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*

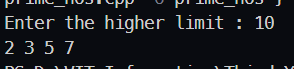
    {

*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);

}

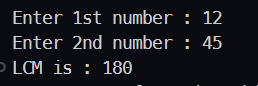
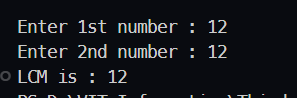
int lcm(int a, int b)

{

*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*

    {

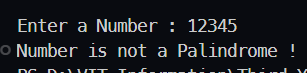
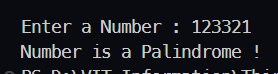
        temp = (num % 10) + (10 \* temp);

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

    }

}

### Output

## Fibonacci Series up to 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;

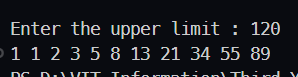
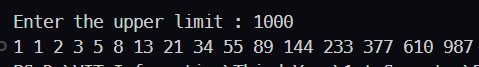
        prev = temp\_prev;

*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()

{

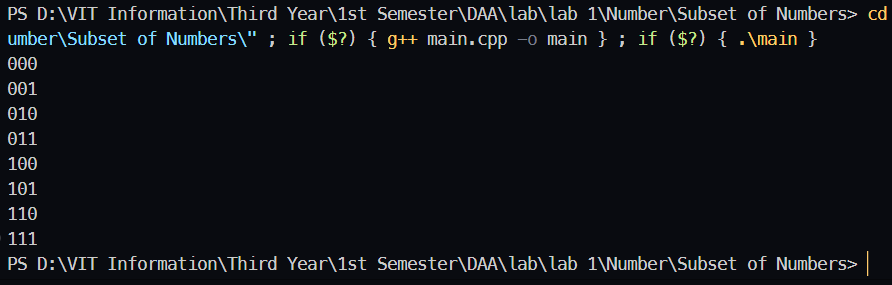
    char set[] = {'0', '1'};

    int k = 3;

    printAllKLength(set, k, 2);

}

### output



# Matrices Problems

## Transpose

### Code

*#include* <iostream>

*#include* <iomanip>

using namespace std;

void transpose(int[3][3], int[3][3]);

void display(int[3][3], string);

int main()

{

    int mat[3][3] = {

        {1, 2, 3},

        {4, 5, 6},

        {7, 8, 9},

    };

    int mat\_T[3][3];

    transpose(mat, mat\_T);

    display(mat, "A");

    display(mat\_T, "A Transpose");

*return* 0;

}

void transpose(int a[3][3], int b[3][3])

{

*for* (int i = 0; i < 3; i++)

    {

*for* (int j = 0; j < 3; j++)

        {

            b[j][i] = a[i][j];

        }

    }

}

void display(int mat[3][3], string name)

{

    cout << "Matrix " << name << endl;

*for* (int i = 0; i < 3; i++)

    {

*for* (int j = 0; j < 3; j++)

        {

            cout << left << setw(5) << mat[i][j];

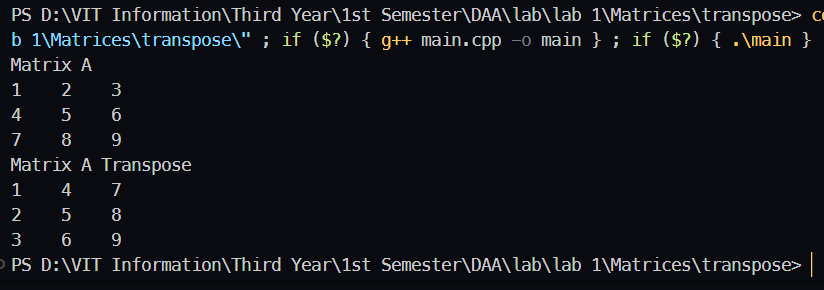
        }

        cout << endl;

    }

}

### output



## Spiral Matrix Representation

### Code

*#include* <iostream>

*#include* <iomanip>

using namespace std;

void printSpiral(int mat[4][4], int R, int C)

{

    int top = 0, left = 0, bottom = R - 1, right = C - 1;

*while* (top <= bottom && left <= right)

    {

*for* (int i = left; i <= right; i++)

            cout << mat[top][i] << " ";

        top++;

*for* (int i = top; i <= bottom; i++)

            cout << mat[i][right] << " ";

        right--;

*if* (top <= bottom)

        {

*for* (int i = right; i >= left; i--)

                cout << mat[bottom][i] << " ";

            bottom--;

        }

*if* (left <= right)

        {

*for* (int i = bottom; i >= top; i--)

                cout << mat[i][left] << " ";

            left++;

        }

    }

}

int main()

{

    const int R = 4, C = 4;

    int arr[R][C] = {{1, 2, 3, 4},

                     {5, 6, 7, 8},

                     {9, 10, 11, 12},

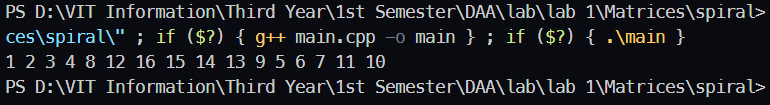
                     {13, 14, 15, 16}};

    printSpiral(arr, R, C);

*return* 0;

}

### output



## Diagonal Printing of Matrix

### Code

*#include* <iostream>

*#include* <iomanip>

using namespace std;

class MinMax

{

public:

    int min;

    int max;

    static int getMin(int a, int b)

    {

*return* (a < b) ? a : b;

    }

    static int getMin(int a, int b, int c)

    {

*return* getMin(getMin(a, b), c);

    }

    static int getMax(int a, int b)

    {

*return* (a > b) ? a : b;

    }

};

void printDiagonal(int[3][3]);

void display(int[3][3], string);

int main()

{

    int mat[3][3] = {

        {1, 2, 3},

        {4, 5, 6},

        {7, 8, 9},

    };

    display(mat, "A");

    cout << "\nDiagonal is\n";

    printDiagonal(mat);

*return* 0;

}

void printDiagonal(int mat[3][3])

{

    int sc = 0;

    int sr = 0;

    int cnt = 0;

*for* (int i = 1; i < 6; i++)

    {

        sc = MinMax::getMax(0, i - 3);

        sr = MinMax::getMin(3, i);

        cnt = MinMax::getMin(i, (3 - sc), 3);

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

        {

            cout << left << setw(5) << mat[sr - j - 1][sc + j];

        }

        cout << endl;

    }

}

void display(int mat[3][3], string name)

{

    cout << "Matrix " << name << endl;

*for* (int i = 0; i < 3; i++)

    {

*for* (int j = 0; j < 3; j++)

        {

            cout << left << setw(5) << mat[i][j];

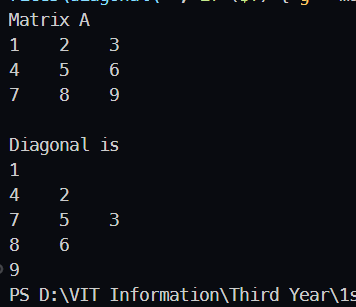
        }

        cout << endl;

    }

}

### Output



## Matrix Multiplication

### Code

### Output

# Graph Problems