Task Description: Counting Palindromes

You subscribed to a local newspaper that has a weekly puzzle. In the puzzle, you are given a R*C matrix with each cell consist of a single lowercase alphabet. In the matrix there are several words which the characters are constructed :

- Horizontally
  - from left to the right
- Vertically
  - from top to the bottom
- Diagonally
  - from top-left to the bottom-right

You are interested in palindromes, and want to count how many palindromes are there in the matrix. A palindrome is a string which reads the same backward or forward. You think that this puzzle is too hard for you, especially when the size of the matrix is big. Therefore, you decide to write a program that can helps you.

Input

The first line will consist of two integers separated by single space representing R and C respectively.
The next R lines consist of C characters representing the matrix.

Output

You have to output a single integer representing the number of palindromes in the matrix.

Sample Input

```
3 4
abba
sadf
qwer
```

Sample Output

```
15
```

Explanation

Obviously all single-letter words in the matrix are palindromes. There are 12 single-letter words. “bb” and “abba” in the first row are palindromes. “aa” in column-1 row-1 and column-2 row-2 is also a palindrome. Therefore, there are a total of 15 palindromes.

Constraint

R and C will be between 1 and 100 inclusive.

Skeleton program

```
/**
*     Name              :
*     Matric-number     :
*     Plab account      :
```

```
*/
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>
#include <cstdio>
#include <string>
using namespace std;

/**
*     Count how many palindromes
*     Pre-condition  :
*     Post-condition :
*/
int solve(int R, int C, string matrix[100]) {

      return 0;
}

int main(){
    // read input

    // process the input

    // output
    return 0;
}
```

Note:

1. You should develop your program in the subdirectory, ex1 and use the cpp file provided. You should not create new file or rename the file provided.
2. You don't have to use OOP in this sit in lab. You are allowed to add more methods inside each file.
3. If your algorithm is different from the given skeleton, you can develop according to your own algorithm.
4. Please be reminded that the marking scheme is **Input:10%, Output:10%, Programming Style:30% and Correctness: 50%**