Lab 5 23k0634

Q1

#include<iostream>

using namespace std;

#include <ctime>    // For time()

// Assume you are developing a program to simulate a game where players take turns to guess a

// number between 1 and 100. The program generates a random number between 1 and 100 at

// the start of the game, and the player who guesses the correct number wins the game. If a player

// guesses incorrectly, the program tells them whether their guess was too high or too low, and

// the turn passes to the next player. Solve it using Direct Recursion?

void game(int target, int player)

{

    int guess;

    if (player == 1){

        cout<<"player 1: ";

        cin>>guess;

        if (guess == target){

            cout<<"player 1 wins!";

            return;

        }

        else{

            if (guess < target)

                cout<<"Guess is low"<<endl;

            else{

                cout<<"Guess is high"<<endl;

            }

            return game(target,2);

        }

    }

    if (player == 2){

        cout<<"player 2: ";

        cin>>guess;

        if (guess == target){

            cout<<"player 2 wins!";

            return;

        }

        else{

            if (guess < target)

                cout<<"Guess is low"<<endl;

            else{

                cout<<"Guess is high"<<endl;

            }

            return game(target, 1);

        }

    }

}

int main()

{

    srand(time(0));

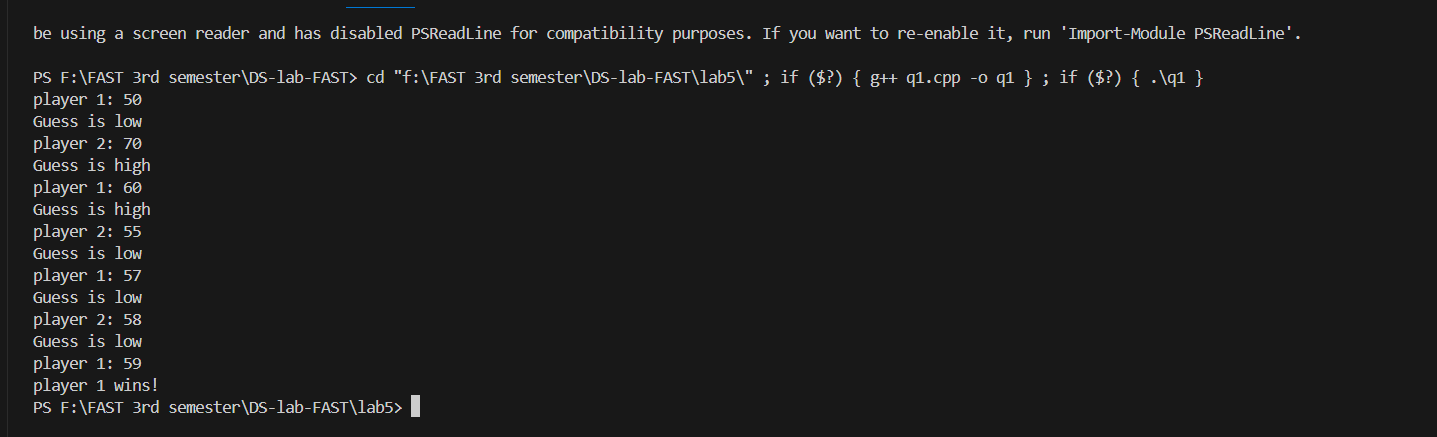
    int max = 100; int min = 1;

    int target = (rand() % (max - min + 1)) + min;

    game(target,1);

    return 0;

}



Q2

#include<iostream>

using namespace std;

// Write a C++ function to find the length of a singly linked list using tail recursion.

class node{

    public:

    int data;

    node\* next;

    node(int data){

        this-> data = data;

        this-> next = NULL;

    }

};

void insert\_end(node\* head, int data)

{

    node\* current = head;

    node\* insertNode = new node(data);

    while(current->next != NULL){

        current = current->next;

    }

    current->next = insertNode;

}

void displayLL(node\* &head)

{

    node\* current = head;

    while (current!=NULL){

        cout<<current->data<<endl;

        current = current->next;

    }

}

int sizeLL(node\* current,int size)

{

    if (current->next == NULL){

        return size;

    }

    return sizeLL(current->next,size+1);

}

int main()

{

    //making LL

    node\* node1 = new node(10);

    node\* head = NULL;

    head = node1;

    node\* current = head;

    insert\_end(head,20);

    insert\_end(head,30);

    insert\_end(head,40);

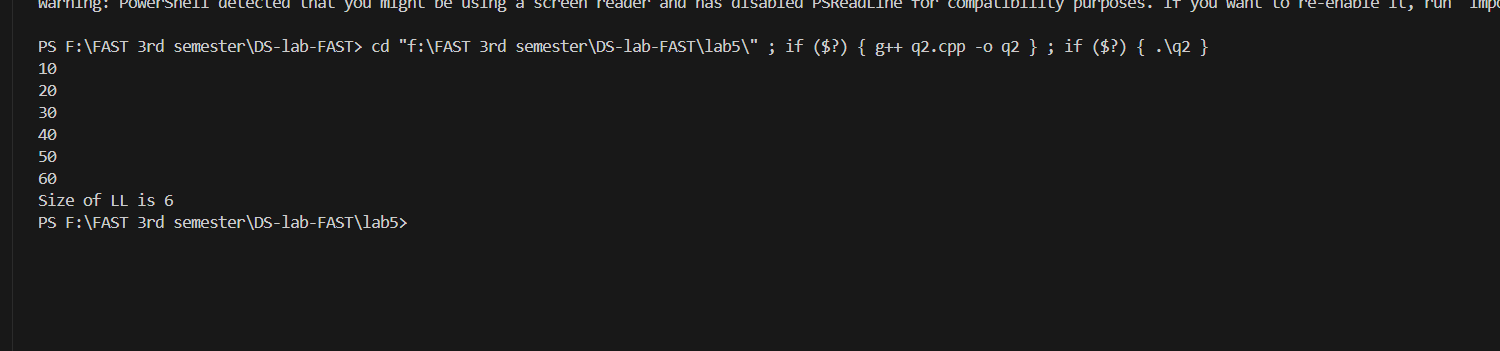
    insert\_end(head,50);

    insert\_end(head,60);

    displayLL(head);

    cout<<"Size of LL is "<<sizeLL(current,1);

}



Q3

#include<iostream>

using namespace std;

// Create a C++ program to search for a value in a singly linked list, using non-tail recursion?

class node{

    public:

    int data;

    node\* next;

    node(int data){

        this-> data = data;

        this-> next = NULL;

    }

};

void insert\_end(node\* head, int data)

{

    node\* current = head;

    node\* insertNode = new node(data);

    while(current->next != NULL){

        current = current->next;

    }

    current->next = insertNode;

}

void displayLL(node\* &head)

{

    node\* current = head;

    while (current!=NULL){

        cout<<current->data<<endl;

        current = current->next;

    }

}

void searchLL(node\* current, int target)

{

    if (current->data == target){

        cout<<"Target found!"<<endl;

        return;

    }

    if (current->next == NULL){

        cout<<"Target not found!"<<endl;

        return;

    }

    searchLL(current->next,target);

    cout<<"Seaching done on "<<current->data<<endl; //backtracked it to show which elemets have been traveresed and checked

}

int main()

{

    //making LL

    node\* node1 = new node(10);

    node\* head = NULL;

    head = node1;

    node\* current = head;

    insert\_end(head,20);

    insert\_end(head,30);

    insert\_end(head,40);

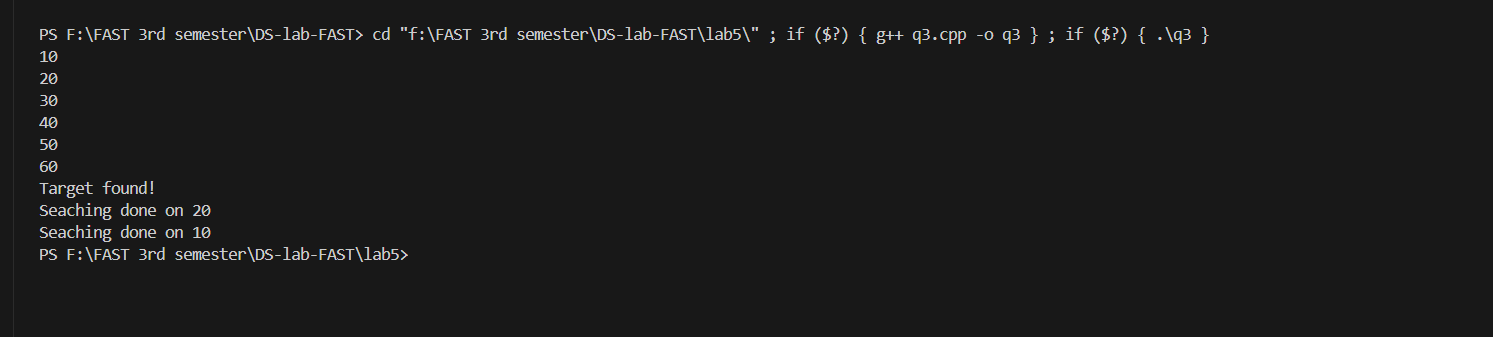
    insert\_end(head,50);

    insert\_end(head,60);

    displayLL(head);

    searchLL(current,30);

}



Q4

#include <iostream>

using namespace std;

int recursiveArraySum(int\* arr[], int sizes[], int dim){

    int sum = 0;

    for(int i = 0; i < dim; i++){

        for(int j = 0; j < sizes[i]; j++){

            sum += arr[i][j];

        }

    }

    return sum;

}

int main(){

    int arr1[] = {1, 2, 3};

    int arr2[] = {4, 5};

    int arr3[] = {6, 7, 8, 9};

    int\* jaggedArray[] = {arr1, arr2, arr3};

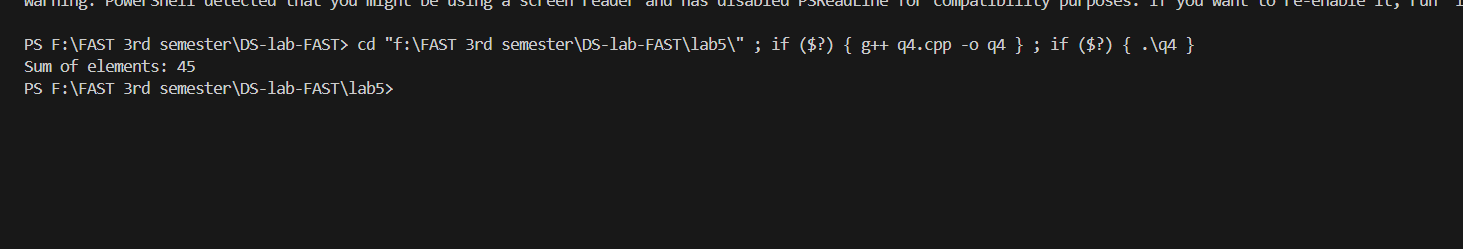
    int sizes[] = {3, 2, 4};

    int sum = recursiveArraySum(jaggedArray, sizes, 3);

    cout<<"Sum of elements: "<<sum<<endl;

    return 0;

}



Q5

#include<iostream>

using namespace std;

// rat/lion maze question

bool isSafe(int arr[5][5], int x, int y,int n)

{

    if (x<n && y<n && arr[x][y] == 1){

        return true;

    }

    return false;

}

bool lionMaze(int arr[5][5], int x, int y, int n, int solArr[5][5])

{

    if ((x==(n-1)) && (y==(n-1))){

        solArr[x][y] = 1;

        return true;

    }

    if (isSafe(arr,x,y,n)){

        solArr[x][y] = 1;

        if (lionMaze(arr,x+1,y,n,solArr)){

            return true;

        }

        if (lionMaze(arr,x,y+1,n,solArr)){

            return true;

        }

        solArr[x][y] = 0;

        return false;

    }

    return false;

}

void display(int solArr[5][5])

{

    for (int i=0;i<5;i++){

        for (int j=0;j<5;j++){

            cout<<solArr[i][j]<<" ";

        }

        cout<<endl;

    }

}

int main()

{

    int arr[5][5] = {{1,0,1,0,1},

                     {1,1,1,1,1},

                     {0,1,0,1,1},

                     {1,0,0,1,1},

                     {1,1,1,0,1}};

    int solArr[5][5] = {{0,0,0,0,0},

                        {0,0,0,0,0},

                        {0,0,0,0,0},

                        {0,0,0,0,0},

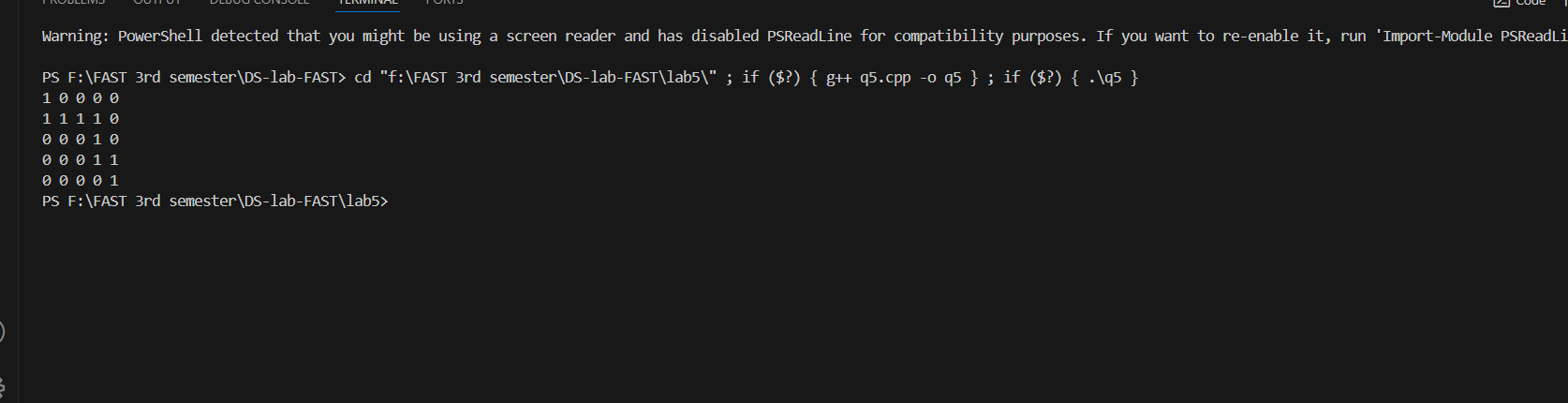
                        {0,0,0,0,0}};

    lionMaze(arr,0,0,5,solArr);

    display(solArr);

    return 0;

}



Q6

#include<iostream>

using namespace std;

// queen/flag qs

bool isSafe(int arr[],int x, int y)

{

    for (int i=0;i<x;i++){

        if (arr[i] == y || abs(arr[i]-y)==abs(i-x)){

            return false;

        }

    }

    return true;

}

bool flags(int arr[],int n, int x)

{

    if(x==n){

        return true;

    }

    for (int y=0;y<n;y++){

        if (isSafe(arr,x,y)){

            arr[x] = y;

            if (flags(arr,n,x+1)){

                return true;

            }

            arr[x] = -1;

        }

    }

    return false;

}

int print(int arr[], int n) {

    int count=0;

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            if (arr[i] == j) {

                cout << " F ";

                count++;

            } else {

                cout << " 0 ";

            }

        }

        cout << endl;

    }

    return count;

}

int main()

{

    int arr[4] = {-1,-1,-1,-1};

    flags(arr,4,0);

    int count = print(arr,4);

    cout<<"Total numbr of flags: "<<count;

    return 0;

}

