

Морской бой. 7-10 классы

Во время одного из заседаний АО «Общество гигантских растений» Незнайка рассказал Миге про игру «Морской бой». В игре на прямоугольном поле расставляются один четырёхпалубный корабль, два трёхпалубных корабля, три двухпалубных корабля и четыре однопалубных корабля. Но, рассказывая об игре, Незнайка забыл, что корабли не могут соприкасаться, и в его версии правил корабли могут располагаться на поле как угодно. Корабли одного типа неразличимы, поэтому если поменять местами два корабля одного типа, получится та же самая расстановка.

Однажды, рассматривая тетрадку, в которой он рисовал поля для игры, Незнайка увидел, что границы между кораблями стёрлись и остался только силуэт конфигурации, то есть для каждой клетки поля известно только, был ли там корабль или нет.

Напишите программу, которая по заданному полю, на котором стёрты границы между кораблями, определит количество различных расстановок кораблей. Все клетки кораблей должны размещаться только на занятых клетках поля, и любая занятая клетка поля должна принадлежать какому-либо кораблю. Каждый корабль может размещаться на поле как горизонтально, так и вертикально.

Например, если дана конфигурация

```
.....  
.#####.  
.....
```

то существует 6 способов разместить один четырёхпалубный и два однопалубных корабля как показано ниже.

111123

111132

211113

311112

231111

321111

Формат ввода: В первой строке вводятся целые числа $N_1 : (0 \leq N_1 \leq 4)$, $N_2 : (0 \leq N_2 \leq 3)$, $N_3 : (0 \leq N_3 \leq 2)$, $N_4 : (0 \leq N_4 \leq 1)$, $R : (0 < R \leq 20)$ и $C : (0 < C \leq 20)$. N_1 – это число однопалубных кораблей, N_2 – это число двухпалубных кораблей, N_3 – это число трёхпалубных кораблей, и N_4 – это число четырёхпалубных кораблей. Затем вводятся R строк, каждая из которых состоит из C символов, не считая конца строки. Символ # («решетка») обозначает клетку с кораблём, а символ . («точка») — пустую клетку. В задаваемой конфигурации поля могут использоваться меньше кораблей, чем в полной игре.

Формат вывода: Выводится одно целое число – количество различных конфигураций кораблей с заданным во вводе заполнением клеток поля. Корабли одного типа (например, однопалубные) различимы, поэтому перестановка местами двух однопалубных кораблей даёт новую конфигурацию.

Ввод примера №1:

2 0 0 1 3 8

```
.....  
.#####.  
.....
```

Вывод примера №1:

6

Код возможного решения

```
#include <iostream>
#include <string>
#include <vector>
#include <array>

using namespace std;

int rows;
int cols;
int poscount = 0;
array<int, 4> cnt;
vector<string> table;

void solve(int kind, int curr, int curc)
{
    char ship;
    int k = 0;
    for (; k < 4; ++k) {
        if (cnt[k] > 0) break;
    }
    if (k == 4) {
        ++poscount;
        return;
    }
    ++curc;
    if (curc == cols) {
        curc = 0;
        ++curr;
    }
    if (k != kind) {
        curr = 0;
        curc = 0;
    }
    for (int r = curr; r < rows; ++r) {
        for (int c = curc; c < cols; ++c) {
            curc = 0;
            if (table[r][c] != '#') continue;

            if (k == 0) {
                // 4 deck
                if (c + 3 < cols && table[r][c+1] == '#' && table[r][c+2] == '#'
                    && table[r][c+3] == '#') {
                    ship = '4';
                    table[r][c] = ship;
                    table[r][c+1] = ship;
                    table[r][c+2] = ship;
                    table[r][c+3] = ship;
                    --cnt[k];
                    solve(k, r, c);
                    ++cnt[k];
                    table[r][c+3] = '#';
                    table[r][c+2] = '#';
                    table[r][c+1] = '#';
                }
            }
        }
    }
}
```

```

        table[r][c] = '#';
    }
    if (r + 3 < rows && table[r+1][c] == '#' && table[r+2][c] == '#'
        && table[r+3][c] == '#') {
        ship = '4';
        table[r][c] = ship;
        table[r+1][c] = ship;
        table[r+2][c] = ship;
        table[r+3][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
        table[r+2][c] = '#';
        table[r+3][c] = '#';
    }
} else if (k == 1) {
    // 3 deck
    if (c + 2 < cols && table[r][c+1] == '#' && table[r][c+2] == '#') {
        ship = '3';
        table[r][c] = ship;
        table[r][c+1] = ship;
        table[r][c+2] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c+2] = '#';
        table[r][c+1] = '#';
        table[r][c] = '#';
    }
    if (r + 2 < rows && table[r+1][c] == '#' && table[r+2][c] == '#') {
        ship = '3';
        table[r][c] = ship;
        table[r+1][c] = ship;
        table[r+2][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
        table[r+2][c] = '#';
    }
}
} else if (k == 2) {
    // 2 deck
    if (c + 1 < cols && table[r][c+1] == '#') {
        ship = '2';
        table[r][c] = ship;
        table[r][c+1] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c+1] = '#';
        table[r][c] = '#';
    }
}
if (r + 1 < rows && table[r+1][c] == '#') {

```

```

        ship = '2';
        table[r][c] = ship;
        table[r+1][c] = ship;
        --cnt[k];
        solve(k, r, c);
        ++cnt[k];
        table[r][c] = '#';
        table[r+1][c] = '#';
    }
} else {
    abort();
}
}
}

int main()
{
    int n1, n2, n3, n4;

    cin >> n1 >> n2 >> n3 >> n4;
    cin >> rows >> cols;
    if (n1 < 0 || n1 > 4 || n2 < 0 || n2 > 3 || n3 < 0 || n3 > 2 || n4 < 0 || n4 > 1)
        abort();
    if (rows < 1 || rows > 20 || cols < 1 || cols > 20) abort();

    string buf;
    getline(cin, buf);

    int hashcount = 0;
    for (int i = 0; i < rows; ++i) {
        getline(cin, buf);
        if (cin.eof()) abort();
        if (int(buf.length()) != cols) abort();
        for (char c : buf) hashcount += (c == '#');
        table.push_back(buf);
    }
    getline(cin, buf);
    if (!cin.eof()) abort();
    if (hashcount != n1 + 2 * n2 + 3 * n3 + 4 * n4) abort();

    cnt = array<int, 4>{ n4, n3, n2, 0 };
    solve(-1, 0, -1);

    int mult = 1;
    for (int i = 2; i <= n1; ++i) mult *= i;
    for (int i = 2; i <= n2; ++i) mult *= i;
    for (int i = 2; i <= n3; ++i) mult *= i;
    for (int i = 2; i <= n4; ++i) mult *= i;

    cout << (poscount * mult) << endl;
}

```