**DS LAB # 5 K23-0607 Sept 28,2024**

**Question # 1**

#include <iostream>

#include <cstdlib>

#include <ctime>

using  namespace std;

void solve(int player,int Random){

    int guess;

    cout << "Player : " << player << " Enter Number To Guess : " << endl;

    cin >> guess;

    if( guess == Random){

        cout << " You Guess the right Number : " << guess << endl;

        return;

    }

    if( guess < Random){

        cout << "The Number You Guess is Less than Number : " << endl;

    }

    else{

        cout << "The Number You Guess is Greater than Number : " << endl;

    }

    if(player == 1) player = 2;

    else            player = 1;

    return solve(player,Random);

}

int main() {

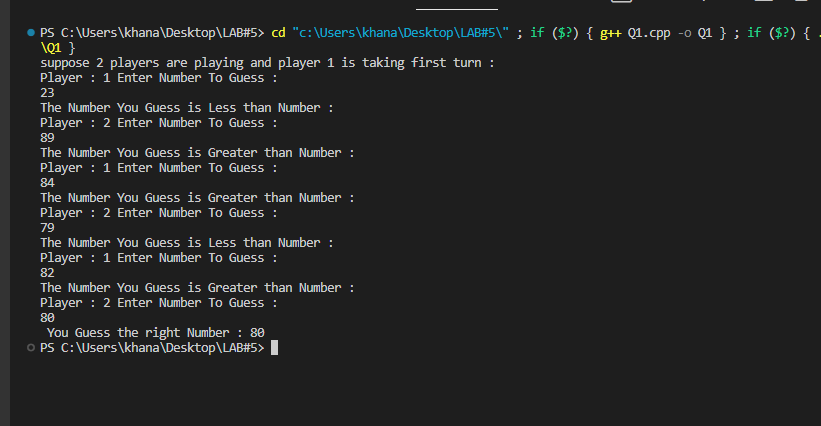
    srand(time(nullptr));

    int Random = rand() % 100 + 1;

    cout << "suppose 2 players are playing and player 1 is taking first turn : " << endl;

    solve(1,Random);

}



**Question # 2**

#include <iostream>

using namespace std;

class Node

{

public:

    int val;

    Node \*next;

    Node(int data)

    {

        val = data;

        next = nullptr;

    }

};

int solve(  Node\*head, int count){

    if( !head ) return count;

    return solve(head->next,count+1);

}

void print(Node \*head)

{

    while (head)

    {

        cout << head->val << " -> ";

        head = head->next;

    }

    cout << "NULL" << endl;

}

int main()

{

    Node \*head = new Node(17);

    head->next = new Node(15);

    head->next->next = new Node(8);

    head->next->next->next = new Node(12);

    head->next->next->next->next = new Node(10);

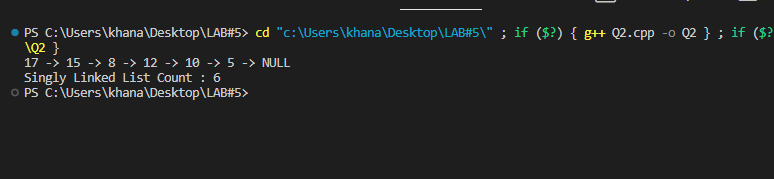
    head->next->next->next->next->next = new Node(5);

    int count = 0;

    print(head);

    cout <<  "Singly Linked List Count : " <<  solve(head,count) << endl;

}



**Question # 3**

#include <iostream>

using namespace std;

class Node

{

public:

    int val;

    Node \*next;

    Node(int data)

    {

        val = data;

        next = nullptr;

    }

};

bool solve(Node\* head,int target){

    if( !head ) return false;

    if( head->val == target ) return true;

    return solve(head->next,target);

}

void print(Node \*head)

{

    while (head)

    {

        cout << head->val << " -> ";

        head = head->next;

    }

    cout << "NULL" << endl;

}

int main()

{

    Node \*head = new Node(17);

    head->next = new Node(15);

    head->next->next = new Node(8);

    head->next->next->next = new Node(12);

    head->next->next->next->next = new Node(10);

    head->next->next->next->next->next = new Node(5);

    int target = 10;

    print(head);

    bool found = solve( head,target );

    if( found ){

        cout << "Target Found : " << target << endl;

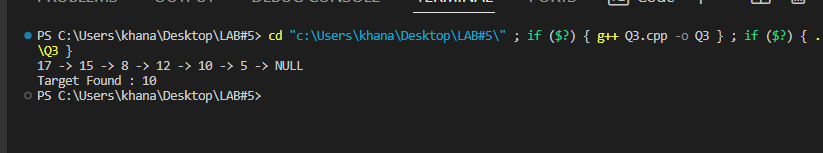
    }

    else{

        cout <<"Target Not Found : " << endl;

    }

}



**Question # 4**

#include <iostream>

using namespace std;

int recursiveArray(int \*\*arr, int size, int dim) {

    int sum = 0;

    if (dim <= 0) {

        return 0;

    } else {

            for(int j=0; j<size; j++){

               sum += arr[dim-1][j];

            }

        sum += recursiveArray(arr, size, dim - 1);

    }

    return sum;

}

int main() {

    int n = 5;

    int size = 3;

    int \*arr[n];

    for (int i = 0; i < n; i++) {

        arr[i] = new int[size];

        for (int j = 0; j < size; j++) {

            arr[i][j] = 1;

        }

    }

    int sum = recursiveArray(arr, size, n);

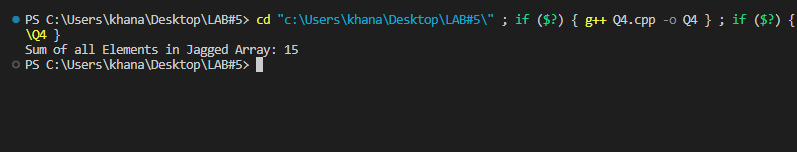
    cout << "Sum of all Elements in Jagged Array: " << sum << endl;

    for (int i = 0; i < n; i++) {

        delete[] arr[i];

    }

}



**Question # 5**

#include <iostream>

using namespace std;

bool Check(int arr[5][5], bool isVisited[5][5], int destination, int srcx, int srcy)

{

    if (srcx >= 0 && srcy >= 0 && srcx <= destination && srcy <= destination && isVisited[srcx][srcy] == 0 && arr[srcx][srcy] == 1)

    {

        return true;

    }

    return false;

}

void print( bool ans[5][5], int size){

    for(int i=0; i<size; i++){

        for(int j=0; j<size; j++){

            cout << ans[i][j] << " ";

        }

        cout << endl;

    }

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void solve(int arr[5][5], bool isVisited[5][5], bool ans[5][5], int destination, int srcx, int srcy)

{

    if ( srcx == destination  && srcy == destination  )

    {

        ans[srcx][srcy] = 1;

        print(ans,5);

        ans[srcx][srcy] = 0;

        return;

    }

    isVisited[srcx][srcy] = true;

    ans[srcx][srcy] = true;

    int newx = srcx - 1;

    int newy = srcy;

    if (Check(arr, isVisited, destination , newx, newy))

    {

        solve(arr, isVisited, ans, destination, newx, newy);

    }

    newx = srcx + 1;

    newy = srcy;

    if (Check(arr, isVisited, destination , newx, newy))

    {

        solve(arr, isVisited, ans, destination, newx, newy);

    }

    newx = srcx;

    newy = srcy + 1;

    if (Check(arr, isVisited, destination , newx, newy))

    {

        solve(arr, isVisited, ans, destination, newx, newy);

    }

    newx = srcx;

    newy = srcy - 1;

    if (Check(arr, isVisited, destination , newx, newy))

    {

        solve(arr, isVisited, ans, destination, newx, newy);

    }

    isVisited[srcx][srcy] = false;

    ans[srcx][srcy] = false;

}

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}};

    bool isVisited[5][5] = {{false}};

    bool ans[5][5]  = {{false}};

    int destination = 4;

    int srcx = 0;

    int srcy = 0;

    solve(arr,isVisited,ans,destination,srcx,srcy);

 cout << "Total 4 paths are found";

    }

**A screenshot of a computer

Description automatically generated**

**Question # 6**

#include <iostream>

using namespace std;

bool isValid(bool arr[4][4], int r, int c, int size)

{

    int x = r;

    int y = c;

    while (y >= 0)

    {

        if (arr[x][y] == true)

        {

            return false;

        }

        y--;

    }

    x = r;

    y = c;

    while (x >= 0 && y >= 0)

    {

        if (arr[x][y] == true)

        {

            return false;

        }

        x--;

        y--;

    }

    x = r;

    y = c;

    while (x < size && y >= 0)

    {

        if (arr[x][y] == true)

        {

            return false;

        }

        x++;

        y--;

    }

    return true;

}

void solve(bool arr[4][4], int col, int size, int &maxi, int curr)

{

    if (col == size)

    {

        maxi = max(maxi, curr);

        return;

    }

    for (int row = 0; row < size; row++)

    {

        if (isValid(arr, row, col, size))

        {

            arr[row][col] = true;

            solve(arr, col + 1, size, maxi, curr + 1);

            arr[row][col] = false;

        }

    }

}

int main()

{

    bool arr[4][4] = {{false}};

    int count = 0;

    solve(arr, 0, 4, count, 0);

    cout << "Max Num of Flags  Place in a 4x4 Grid: " << count << endl;

}

A screen shot of a computer code

Description automatically generated

**END**