C - Takahashi Gets Lost Editorial by en_translator

Once Takahashi's current square is fixed, the square on which he crash-landed for that case is uniquely determined, so we count the number of possible crash-land squares instead of the current ones.

For that purpose, one has to check for each square whether Takahashi may have crash-landed on that squares; in other words, whether he can follow the moves according to string T without entering sea if he crash-lands onto that square. This is the essential part of this problem; all that left is just to count conforming squares.

For HW candidates squares, a simulation according to the length-N string T is required, so the time complexity is $O(HWN)$.

The following is sample code for this problem in C++ language.

Copy

```
#include <iostream>
using namespace std;
int h, w, n;
string t, s[505];
int main(void)
{
  cin >> h >> w >> n;
  cin >> t;
  for(int i = 1; i <= h; i++) cin >> s[i];

  int ans = 0;
  for(int i = 1; i <= h; i++){
    for(int j = 0; j < w; j++){
      if(s[i][j] == '#') continue;
      int I = i, J = j; bool ok = true;
      for(auto c : t){
        if(c == 'L') J--;
        if(c == 'R') J++;
        if(c == 'U') I--;
        if(c == 'D') I++;
```

```
        if(s[I][J] == '#'){
          ok = false;
          break;
        }
      }
      if(ok) ans++;
    }
  }
  cout << ans << endl;

  return 0;
}
```