*Muhammad Mufeez*

*BCS-3J*

***DS Lab 05***

**Q1:**

*Code:*

*#include <bits/stdc++.h>*

*using namespace std;*

*class Node*

*{*

*public:*

*int data;*

*Node \*head;*

*Node \*next;*

*Node()*

*{*

*this->next = NULL;*

*this->head = NULL;*

*}*

*Node(int data, Node \*next = NULL)*

*{*

*this->data = data;*

*this->next = next;*

*this->head = NULL;*

*}*

*void insertElement(Node \*temp, int val)*

*{*

*if (temp == NULL)*

*{*

*temp = new Node(val);*

*head = temp;*

*return;*

*}*

*if (temp->next == NULL)*

*{*

*temp->next = new Node(val);*

*return;*

*}*

*insertElement(temp->next, val);*

*}*

*void displayReverse(Node \*temp)*

*{*

*if (temp->next == NULL)*

*{*

*cout << temp->data << " ";*

*return;*

*}*

*else if (temp == NULL)*

*return;*

*displayReverse(temp->next);*

*cout << temp->data << " ";*

*}*

*};*

*int main()*

*{*

*Node \*list = new Node(20);*

*list->insertElement(list, 56);*

*list->insertElement(list, 20);*

*list->insertElement(list, 19);*

*list->insertElement(list, 15);*

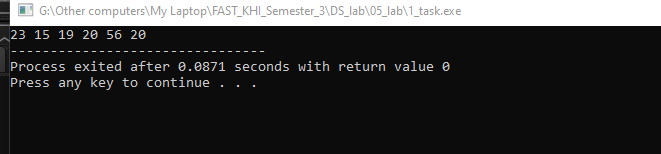
*list->insertElement(list, 23);*

*list->displayReverse(list);*

*return 0;*

*}*

*Output:*

**

**Q2:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

string *guessNumber*(int n)

{

int guess;

cout << "Enter your guess Mr player 1:";

cin >> guess;

*if* (n == guess)

*return* "Player 1";

*else* *if* (guess > n)

cout << "Your guess was too high" << endl;

*else*

cout << "Your guess was too low" << endl;

cout << "Enter your guess Mr player 2:";

cin >> guess;

*if* (n == guess)

*return* "Player 2";

*else* *if* (guess > n)

cout << "Your guess was too high" << endl;

*else*

cout << "Your guess was too low" << endl;

*return* *guessNumber*(n);

}

int *main*()

{

*srand*(static\_cast<unsigned>(*time*(0)));

int n = *rand*() % 100 + 1;

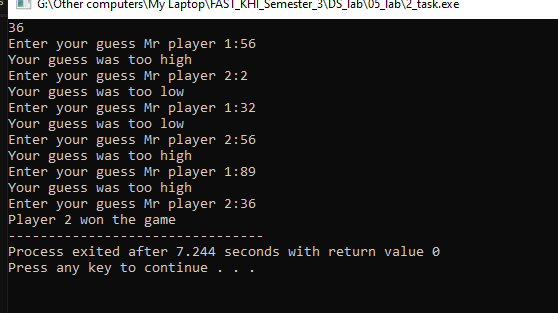
cout << n << endl;

cout << *guessNumber*(n) << " won the game";

*return* 0;

}

*Output:*



**Q3:**

*Code:*

#*include* <iostream>

using namespace std;

class Node

{

public:

int data;

Node *\**next;

*Node*()

{

this->next = NULL;

data = 0;

}

*Node*(int data)

{

this->data = data;

this->next = NULL;

}

void *insertElement*(Node *\**temp, int val)

{

*if* (temp->next == NULL)

{

temp->next = new *Node*(val);

*return*;

}

*insertElement*(temp->next, val);

}

void *displayLLUsingRecursion*(Node *\**mover)

{

*if* (mover == NULL)

{

*return*;

}

cout << mover->data << " ";

*if* (mover->next == NULL)

{

*return*;

}

*displayLLUsingRecursion*(mover->next);

}

int *findLength*(Node *\**temp, int count = 0)

{

*if* (temp == NULL)

{

*return* count;

}

*return* *findLength*(temp->next, count + 1);

}

};

int *main*()

{

Node *\**list = new *Node*(23);

list->*insertElement*(list, 56);

list->*insertElement*(list, 20);

list->*insertElement*(list, 19);

list->*insertElement*(list, 15);

list->*insertElement*(list, 23);

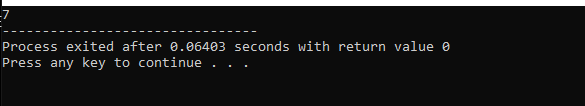
list->*insertElement*(list, 495);

cout << list->*findLength*(list);

*return* 0;

}

*Output:*



**Q4:**

*Code:*

#*include* <iostream>

using namespace std;

class Node

{

public:

int data;

Node *\**next;

*Node*()

{

this->next = NULL;

data = 0;

}

*Node*(int data)

{

this->data = data;

this->next = NULL;

}

void *insertElement*(Node *\**temp, int val)

{

*if* (temp->next == NULL)

{

temp->next = new *Node*(val);

*return*;

}

*insertElement*(temp->next, val);

}

void *displayLLUsingRecursion*(Node *\**mover)

{

*if* (mover == NULL)

{

*return*;

}

cout << mover->data << " ";

*if* (mover->next == NULL)

{

*return*;

}

*displayLLUsingRecursion*(mover->next);

}

Node *\*findVal*(Node *\**temp, int n)

{

*if* (!temp)

{

cout << "Value Not Found" << endl;

*return* temp;

}

*if* (temp->data == n)

{

cout << "Value Found" << endl;

*return* temp;

}

*return* *findVal*(temp->next, n);

}

};

int *main*()

{

Node *\**list = new *Node*(23);

list->*insertElement*(list, 56);

list->*insertElement*(list, 20);

list->*insertElement*(list, 19);

list->*insertElement*(list, 15);

list->*insertElement*(list, 23);

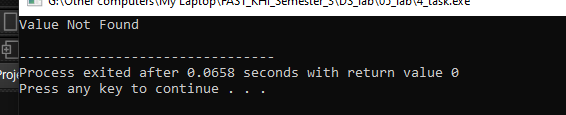
list->*insertElement*(list, 495);

Node *\**search = list->*findVal*(list, 95);

*return* 0;

}

*Output:*



**Q5:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

int *sum*(int i, int cols, int *\**arr)

{

*if* (cols == i)

*return* 0;

*return* arr[i] + *sum*(i + 1, cols, arr);

}

int rows;

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

{

int sumOfRow = *sum*(0, sizes[dim], arr[dim]);

*if* (dim == rows)

{

*return* sumOfRow;

}

*return* sumOfRow + *recursiveArraySum*(arr, sizes, dim + 1);

}

int *main*()

{

cin >> rows;

int *\*\**array = new int \*[rows];

int *\**sizes = new int[rows];

*for* (int i = 0; i < rows; i++)

{

cin >> sizes[i];

array[i] = new int[sizes[i]];

}

*for* (int i = 0; i < rows; i++)

{

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

{

cin >> array[i][j];

}

}

cout << "Total Sum: " << *recursiveArraySum*(array, sizes, 0) << endl;

*return* 0;

}

/\*

*input values*

*4*

*4 3 1 5*

*2 3 5 6*

*3 2 1*

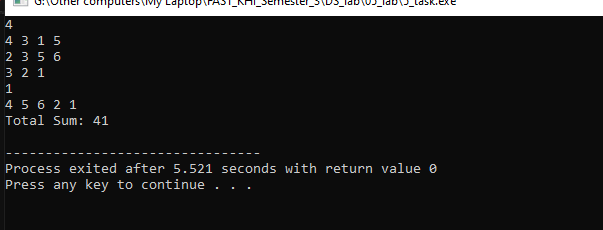
*1*

*4 5 6 2 1*

*output : 41*

\*/

*Output:*



**Q6:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

void *displaysolved*(int array[5][5], int N)

{

*for* (int i = 0; i < N; i++)

    {

*for* (int j = 0; j < N; ++j)

        {

            cout *<<* array[i][j] *<<* " ";

        }

        cout *<<* *endl*;

    }

}

bool *isValid*(int x, int y, int maze[5][5], int sol[5][5], int n)

{

*if* (x >= 0 && x < n && y >= 0 && y < n && maze[x][y] == 1 && !sol[x][y])

    {

        sol[x][y] = 1;

*return* true;

    }

*return* false;

}

bool *solve*(int x, int y, int maze[5][5], int sol[5][5], int n, int xarr[], int yarr[])

{

*if* (x == n - 1 && y == n - 1)

    {

        sol[n - 1][n - 1] = 1;

*displaysolved*(sol, n);

*return* true;

    }

*if* (*isValid*(x, y, maze, sol, n))

    {

*for* (int i = 0; i < 5; ++i)

        {

*if* (*solve*(x + xarr[i], y + yarr[i], maze, sol, n, xarr, yarr))

            {

*return* true;

            }

        }

        sol[x][y] = 0;

*return* false;

    }

*return* false;

}

int *main*()

{

    int n = 5;

    int maze[5][5] = {

        {1, 0, 1, 0, 1},

        {1, 1, 1, 1, 1},

        {0, 1, 0, 1, 1},

        {1, 0, 0, 1, 1},

        {0, 0, 0, 0, 1}};

    int sol[5][5];

    int xarr[n] = {1, 0, 0, -1};

    int yarr[n] = {0, -1, 1, 0};

*for* (int i = 0; i < 5; i++)

    {

*for* (int j = 0; j < 5; ++j)

        {

            sol[i][j] = 0;

        }

    }

*displaysolved*(sol, n);

    cout *<<* *endl*;

*if* (maze[0][0] != 1)

    {

        cout *<<* "Not possible";

*return* 0;

    }

*if* (!*solve*(0, 0, maze, sol, n, xarr, yarr))

    {

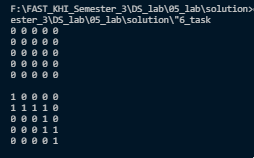
        cout *<<* "Not possible";

    }

*return* 0;

}

*Output:*



**Q7:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

void *displaysolved*(char *\*\**array, int N)

{

*for* (int i = 0; i < N; i++)

    {

*for* (int j = 0; j < N; ++j)

        {

            cout << array[i][j] << " ";

        }

        cout << endl;

    }

}

void *solve*(int col, char *\*\**board, char *\*\**solved, int N, vector<int> row, vector<int> lowerDiagonal, vector<int> upperDiagonal)

{

*if* (col == N)

    {

*for* (int i = 0; i < N; ++i)

        {

*for* (int j = 0; j < N; ++j)

            {

                solved[i][j] = board[i][j];

            }

        }

*return*;

    }

*for* (int i = 0; i < N; ++i)

    {

*if* (

            row[i] == 0 && lowerDiagonal[i + col] == 0 && upperDiagonal[N - 1 + col - i] == 0)

        {

            board[i][col] = 'Q';

            row[i] = 1;

            lowerDiagonal[i + col] = 1;

            upperDiagonal[N - 1 + col - i] = 1;

*solve*(col + 1, board, solved, N, row, lowerDiagonal, upperDiagonal);

            board[i][col] = '-';

            row[i] = 0;

            lowerDiagonal[i + col] = 0;

            upperDiagonal[N - 1 + col - i] = 0;

        }

    }

}

int *main*()

{

    int N;

    cin >> N;

    char *\*\**board = new char \*[N];

    char *\*\**solved = new char \*[N];

*for* (int i = 0; i < N; ++i)

    {

        board[i] = new char[N];

        solved[i] = new char[N];

    }

*for* (int i = 0; i < N; ++i)

    {

*for* (int j = 0; j < N; ++j)

        {

            board[i][j] = '-';

            solved[i][j] = '-';

        }

    }

    vector<int> *row*(N, 0);

    vector<int> *lowerDiagonal*(N \* 2 - 1, 0);

    vector<int> *upperDiagonal*(N \* 2 - 1, 0);

*solve*(0, board, solved, N, row, lowerDiagonal, upperDiagonal);

*displaysolved*(solved, N);

*for* (int i = 0; i < N; ++i)

    {

        delete[] board[i];

        delete[] solved[i];

    }

    delete[] board;

    delete[] solved;

*return* 0;

}

*Output:*

