Brodovi (brodovi)

Vaš prijatelj, koji je nedavno naučio programirati, je napravio igru koja je vrlo slična poznatoj *Battleship* igri. Igra se na matrici koja ima tačno *N* redova i *M* kolona i svaki element predstavlja kvadratić istih dimenzija. Na početku igre, program koji je vaš prijatelj napisao bira jedan horizontalni brod širine *W* koji zapravo predstavlja crticu dimenzija 1 x *W* i stavlja ga na ploču bez rotacije (uvijek je horizontalan), tako da su sve ćelije broda uzastopne i ne prelaze granice matrice (tj. niti jedan dio broda ne može biti van ploče). Vi naravno ne znate gdje je brod tačno postavljen i to morate da pogodite.

Zadatak i dodatna objašnjenja

Na početku svakog pogađanja se poziva funkcija koju vi implementirate:

```
void Napadaj(int N, int M, int W);
```

Značenje parametara *N*, *M* i *W* je opisano u tekstu iznad. Za pogađanje vam je dao na raspolaganje funkciju:

```
int Probaj(int x, int y);
```

Koju pozivate s koordinatama (*x*, *y*) koje su cijeli brojevi iz segmenata [1, *N*] i [1, *M*] respektivno. Cilj vam je da u što manje poziva ove funkcije otkrijete gdje se tačno brod nalazi. Nakon svakog vašeg poziva, ta funkcija će vam vratiti jedan od brojeva iz sljedećeg skupa {-1, 1, 0}, a njihova značenja su sljedeća:

-1 -> funkcija nije pozvana s ispravnim parametrima, ali poziv se ipak broji kao pokušaj;
1 -> na ćeliji s koordinatama (*x*, *y*) se nalazi jedan dio broda;
0 -> na ćeliji s koordinatama (*x*, *y*) se ne nalazi dio broda.

Vaš cilj je da pozivanjem funkcije *Probaj* pronađete čitav brod. Nakon što ta funkcija vrati za tačno *W* različitih vrijednosti koordinata (*x*, *y*) rezultat 1, smatrat će se da ste brod pronašli i vaša funkcija *Napadaj* bi trebala da se završi nakon toga. Kako ne bi bilo isuviše lagano pronaći brod, program je iskoristio to da vi u svakom trenutku znate samo rezultate svakog prethodnog poziva funkcije *Probaj*, pa u cilju povećavanja potrebnog broja pokušaja za pronalazak broad, može uraditi sljedeće:
Ako postoji nova pozicija na koju se brod može postaviti legalno, tj. horizontalno, bez da prelazi granice matrice i da je svaki rezultat svih prethodnih poziva funkcije *Probaj* opet validan (npr. ako je rečeno da se dio broda nalazi u ćeliji (1, 2), a ne nalazi na (1, 1), to mora biti zadovoljeno i novom pozicijom), onda program može izabrati da u bilo kojem trenutku igre premjesti brod na tu poziciju kako bi vam otežao pogađanje.
Naravno, uz dovoljno mnogo pokušaja ćete mu ovu mogućnost oduzeti, jer funkcija *Probaj* vam nikada neće vratiti lažan rezultat. Također, smatrajte da je program vrlo pametan, te će

uvijek na optimalan način premiještati brod. Optimalan način znači da je broj potrebnih poziva funkcije *Probaj* za pronalazak broda uvijek maksimiziran.

Bodovanje i ograničenja

Zadatak će biti testiran na pet podzadatka, od kojih svaki nosi određeni broj bodova i ima sljedeća ograničenja:

Podzadatak 1 (10 bodova): *N* = 1 i 3 <= *M*, *W*<= 10 i *W* <= *M*
Podzadatak 2 (10 bodova): 3 <= *M*, *N* <= 10
Podzadatak 3 (13 bodova): 3 <= *M*, *N* <= 10
Podzadatak 4 (31 bodova): 3 <= *M*, *N* <= 50
Podzadatak 5 (36 bodova): 3 <= *M*, *N* <= 50

U svim podzadacima će važiti *W* <= *M* i *M*, *W* >= 2.

Ako u svakom od testova unutar podzadatka uspijete pronaći brod, onda za taj podzadatak dobijate bodove i to tako da se svaki test pojedinačno boduje s postotkom od 0% do 100% prema formuli: (50%) ^ ((*VS* - *OS*) / *OS*), gdje je *VS* broj poziva koji ste vi napravili, a *OS* optimalan broj poziva s kojim se garantovano mogao pronaći brod, koliko god je program dobro igrao. Nakon tog bodovanja, postoci se množe po testovima i time dobijate bodove za cijeli podzadatak. Ukoliko na nekom testu u podzadatku vaš program koristi mnogo poziva funkcije *Probaj*, onda rezultat tog testa može bitno da smanji ukupan broj bodova koje dobijate na tom podzadatku.