Zadatak **MOSTOVI**:

'Mostovi' su logička zagonetka u kojoj treba izgraditi horizontalne i vertikalne mostove između otoka tako da vrijede slijedeći uvjeti:

- Broj mostova spojen s otokom odgovara zadanom broju koji je pridružen tom otoku.
- Najviše dva mosta spajaju dva ista otoka
- Mostovi se ne smiju međusobno sjeći, niti smiju prelaziti preko otoka

Napišite program koji će izračunati na koliko je načina moguće rješiti zagonetku.

Ulaz:
U prvom retku nalaze se brojevi R i S. ($2 \leq R \leq 9$, $2 \leq S \leq 9$)
U slijedećih R redaka nalazi se opis područja, gdje su prazna polja zadana znakom '.', a otok je zadan znamenkom između '1' i '8' koja predstavlja ukupan broj mostova koji spajaju taj otok. Nijedne dvije znamenke neće biti susjedne horizontalno niti vertikalno.

Izlaz:
Ukupni broj.

ulaz:
```
4 4
1..1
....
....
1..1
```

izlaz:
```
2
```

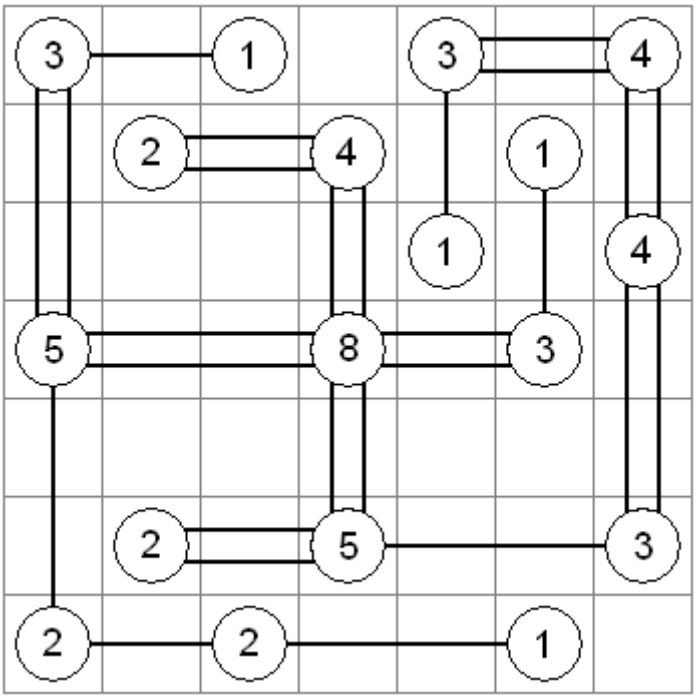
ulaz:
```
5 9
2.2.2.2.2
.........
2.......2
.........
2.2.2.2.2
```

izlaz:
```
6
```

ulaz:
```
7 7
3.1.3.4
.2.4.1.
....1.4
5..8.3.
.......
.2.5..3
2.2..1.
```

izlaz:
```
1
```

Rješenje 3. primjera:



Popunjavamo tablicu po stupcima, a unutar stupca po recima. Za svaki redak pamtimo koliko mostova dolazi slijeva, i za trenutnu ćeliju pamtimo koliko mostova dolazi odozgo. Ti podaci su nam dovoljni da potpuno opišemo stanje.

```
 1: #include <cstdio>
 2: #include <cstring>
 3:
 4: const int pot3[10] = { 1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683 };
 5:
 6: int R, S;
 7: char a[9][10];
 8: int memo[9][9][19683][3];
 9:
10: int rec( int r, int s, int mask, int gore ) {
11:    if( s == S ) return mask == 0;
12:    if( r == R ) return gore ? 0 : rec( 0, s+1, mask, 0 );
13:
14:    int &ret = memo[r][s][mask][gore];
15:    if( ret >= 0 ) return ret;
16:    ret = 0;
17:
18:    int lijevo = mask/pot3[r]%3;
19:
20:    if( a[r][s] == '.' ) {
21:       if( lijevo && gore ) return ret = 0;
22:       return ret = rec( r+1, s, mask, gore );
23:    }
24:
25:    int ostalo = a[r][s] - '0' - lijevo - gore;
26:
27:    for( int dolje = 0; dolje <= 2; ++dolje )
28:       for( int desno = 0; desno <= 2; ++desno )
29:          if( dolje+desno == ostalo )
30:             ret += rec( r+1, s, mask + (desno-lijevo)*pot3[r], dolje );
31:
32:    return ret;
33: }
34:
35: int main( void ) {
36:    scanf( "%d%d", &R, &S );
37:    for( int r = 0; r < R; ++r ) scanf( "%s", a[r] );
38:
39:    memset( memo, -1, sizeof memo );
40:    printf( "%d\n", rec( 0, 0, 0, 0 ) );
41:    return 0;
42: }
```

Rješenje koje traži jedan, bili koji, valjan raspored mostova i iscrtava ga.
‘-‘ predstavlja horizontalni most
‘|’ predstavlja vertikalni most
‘=’ predstavlja 2 horizontalna mosta
‘H’ predstavlja 2 vertikalna mosta

```
 1: #include <cstdio>
 2:
 3: const int pot3[10] = { 1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683 };
 4:
 5: int R, S;
 6: char a[9][10];
 7: char bio[9][9][19683][3];
 8:
 9: void rec( int r, int s, int mask, int gore ) {
10:    if( s == S ) {
11:       if( mask == 0 ) throw 1;
12:       return;
13:    }
14:    if( r == R ) {
15:       if( gore == 0 ) rec( 0, s+1, mask, 0 );
16:       return;
17:    }
18:
19:    if( bio[r][s][mask][gore] ) return;
20:    bio[r][s][mask][gore] = 1;
21:
22:    int lijevo = (mask/pot3[r])%3;
23:
24:    if( a[r][s] == '.' ) {
25:       if( lijevo && gore ) return;
26:
27:       if( lijevo == 1 ) a[r][s] = '-';
28:       if( lijevo == 2 ) a[r][s] = '=';
29:       if( gore == 1 ) a[r][s] = '|';
30:       if( gore == 2 ) a[r][s] = 'H';
31:
32:       rec( r+1, s, mask, gore );
33:
34:       a[r][s] = '.';
35:    } else {
36:       int ostalo = a[r][s] - '0' - lijevo - gore;
37:
38:       for( int dolje = 0; dolje <= 2; ++dolje )
39:          for( int desno = 0; desno <= 2; ++desno )
40:             if( dolje+desno == ostalo )
41:                rec( r+1, s, mask + (desno-lijevo)*pot3[r], dolje );
42:    }
43: }
44:
45: int main( void ) {
46:    scanf( "%d%d", &R, &S );
47:    for( int r = 0; r < R; ++r ) scanf( "%s", a[r] );
48:
49:    try {
50:       rec( 0, 0, 0, 0 );
51:       printf( "Nema rjesenja\n" );
52:    } catch( int nasao ) {
53:       for( int r = 0; r < R; ++r ) printf( "%s\n", a[r] );
54:    }
55:    return 0;
56: }
```