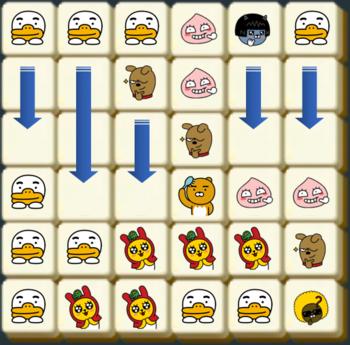
문제 설명

**프렌즈4블록**

블라인드 공채를 통과한 신입 사원 라이언은 신규 게임 개발 업무를 맡게 되었다. 이번에 출시할 게임 제목은 "프렌즈4블록".  
같은 모양의 카카오프렌즈 블록이 2×2 형태로 4개가 붙어있을 경우 사라지면서 점수를 얻는 게임이다.

  
만약 판이 위와 같이 주어질 경우, 라이언이 2×2로 배치된 7개 블록과 콘이 2×2로 배치된 4개 블록이 지워진다. 같은 블록은 여러 2×2에 포함될 수 있으며, 지워지는 조건에 만족하는 2×2 모양이 여러 개 있다면 한꺼번에 지워진다.



블록이 지워진 후에 위에 있는 블록이 아래로 떨어져 빈 공간을 채우게 된다.



만약 빈 공간을 채운 후에 다시 2×2 형태로 같은 모양의 블록이 모이면 다시 지워지고 떨어지고를 반복하게 된다.  


위 초기 배치를 문자로 표시하면 아래와 같다.

TTTANT

RRFACC

RRRFCC

TRRRAA

TTMMMF

TMMTTJ

각 문자는 라이언(R), 무지(M), 어피치(A), 프로도(F), 네오(N), 튜브(T), 제이지(J), 콘(C)을 의미한다

입력으로 블록의 첫 배치가 주어졌을 때, 지워지는 블록은 모두 몇 개인지 판단하는 프로그램을 제작하라.

**입력 형식**

* 입력으로 판의 높이 m, 폭 n과 판의 배치 정보 board가 들어온다.
* 2 <= n, m <= 30
* board는 길이 n인 문자열 m개의 배열로 주어진다. 블록을 나타내는 문자는 대문자 A에서 Z가 사용된다.

**출력 형식**

입력으로 주어진 판 정보를 가지고 몇 개의 블록이 지워질지 출력하라.

**입출력 예제**

| **m** | **n** | **board** | **answer** |
| --- | --- | --- | --- |
| 4 | 5 | ["CCBDE", "AAADE", "AAABF", "CCBBF"] | 14 |
| 6 | 6 | ["TTTANT", "RRFACC", "RRRFCC", "TRRRAA", "TTMMMF", "TMMTTJ"] | 15 |

**예제에 대한 설명**

* 입출력 예제 1의 경우, 첫 번째에는 A 블록 6개가 지워지고, 두 번째에는 B 블록 4개와 C 블록 4개가 지워져, 모두 14개의 블록이 지워진다.
* 입출력 예제 2는 본문 설명에 있는 그림을 옮긴 것이다. 11개와 4개의 블록이 차례로 지워지며, 모두 15개의 블록이 지워진다.

#include "stdafx.h"

#include <iostream>

#include <string>

#include <vector>

using namespace std;

typedef struct

{

int r;

int c;

}element;

bool leftTop(vector<vector<char>> board, int i, int j)

{

bool isSquare = false;

if (board[i][j] == board[i - 1][j] && board[i][j] == board[i - 1][j - 1] && board[i][j] == board[i][j - 1])

{

isSquare = true;

}

return isSquare;

}

bool rightTop(vector<vector<char>> board, int i, int j)

{

bool isSquare = false;

if (board[i][j] == board[i - 1][j] && board[i][j] == board[i - 1][j + 1] && board[i][j] == board[i][j + 1])

{

isSquare = true;

}

return isSquare;

}

bool leftBottom(vector<vector<char>> board, int i, int j)

{

bool isSquare = false;

int a = i;

int b = j;

int c = 0;

if (board[i][j] == board[i][j - 1] && board[i][j] == board[i + 1][j - 1] && board[i][j] == board[i + 1][j])

{

isSquare = true;

}

return isSquare;

}

bool rightBottom(vector<vector<char>> board, int i, int j)

{

bool isSquare = false;

if (board[i][j] == board[i][j + 1] && board[i][j] == board[i + 1][j + 1] && board[i][j] == board[i + 1][j])

{

isSquare = true;

}

return isSquare;

}

void eraseBoard(vector<vector<char>> &board, vector<element> &eraseNum, int& count)

{

count += eraseNum.size();

for (int i = 0; i < eraseNum.size(); i++)

{

element tmp = eraseNum[i];

if (tmp.r == 0) board[tmp.r][tmp.c] = 0; //맨위의 경우 0을 넣는다

else

{

for (int i = 0; i < tmp.r; i++)

{

board[tmp.r - i][tmp.c] = board[tmp.r - i - 1][tmp.c];

board[tmp.r - i - 1][tmp.c] = '0';

}

}

}

eraseNum.clear();

}

int solution(int m, int n, vector<string> board) {

int answer = 0;

vector<vector<char>> tmpBoard(m, vector<char>(n));

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

string tmpStr = board[i];

tmpBoard[i][j] = tmpStr[j];

}

}

vector<element> eraseSpcae;

do

{

// 지우기

if (eraseSpcae.size() > 0)

{

eraseBoard(tmpBoard, eraseSpcae, answer);

}

// 검사

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

scount++;

element tmp;

tmp.r = i;

tmp.c = j;

if (tmpBoard[i][j] == '0')

{

continue;

}

else if (i == 0 && j == 0)

{

//왼쪽 위 모서리 처리

if (rightBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (i == 0 && j == n - 1)

{

//오른쪽 위 모서리 처리

if (leftBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (i == m - 1 && j == 0)

{

//왼쪽 아래 모서리 처리

if (rightTop(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (i == m - 1 && j == n - 1)

{

//오른쪽 아래 모서리 처리

if (leftTop(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (i == 0)

{

//맨 윗줄

if (leftBottom(tmpBoard, i, j) || rightBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (i == m - 1)

{

//맨 아랫줄

if (leftTop(tmpBoard, i, j) || rightTop(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (j == 0)

{

//맨 왼쪽 줄

if (rightTop(tmpBoard, i, j) || rightBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else if (j == n - 1)

{

//맨 오른줄

if (leftTop(tmpBoard, i, j) || leftBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

else

{

//나머지

if (leftTop(tmpBoard, i, j) || leftBottom(tmpBoard, i, j) || rightTop(tmpBoard, i, j) || rightBottom(tmpBoard, i, j))

{

eraseSpcae.push\_back(tmp);

}

}

}

}

cout << endl;

for (int i = 0; i < tmpBoard.size(); i++)

{

for (int j = 0; j < tmpBoard[i].size(); j++)

{

cout << tmpBoard[i][j] << " ";

}

cout << endl;

}

} while (eraseSpcae.size() != 0);

return answer;

}

int main()

{

//m n board answer

//4 5 ["CCBDE", "AAADE", "AAABF", "CCBBF"] 14

//{{"TTTANT"}, {"RRFACC"}, {"RRRFCC"}, {"TRRRAA"}, {"TTMMMF"}, {"TMMTTJ"}}

int m = 6;

int n = 6;

vector<string> board{ { "TTTANT" },{ "RRFACC" },{ "RRRFCC" },{ "TRRRAA" },{ "TTMMMF" },{ "TMMTTJ" } };

int a = solution(m, n, board);

cout << a << endl;

return 0;

}