Worksheet - 5

Student Name: ANIKET KUMAR **UID:**20BCS5306
Branch: BE-CSE **Semester:** 5th
Subject Name: Competitive Coding **Section/Group:** 20BCS_WM-703/B

Task-1: Journey-to-the-moon

https://www.hackerrank.com/challenges/journey-to-the-moon/problem?isFullScreen=true

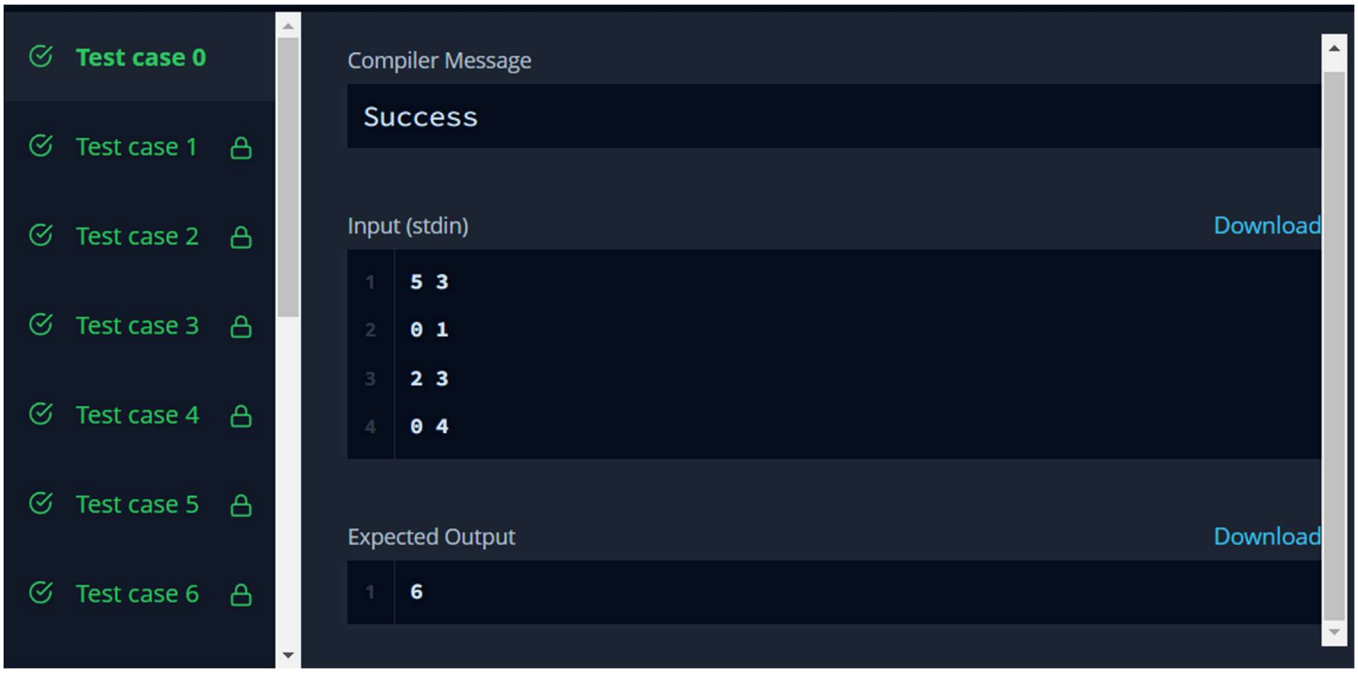
Code:

```
#include <cstdio>
#include <vector>
#include <queue>
#include <algorithm>

using namespace std;

bool visited[100001] = {0};

struct node {
  vector<long long> neighbour;
};

long long bfs(long long, node *);

int main() {
  long long n,m;
  scanf("%lld %lld", &n, &m);
  node nodelist[n];
  long long a,b;
  while(m--) {
    scanf("%lld %lld", &a, &b);
```

```
    nodelist[a].neighbour.push_back(b);
    nodelist[b].neighbour.push_back(a);
  }

  long long connected = 0;
  long long total = 0;
  long long temp = 0;
  std::vector<int> count;
  for (long long i = 0; i < n; ++i) {
    if(!visited[i]) {
      temp = bfs(i, nodelist);
      count.push_back( temp );
      total += temp;
      connected++;
    }
  }
  long long answer = (total * (total - 1)) / 2;
  for (int i = 0; i < connected; ++i) {
    answer -= (count[i] * (count[i] - 1)) / 2;
  }
  printf("%lld", answer);

}

long long bfs(long long nod, node *nodelist) {
  int count = 0;
  queue<long long> Q;
  Q.push(nod);
  long long n;
  while(!Q.empty()) {
    n = Q.front();
    Q.pop();
    if(visited[n]) {
      continue;
    }
    visited[n] = true;
    count++;
    for (vector<long long>::iterator itr = nodelist[n].neighbour.begin(); itr !=
nodelist[n].neighbour.end(); ++itr) {
```

```
    if(!visited[*itr]) {
      Q.push(*itr);
    }
  }
 }
 return count;
}
```

Hacker Rank Test Case / Output:



Task-2: Frog-in-maze

https://www.hackerrank.com/challenges/frog-in-maze/problem?isFullScreen=true

Code:

```
#include<cstdio>

char M[25][25];
int T[25][25][2];
double P[2][25][25];

const int D[4][2] = {{-1,0}, {1, 0}, {0,-1}, {0,1}};
int h,w,t;

void calc(int in, int out) {
    for(int x=0;x<w;x++)
        for(int y=0;y<h;y++) {
            if(M[y][x] == '*' || M[y][x] == '#')
                P[out][y][x] = 0.0;
            if(M[y][x] == '%')
                P[out][y][x] = 1.0;
            if(M[y][x] == 'O' || M[y][x] == 'A') {
                int count = 0; double suma = 0.0;
                int px=x, py=y;
                if(T[y][x][0] != -1) {px = T[y][x][0]; py = T[y][x][1];}

                for(int i=0;i<4;i++) {
                    int x2 = px+D[i][0], y2 = py + D[i][1];
                    if(x2 < 0 || x2 >= w || y2 < 0 || y2 >= h)continue;
                    if(M[y2][x2] == '#')continue;
                    suma += P[in][y2][x2];
                    count++;
                }
                if(count == 0)
                    P[out][y][x] = 0.0;
```

```
            else P[out][y][x] = suma / count;
        }
    }
}

double get_ans(int p) {
    for(int i=0;i<h;i++)
        for(int j=0;j<w;j++)
            if(M[i][j] == 'A')
                return P[p%2][i][j];
    return -1.0;
}

int main() {
    scanf("%d%d%d", &h, &w, &t);

    for(int i=0;i<h;i++)
        scanf("%s", M[i]);

    for(int i=0;i<h;i++)
        for(int j=0;j<w;j++)
            T[i][j][0] = T[i][j][1] = -1;

    for(int i=0;i<t;i++){
        int x0, y0, x1, y1;
        scanf("%d%d%d%d", &y0, &x0, &y1, &x1);
        x0--;y0--;x1--;y1--;
        T[y0][x0][0] = x1;
        T[y0][x0][1] = y1;
        T[y1][x1][0] = x0;
        T[y1][x1][1] = y0;
    }

    const int limit = 80000;

    for(int i=0;i<limit;i++) {
        calc(i%2, (i+1)%2);
    }
    printf("%lf\n", get_ans(limit));
```

```
}
```

Hacker Rank Test Case / Output: