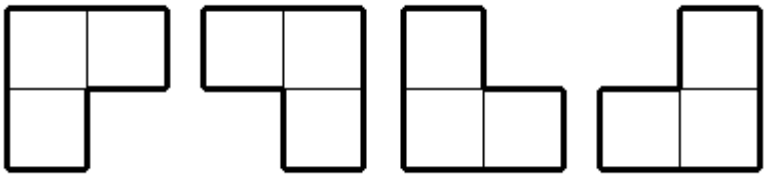
Zadatak **TRI**:

Na koliko načina možemo popločiti sobu dimenzija R x S s pločicama oblika kao na slici? ($2 \leq R \leq 100$, $2 \leq S \leq 10$). Rješenje ispišite modulo 2^{20}.

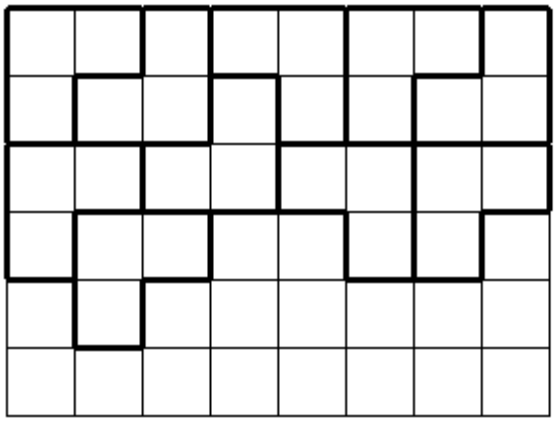


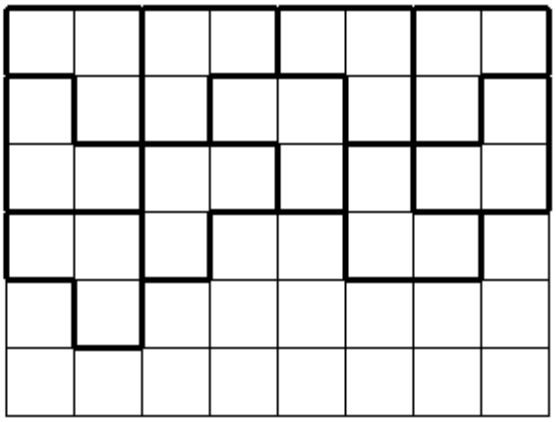
ulaz:	**ulaz:**	**ulaz:**	**ulaz:**
2 3	4 3	9 8	47 9
izlaz:	**izlaz:**	**izlaz:**	**izlaz:**
2	4	204184	6656

Napišimo program koji rekurzivno prebrojava sva moguća rješenja. Idemo po recima, a unutar redka po stupcima, te ako na trenutnom polju nema pločice probamo staviti svaku od četiri kombinacije i pozivamo rekurziju za svaku od mogućih kombinacija.

```
 1: #include <cstdio>
 2:
 3: int R, S;
 4: char a[100][10];
 5:
 6: int rec( int r, int s ) {
 7:    if( s == S ) return rec( r+1, 0 );
 8:    if( r == R ) return 1;
 9:
10:    if( a[r][s] == 1 ) return rec( r, s+1 );
11:
12:    int ret = 0;
13:
14:    int DL = r+1<R && s-1>=0 && a[r+1][s-1]==0; // je li polje dolje-lijevo slobodno
15:    int D  = r+1<R &&      1 && a[r+1][s  ]==0; // je li polje dolje slobodno
16:    int DR = r+1<R &&  s+1<S && a[r+1][s+1]==0; // je li polje dolje-desno slobodno
17:    int R  =     1 &&  s+1<S && a[r  ][s+1]==0; // je li polje desno slobodno
18:
19:    if( D && R ) {                                      //  **
20:       a[r][s] = 1; a[r+1][s  ] = 1; a[r  ][s+1] = 1;  //  *
21:       ret += rec( r, s+1 );
22:       a[r][s] = 0; a[r+1][s  ] = 0; a[r  ][s+1] = 0;
23:    }
24:    if( D && DR ) {                                     //  *
25:       a[r][s] = 1; a[r+1][s  ] = 1; a[r+1][s+1] = 1;  //  **
26:       ret += rec( r, s+1 );
27:       a[r][s] = 0; a[r+1][s  ] = 0; a[r+1][s+1] = 0;
28:    }
29:    if( D && DL ) {                                     //  *
30:       a[r][s] = 1; a[r+1][s  ] = 1; a[r+1][s-1] = 1;  // **
31:       ret += rec( r, s+1 );
32:       a[r][s] = 0; a[r+1][s  ] = 0; a[r+1][s-1] = 0;
33:    }
34:    if( R && DR ) {                                     //  **
35:       a[r][s] = 1; a[r  ][s+1] = 1; a[r+1][s+1] = 1;  //   *
36:       ret += rec( r, s+1 );
37:       a[r][s] = 0; a[r  ][s+1] = 0; a[r+1][s+1] = 0;
38:    }
39:
40:    return ret & 0xFFFFF; // mod 2 na 20
41: }
42:
43: int main( void ) {
44:    scanf( "%d%d", &R, &S );
45:
46:    printf( "%d\n", rec( 0, 0 ) );
47:    return 0;
48: }
```

Međutim gornje rješenje radi presporo za velike primjere jer prebrojava jedno po jedno, svako moguće rješenje. Pogledajmo slijedeće dvije situacije koje dobijemo kada zaustavimo rekurziju s parametrima `r = 3, s = 3` u dvije različite grane.

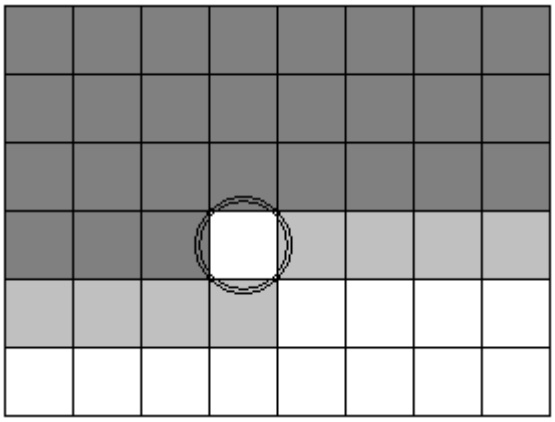




Gornje dvije situacije dobivene su na sasvim različit način, no ukupan broj popločavanja sobe je od tog trenutka na dalje jednak za obje situacije. Točnije, sve situacije u kojima su popločana ista polja kao i u gornjim situacijama, imati će isto rješenje.

Možemo li naći ukupan broj takvih situacija? Ako promotrimo na koji način naša rekurzija popunjava tablicu možemo doći do nekakvih zaključaka. Neka je polje koje trebamo popuniti označeno kružićem na donjoj slici. Vrijedi:

- Tamno-sivo obojana polja sigurno su popločana.
- Bijelo obojana polja su sigurno slobodna, tj. nepopločana.



Kako svijetlo-sivo obojana polja mogu biti bilo što, to je ukupan broj situacija s fiksiranim kružićem jednak 2^S. Ako svijetlo-siva polja prikažemo nekim binarnim brojem (tzv. maskom) gdje 1 označava popločano polje, a 0 slobodno polje, onda sve situacije kod kojih je taj broj jednak imaju isto rješenje.

Svaku situaciju, dakle, možemo prikazati uređenom trojkom (red, stupac, sivo).

Dodajmo jednostavnu memoizaciju u našu rekurziju. Za svako stanje izračunajmo traženi binarni broj i pogledajmo jesmo li već izračunali rješenje za tu situaciju. Ako jesmo, odmah vratimo zapamćeni broj, a ako nismo onda nastavljamo dalje.

```
 1: #include <cstdio>
 2: #include <cstring>
 3:
 4: int R, S;
 5: char a[100][10];
 6:
 7: int memo[100][10][1<<10];
 8:
 9: int rec( int r, int s ) {
10:    if( s == S ) return rec( r+1, 0 );
11:    if( r == R ) return 1;
12:
13:    if( a[r][s] == 1 ) return rec( r, s+1 );
14:
15:    // memoizacija
16:    int mask = 0;
17:    for( int i = s+1; i < S; ++i ) mask = mask*2 + a[r][i];
18:    if( r+1 < R )
19:       for( int i = 0; i <= s; ++i ) mask = mask*2 + a[r+1][i];
20:
21:    int & ret = memo[r][s][mask];
22:    if( ret >= 0 ) return ret;
23:
24:    ret = 0;
25:
26:    int DL = r+1<R && s-1>=0 && a[r+1][s-1]==0; // je li polje dolje-lijevo slobodno
27:    int D  = r+1<R &&      1 && a[r+1][s  ]==0; // je li polje dolje slobodno
28:    int DR = r+1<R &&  s+1<S && a[r+1][s+1]==0; // je li polje dolje-desno slobodno
29:    int R  =     1 &&  s+1<S && a[r  ][s+1]==0; // je li polje desno slobodno
30:
31:    if( D && R ) {                                        //  **
32:       a[r][s] = 1; a[r+1][s  ] = 1; a[r  ][s+1] = 1;  //  *
33:       ret += rec( r, s+1 );
34:       a[r][s] = 0; a[r+1][s  ] = 0; a[r  ][s+1] = 0;
35:    }
36:    if( D && DR ) {                                       //  *
37:       a[r][s] = 1; a[r+1][s  ] = 1; a[r+1][s+1] = 1;  //  **
38:       ret += rec( r, s+1 );
39:       a[r][s] = 0; a[r+1][s  ] = 0; a[r+1][s+1] = 0;
40:    }
41:    if( D && DL ) {                                       //  *
42:       a[r][s] = 1; a[r+1][s  ] = 1; a[r+1][s-1] = 1;  // **
43:       ret += rec( r, s+1 );
44:       a[r][s] = 0; a[r+1][s  ] = 0; a[r+1][s-1] = 0;
45:    }
46:    if( R && DR ) {                                       //  **
47:       a[r][s] = 1; a[r  ][s+1] = 1; a[r+1][s+1] = 1;  //   *
48:       ret += rec( r, s+1 );
49:       a[r][s] = 0; a[r  ][s+1] = 0; a[r+1][s+1] = 0;
50:    }
51:
52:    return ret &= 0xFFFFF; // mod 2 na 20
53: }
54:
55: int main( void ) {
56:    scanf( "%d%d", &R, &S );
57:    memset( memo, -1, sizeof memo );
58:    printf( "%d\n", rec( 0, 0 ) );
59:    return 0;
60: }
```

Ako od početka krenemo pisat rekurziju s memoizacijom možemo neke stvari praktičnije i brže izvest tako da za parametre u rekurziju uzimamo red, stupac i masku za tu situaciju, pa sve računamo samo iz tih podataka.

```
 1: #include <cstdio>
 2: #include <cstring>
 3:
 4: int R, S;
 5: int memo[100][10][1<<10];
 6:
 7: int rec( int r, int s, int mask ) {
 8:    if( s == S ) return rec( r+1, 0, mask );
 9:    if( r == R ) return 1;
10:
11:    int X = mask & 1; // je li ovo polje popunjeno
12:
13:    mask >>= 1;
14:    int &ret = memo[r][s][mask];
15:    if( ret >= 0 ) return ret;
16:
17:    if( X ) return ret = rec( r, s+1, mask );
18:
19:    ret = 0;
20:
21:    int DL = r+1<R && s-1>=0 && ((mask>>(S-2))&1)==0; // je li polje dolje-lijevo slobodno
22:    int D  = r+1<R &&      1 && ((mask>>(S-1))&1)==0; // je li polje dolje slobodno
23:    int DR = r+1<R &&  s+1<S && ((mask>>(S  ))&1)==0; // je li polje dolje-desno slobodno
24:    int R  =     1 &&  s+1<S && ((mask>>(0  ))&1)==0; // je li polje desno slobodno
25:
26:    if( D &&  R ) ret += rec( r, s+1, mask | (1<<(S-1)) | (1<<(0  )) );  //  **
27:                                                                        //  *
28:
29:    if( D && DR ) ret += rec( r, s+1, mask | (1<<(S-1)) | (1<<(S  )) );  //  *
30:                                                                        //  **
31:
32:    if( D && DL ) ret += rec( r, s+1, mask | (1<<(S-1)) | (1<<(S-2)) );  //  *
33:                                                                        // **
34:
35:    if( R && DR ) ret += rec( r, s+1, mask | (1<<(0  )) | (1<<(S  )) );  //  **
36:                                                                        //   *
37:
38:    return ret &= 0xFFFFF; // mod 2 na 20
39: }
40:
41: int main( void ) {
42:    scanf( "%d%d", &R, &S );
43:    memset( memo, -1, sizeof memo );
44:    printf( "%d\n", rec( 0, 0, 0 ) );
45:    return 0;
46: }
```