Design and Analysis of Algorithms – Lab 1

Name Mitrajeet Golsangi

Roll No 01

Div CS-B

Batch B1

# Number Problems

## Prime Numbers from 1 to n

### Code

*#include* <iostream>

using namespace std;

int check\_prime(int, int);

int main()

{

    int n;

    cout << "Enter the higher limit : ";

    cin >> n;

*for* (int i = 2; i <= n; i++)

    {

*if* (check\_prime(2, i) == 0)

            cout << i << " ";

    }

*return* 0;

}

int check\_prime(int i, int n)

{

*if* (n == i)

*return* 0;

*else* *if* (n % i == 0)

*return* 1;

*else*

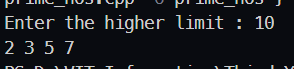
    {

*return* check\_prime(i + 1, n);

    }

}

### Output

## Euclidean algorithm for LCM

### Code

*#include* <iostream>

using namespace std;

int gcd(int, int);

int lcm(int, int);

int main()

{

    system("cls");

    int a, b;

    cout << "Enter 1st number : ";

    cin >> a;

    cout << "Enter 2nd number : ";

    cin >> b;

    cout << "LCM is : " << lcm(a, b);

*return* 0;

}

int gcd(int a, int b)

{

*if* (b == 0)

*return* a;

*else*

*return* gcd(b, a % b);

}

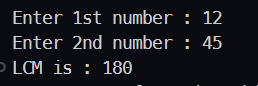
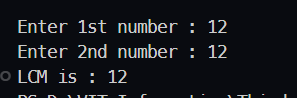
int lcm(int a, int b)

{

*return* (a \* b) / gcd(a, b);

}

### Output

## Palindrome checking

### Code

*#include* <iostream>

using namespace std;

int reverse(int, int);

int main()

{

    system("cls");

    int num, rev;

    cout << "Enter a Number : ";

    cin >> num;

    rev = reverse(num, 0);

*if* (rev == num)

        cout << "Number is a Palindrome !";

*else*

        cout << "Number is not a Palindrome !";

*return* 0;

}

int reverse(int num, int temp)

{

*if* (num == 0)

*return* temp;

*else*

    {

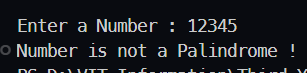
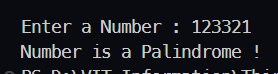
        temp = (num % 10) + (10 \* temp);

*return* reverse(num / 10, temp);

    }

}

### Output

## Fibonacci Series upto given number

### Code

*#include* <iostream>

using namespace std;

void fibonacci(int, int, int);

int main()

{

    system("cls");

    int max;

    cout << "Enter the upper limit : ";

    cin >> max;

    fibonacci(1, 1, max);

*return* 0;

}

void fibonacci(int prev, int current, int max)

{

*if* (current >= max)

*return*;

*else* {

*if* (prev == current == 1)

            cout << prev << " ";

        cout << current << " ";

        int temp\_prev = current;

        current = prev + current;

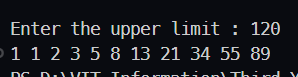
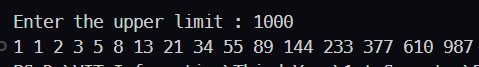
        prev = temp\_prev;

*return* fibonacci(prev, current, max);

    }

}

### Output

## Print all possible numbers of length k that can be formed from a given set of numbers.

### code

*#include* <iostream>

using namespace std;

void printAllKLengthRec(char set[], string prefix,

                        int n, int k)

{

*if* (k == 0)

    {

        cout << (prefix) << endl;

*return*;

    }

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

    {

        string newPrefix;

        newPrefix = prefix + set[i];

        printAllKLengthRec(set, newPrefix, n, k - 1);

    }

}

void printAllKLength(char set[], int k, int n)

{

    printAllKLengthRec(set, "", n, k);

}

int main()

{

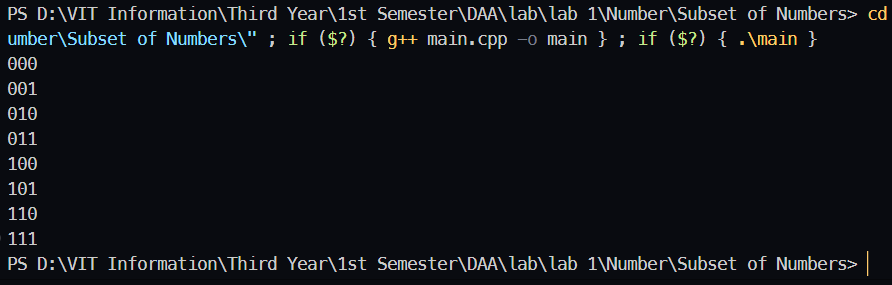
    char set[] = {'0', '1'};

    int k = 3;

    printAllKLength(set, k, 2);

}

### output



# Matrices Problems

# Graph Problems