Worksheet - 4

**Student Name:** ANIKET KUMAR **UID:**20BCS5306

**Branch:** BE-CSE **Semester:** 5th

**Subject Name:** Competitive Coding **Section/Group:** 20BCS\_WM-703/B

# Task-1: Find the merge point of two joined linked list

# <https://www.hackerrank.com/challenges/find-the-merge-point-of-two-joined-linked-lists/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--;#include <bits/stdc++.h>

using namespace std;

class SinglyLinkedListNode {

public:

int data;

SinglyLinkedListNode \*next;

SinglyLinkedListNode(int node\_data) {

this->data = node\_data;

this->next = nullptr;

}

};

class SinglyLinkedList {

public:

SinglyLinkedListNode \*head;

SinglyLinkedListNode \*tail;

SinglyLinkedList() {

this->head = nullptr;

this->tail = nullptr;

}

void insert\_node(int node\_data) {

SinglyLinkedListNode\* node = new SinglyLinkedListNode(node\_data);

if (!this->head) {

this->head = node;

} else {

this->tail->next = node;

}

this->tail = node;

}

};

void print\_singly\_linked\_list(SinglyLinkedListNode\* node, string sep, ofstream& fout) {

while (node) {

fout << node->data;

node = node->next;

if (node) {

fout << sep;

}

}

}

void free\_singly\_linked\_list(SinglyLinkedListNode\* node) {

while (node) {

SinglyLinkedListNode\* temp = node;

node = node->next;

free(temp);

}

}

int findMergeNode(SinglyLinkedListNode\* headA, SinglyLinkedListNode\* headB) {

while(headA){

SinglyLinkedListNode \*tmp = headA->next;

headA->next = NULL;

headA = tmp;

}

while(headB){

if(headB->next == NULL){

return headB->data;

}

headB = headB->next;

}

return 0;

}

int main()

{

ofstream fout(getenv("OUTPUT\_PATH"));

int tests;

cin >> tests;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

for (int tests\_itr = 0; tests\_itr < tests; tests\_itr++) {

int index;

cin >> index;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

SinglyLinkedList\* llist1 = new SinglyLinkedList();

int llist1\_count;

cin >> llist1\_count;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

for (int i = 0; i < llist1\_count; i++) {

int llist1\_item;

cin >> llist1\_item;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

llist1->insert\_node(llist1\_item);

}

SinglyLinkedList\* llist2 = new SinglyLinkedList();

int llist2\_count;

cin >> llist2\_count;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

for (int i = 0; i < llist2\_count; i++) {

int llist2\_item;

cin >> llist2\_item;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

llist2->insert\_node(llist2\_item);

}

SinglyLinkedListNode\* ptr1 = llist1->head;

SinglyLinkedListNode\* ptr2 = llist2->head;

for (int i = 0; i < llist1\_count; i++) {

if (i < index) {

ptr1 = ptr1->next;

}

}

for (int i = 0; i < llist2\_count; i++) {

if (i != llist2\_count-1) {

ptr2 = ptr2->next;

}

}

ptr2->next = ptr1;

int result = findMergeNode(llist1->head, llist2->head);

fout << result << "\n";

}

fout.close();

return 0;

}

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++) {

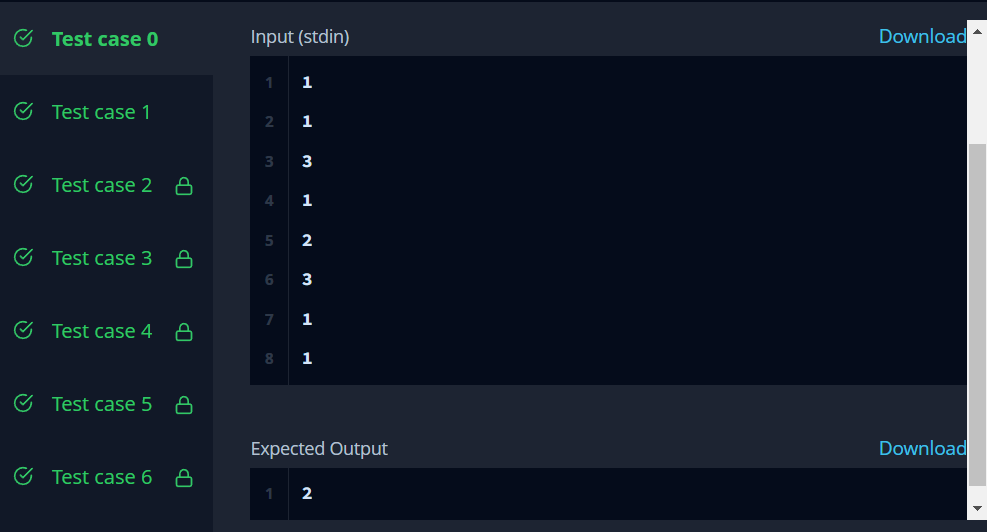
calc(i%2, (i+1)%2);

}

printf("%lf\n", get\_ans(limit));

}

# Hacker Rank Test Case / Output:

****

# Task-2: Whether a linked list contains a cycle

# <https://www.hackerrank.com/challenges/detect-whether-a-linked-list-contains-a-cycle/problem?isFullScreen=true>

**Code:**

#include <bits/stdc++.h>

using namespace std;

class SinglyLinkedListNode {

public:

int data;

SinglyLinkedListNode \*next;

SinglyLinkedListNode(int node\_data) {

this->data = node\_data;

this->next = nullptr;

}

};

class SinglyLinkedList {

public:

SinglyLinkedListNode \*head;

SinglyLinkedListNode \*tail;

SinglyLinkedList() {

this->head = nullptr;

this->tail = nullptr;

}

void insert\_node(int node\_data) {

SinglyLinkedListNode\* node = new SinglyLinkedListNode(node\_data);

if (!this->head) {

this->head = node;

} else {

this->tail->next = node;

}

this->tail = node;

}

};

void print\_singly\_linked\_list(SinglyLinkedListNode\* node, string sep, ofstream& fout) {

while (node) {

fout << node->data;

node = node->next;

if (node) {

fout << sep;

}

}

}

void free\_singly\_linked\_list(SinglyLinkedListNode\* node) {

while (node) {

SinglyLinkedListNode\* temp = node;

node = node->next;

free(temp);

}

}

bool has\_cycle(SinglyLinkedListNode\* head) {

SinglyLinkedListNode\* cur1 = head;

SinglyLinkedListNode\* cur2 = head;

int result = 0;

while (cur1 && cur2)

{

cur1 = cur1->next;

cur2 = cur2->next;

if (cur2)

{

cur2 = cur2->next;

}

if (cur1 == cur2)

{

result = 1;

break;

}

}

return result;

}

int main()

{

ofstream fout(getenv("OUTPUT\_PATH"));

int tests;

cin >> tests;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

for (int tests\_itr = 0; tests\_itr < tests; tests\_itr++) {

int index;

cin >> index;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

SinglyLinkedList\* llist = new SinglyLinkedList();

int llist\_count;

cin >> llist\_count;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

for (int i = 0; i < llist\_count; i++) {

int llist\_item;

cin >> llist\_item;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

llist->insert\_node(llist\_item);

}

SinglyLinkedListNode\* extra = new SinglyLinkedListNode(-1);

SinglyLinkedListNode\* temp = llist->head;

for (int i = 0; i < llist\_count; i++) {

if (i == index) {

extra = temp;

}

if (i != llist\_count-1) {

temp = temp->next;

}

}

temp->next = extra;

bool result = has\_cycle(llist->head);

fout << result << "\n";

}

fout.close();

return 0;

}

**Hacker Rank Test Case / Output:**

****