****

**LAB EXERCISE 1**

Name: Jayannthan P T

Dept: CSE ‘A’

Roll No.: 205001049

1. [Collatz Conjecture] Consider the following algorithm to generate a sequence of numbers. Start with an integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when n = 1. For example, the following sequence of numbers will be generated for n = 22: 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1. Count the sequence length.

**Code:**

#include <bits/stdc++.h>

using **namespace** std;

**bool** arraymaker(**int** n, unordered\_set<**int**> **&**s)

{

    if (n == 1)

    {

        return true;

    }

    if (s.find(n) != s.end())

    {

        return false;

    }

    s.insert(n);

    if (n % 2)

    {

        return arraymaker(3 \* n + 1, s);

    }

    else

    {

        return arraymaker(n / 2, s);

    }

}

**bool** checker(**int** n)

{

    unordered\_set<**int**> s;

    return arraymaker(n, s);

}

**int** main()

{

**int** n = 5;

    if (checker)

    {

        cout << "Yes";

    }

    else

    {

        cout << "NO";

    }

    return 0;

}

**Output:**



2. Implement Fibonacci Series using Iterative, Recursive and Golden ratio.

**Code:**

#include <bits/stdc++.h>

#include <iostream>

using **namespace** std;

**int** fibonaccirec(**int** n)

{

    if ((n == 1) || (n == 0))

    {

        return (n);

    }

    else

    {

        return (fibonaccirec(n - 1) + fibonaccirec(n - 2));

    }

}

**void** fibreccaller(**int** n)

{

**int** i = 0;

    while (i < n)

    {

        cout << " " << fibonaccirec(i);

        i++;

    }

}

**void** fibiter(**int** n)

{

**int** x = 0, y = 1, z = 0;

    for (**int** i = 0; i < n; i++)

    {

        cout << x << " ";

        z = x + y;

        x = y;

        y = z;

    }

}

**double** PHI = 1.6180339;

**int** f[6] = {0, 1, 1, 2, 3, 5};

**int** fibgolden(**int** n)

{

    if (n < 6)

        return f[n];

**int** t = 5, fn = 5;

    while (t < n)

    {

        fn = round(fn \* PHI);

        t++;

    }

    return fn;

}

**void** fibgolcaller(**int** n)

{

**int** i = 0;

    while (i < n)

    {

        cout << " " << fibgolden(i);

        i++;

    }

}

**int** main()

{

**int** n;

    cout << "Enter no. of numbbers in fib. series:";

    cin >> n;

    cout << "\nRecursive:";

    fibreccaller(n);

    cout << "\nIterative:";

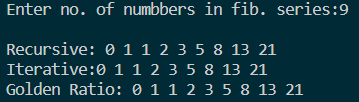
    fibiter(n);

    cout << "\nGolden Ratio:";

    fibgolcaller(n);

}

**Output:**



1. Count ways to reach the nth stair using step 1, 2 or 3

**Code:**

#include <bits/stdc++.h>

#include <iostream>

using **namespace** std;

**int** findStep(**int** n)

{

    if (n == 0)

        return 1;

    else if (n < 0)

        return 0;

    else

        return findStep(n - 3) + findStep(n - 2) + findStep(n - 1);

}

**int** main()

{

**int** n;

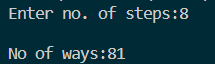
    cout << "Enter no. of steps:";

    cin >> n;

    cout << "\nNo of ways:" << findStep(n);

}

**Output:**



4. Karatsuba algorithm for fast multiplication using Divide and Conquer algorithm

**Code:**

#include<iostream>

#include<stdio.h>

#include<math.h>

using **namespace** std;

**long** karatsuba(**long** x,**long** y)

{

    if(x<10 and y<10)

    {

        return x\*y;

    }

**long** size = max(to\_string(x).length(),to\_string(y).length());

**int** n = (**int**)ceil(size / 2.0);

**long** p = (**long**)pow(10, n);

**long** a = (**long**)floor(x / (**double**)p);

**long** b = x % p;

**long** c = (**long**)floor(y / (**double**)p);

**long** d = y % p;

**long** ac = karatsuba(a, c);

**long** bd = karatsuba(b, d);

**long** e = karatsuba(a + b, c + d) - ac - bd;

    return (**long**)(pow(10 \* 1L, 2 \* n) \* ac + pow(10 \* 1L, n) \* e + bd);

}

**int** main()

{

**int** a,b;

    cout<<"Enter a:";

    cin>>a;

    cout<<"Enter b:";

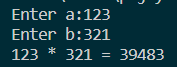
    cin>>b;

    cout<<a<<" \* "<<b<<" = "<<karatsuba(a,b)<<endl;

    return 0;

}

**Output:**

****