Data Structures (Lab)

Assignment-1 (Fundamentals)

1. WAP to check whether a given matrix is Magic Square or not?

#include <iostream>

using namespace std;

bool isMagicSquare(int mat[][3], int n)

{

    // Your implementation here

    for (int i = 0; i < n; i++)

    {

        int sum = 0;

        for (int j = 0; j < n; j++)

        {

            sum += mat[i][j];

        }

        if (sum != 15)

            return false;

    }

    for (int i = 0; i < n; i++)

    {

        int sum = 0;

        for (int j = 0; j < n; j++)

        {

            sum += mat[j][i];

        }

        if (sum != 15)

            return false;

    }

    int sum = 0;

    for (int j = 0; j < n; j++)

    {

        sum += mat[j][j];

    }

    if (sum != 15)

        return false;

    return true;

}

int main()

{

    int mat[3][3] = {

        {2, 7, 6},

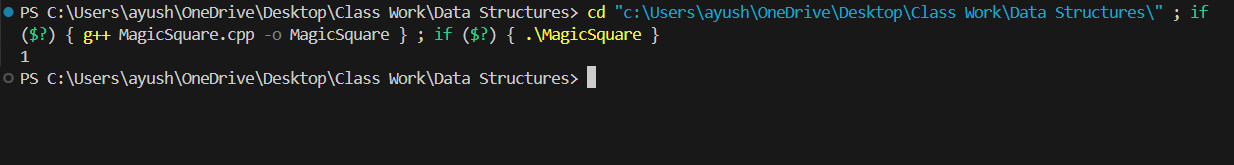
        {9, 5, 1},

        {4, 3, 8}};

    cout << isMagicSquare(mat, 3) ? "true": "false";

    return 0;

}



1. WAP to implement Call by Value and Call by Reference mechanisms.

#include <iostream>

// Function that uses call by value

void callByValue(int a) {

    a = 20; // Change local copy of the argument

    std::cout << "Inside callByValue: a = " << a << std::endl;

}

// Function that uses call by reference

void callByReference(int &b) {

    b = 30; // Change the actual argument

    std::cout << "Inside callByReference: b = " << b << std::endl;

}

int main() {

    int x = 10, y = 10;

    std::cout << "Before callByValue: x = " << x << std::endl;

    callByValue(x); // Pass by value

    std::cout << "After callByValue: x = " << x << std::endl;

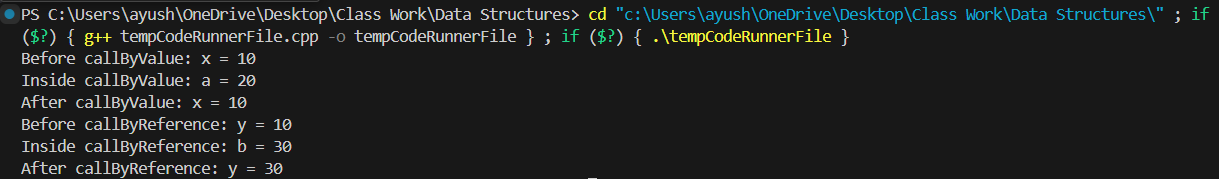
    std::cout << "Before callByReference: y = " << y << std::endl;

    callByReference(y); // Pass by reference

    std::cout << "After callByReference: y = " << y << std::endl;

    return 0;

}



1. WAP to implement three different ways to swap two variables without using a third variable.

#include <iostream>

void swap1(int &a, int &b) {

    a = a + b;

    b = a - b;

    a = a - b;

}

void swap2(int &a, int &b) {

    a = a \* b;

    b = a / b;

    a = a / b;

}

void swap3(int &a, int &b) {

    a = a ^ b;

    b = a ^ b;

    a = a ^ b;

}

int main() {

    int a = 5, b = 10;

    std::cout << "Original: " << a << ", " << b << std::endl;

    swap1(a, b);

    std::cout << "After swap1: " << a << ", " << b << std::endl;

    swap2(a, b);

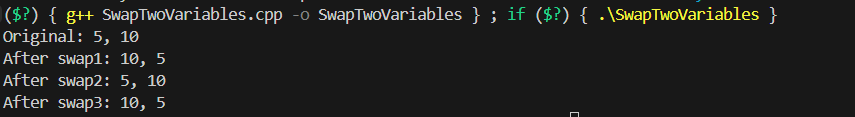
    std::cout << "After swap2: " << a << ", " << b << std::endl;

    swap3(a, b);

    std::cout << "After swap3: " << a << ", " << b << std::endl;

    return 0;

}



1. WAP to implement the following programs using recursion.

Factorial

#include <iostream>

using namespace std;

int factorial(int n)

{

    if (n == 0 || n == 1)

        return 1;

    return n \* factorial(n - 1);

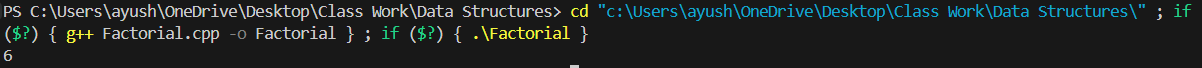
}

int main()

{

    cout << factorial(3);

}



Fibonacci Series

Greatest Common Divisor

#include <iostream>

using namespace std;

int gcd(int x, int y){

    if (y == 0)

        return x;

    return gcd(y, x % y);

}

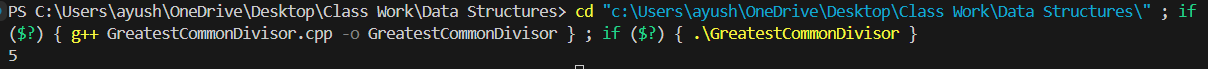
int main(int argc, char const \*argv[])

{

    cout << gcd(15, 10);

    return 0;

}



Linear Search

#include <iostream>

using namespace std;

int linear\_search(int \*a, int size, int n){

    for (int i = 0; i < size; i++){

        if (a[i] == n)

        {

            return i;

        }

    }

    return -1;

}

int main(int argc, char const \*argv[])

{

    int arr[] = {1, 2, 3, 4, 5};

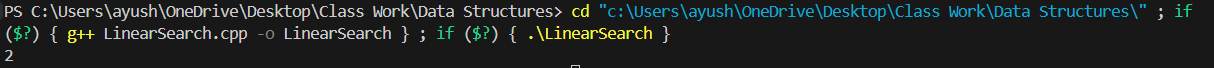
    int num = 3;

    int result = linear\_search(arr, sizeof(arr)/sizeof(arr[0]), num);

    cout << result << endl;

    return 0;

}



Binary Search

#include <iostream>

using namespace std;

int binarySearch(int \*a, int n, int num){

    int left = 0;

    int right = n - 1;

    while (left <= right) {

        int mid = left + (right - left) / 2;

        if (a[mid] == num)

            return mid;

        else if (a[mid] < num)

            left = mid + 1;

        else

            right = mid - 1;

    }

    return -1; // Element not found in the array

}

int main(int argc, char const \*argv[])

{

    int arr[] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

    int n = sizeof(arr) / sizeof(arr[0]);

    int num = 13;

    int result = binarySearch(arr, n, num);

    cout << result << endl;

    return 0;

}



Tower of Hanoi

#include <iostream>

using namespace std;

void towerOfHanoi(double n, double x, double y, double z) {

    if (n == 1) {

        cout << "Move disk 1 from rod A to rod C" << endl;

        return;

    }

    towerOfHanoi(n - 1, 'A', 'C', 'B');

    cout << "Move disk " << n << " from rod A to rod C" << endl;

    towerOfHanoi(n - 1, 'B', 'A', 'C');

}

int main(int argc, char const \*argv[])

{

    double n;

    cout << "Enter the number of disks: ";

    cin >> n;

    towerOfHanoi(n, 'A', 'B', 'C');

    return 0;

}

