nakon okretanja smeru veza sa odgovarajućim '1' u izlazu. Jasno je da u početku koristeći usmerene veze (1, 2), (2, 3), (3, 4) i (4, 1) možemo da počnemo</p>

		<p>od pozicije 1 i krećemo se ovim usmerenim vezama, vratićemo se na tu poziciju. Može se videti da je <b>Deninuslov</b> zadovoljen na rezultujućoj tabli. Obratite pažnju na to da postoje i druga važeća rešenja:</p> <p>(4, 1), (5, 3), (6, 1), (6, 4)</p> <p>(1, 5), (2, 3), (2, 5), (3, 4), (3, 6), (4, 5), (5, 6)</p>
--	--	---