Chapter 1:- Program Logic development

1.1 Fundamentals of algorithms: An algorithm is a step-by-step procedure or a set of rules for solving a specific problem. It is a well-defined sequence of instructions that takes some input, performs a series of operations, and produces an output. Algorithms can be represented using pseudo-code, which is a high-level description of the algorithm using a combination of natural language and programming language-like syntax.
Pseudo-code conventions typically include assignment statements, which involve assigning a value to a variable. For example, "x = 5" assigns the value 5 to the variable x. Basic control structures like loops and conditionals are also commonly used in pseudo-code. Loops allow for repeated execution of a set of instructions, while conditionals allow for the execution of different sets of instructions based on certain conditions.

1.2 Algorithmic problems: (i) Exchange the values of two variables with and without a temporary variable:

- With a temporary variable: temp = a a = b b = temp
- Without a temporary variable: a = a + b b = a - b a = a - b

(ii) Counting positive numbers from a set of integers: Initialize a counter variable to 0 For each number in the set: If the number is positive, increment the counter Output the value of the counter
(iii) Summation of a set of numbers: Initialize a sum variable to 0 For each number in the set: Add the number to the sum variable Output the value of the sum variable
(iv) Reversing the digits of an integer: Convert the integer to a string Reverse the characters in the string Convert the string back to an integer Output the reversed integer
(v) Find the smallest positive divisor of an integer other than 1: Initialize a divisor variable to 2 While the integer is divisible by the divisor: Divide the integer by the divisor Output the divisor
(vi) Find the G.C.D. and L.C.M. of two as well as three positive integers: For two integers: Use the Euclidean algorithm to find the G.C.D. L.C.M. can be calculated using the formula: (a * b) / G.C.D. For three integers: Find the G.C.D. of the first two integers Then find the G.C.D. of the result and the third integer L.C.M. can be calculated using the formula: (a * b * c) / G.C.D.
(vii) Generating prime numbers: Start with a list of prime numbers containing 2 For each number from 3 to the desired range: Check if the number is divisible by any prime number in the list If not, add it to the list of prime numbers Output the list of prime numbers

1.3 Flow chart: A flow chart is a graphical representation of an algorithm or a process. It uses various symbols to represent different steps, decisions, or actions in the algorithm. Flow charts can be used to visually understand the flow of control and the sequence of operations in an algorithm.
To draw a flow chart for each algorithm developed, you can use symbols such as rectangles for process steps, diamonds for decision points, arrows to indicate the flow of control, and ovals or rounded rectangles for start and end points. Each symbol should be labeled with a brief description of the step or action it represents.

(i) Exchange the values of two variables with and without a temporary variable:

```c
// With a temporary variable
#include<stdio.h>
void main()
{
    int a, b, temp;
    printf("Enter the values of a and b: ");
    scanf("%d %d", &a, &b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping, a = %d and b = %d\n", a, b);
}

// Without a temporary variable
#include<stdio.h>
void main()
{
    int a, b;
    printf("Enter the values of a and b: ");
    scanf("%d %d", &a, &b);

    a = a + b;
    b = a - b;
    a = a - b;

    printf("After swapping, a = %d and b = %d\n", a, b);
}
```

(ii) Counting positive numbers from a set of integers:

```c
#include<stdio.h>
void main()
{
    int set[10], count = 0;
    printf("Enter 10 integers: ");
    for (int i = 0; i < 10; i++)
    {
        scanf("%d", &set[i]);
        if (set[i] > 0)
            count++;
    }
    printf("Number of positive integers: %d\n", count);
}
```

(iii) Summation of a set of numbers:

```c
#include<stdio.h>
void main()
{
    int set[5], sum = 0;
    printf("Enter 5 integers: ");
    for (int i = 0; i < 5; i++)
    {
        scanf("%d", &set[i]);
        sum += set[i];
    }
    printf("Sum of the numbers: %d\n", sum);
}
```

(iv) Reversing the digits of an integer:

```c
#include<stdio.h>
void main()
{
    int num, reversed = 0;
    printf("Enter an integer: ");
    scanf("%d", &num);

    while (num != 0)
    {
        int digit = num % 10;
        reversed = reversed * 10 + digit;
        num /= 10;
    }

    printf("Reversed integer: %d\n", reversed);
}
```

(v) Find the smallest positive divisor of an integer other than 1:

```c
#include<stdio.h>
void main()
{
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    int divisor = 2;
    while (num % divisor != 0)
    {
        divisor++;
    }

    printf("Smallest positive divisor (other than 1): %d\n", divisor);
}
```

(vi) Find the G.C.D. and L.C.M. of two positive integers:

```c
#include<stdio.h>
void main()
{
    int num1, num2, gcd, lcm, temp;
    printf("Enter two positive integers: ");
    scanf("%d %d", &num1, &num2);

    int a = num1;
    int b = num2;

    while (b != 0)
    {
        temp = b;
        b = a % b;
        a = temp;
    }

    gcd = a;
    lcm = (num1 * num2) / gcd;

    printf("G.C.D: %d\n", gcd);
    printf("L.C.M: %d\n", lcm);
}
```

(vii) Generating prime numbers:

```
#include<stdio.h>
#include<stdbool.h>

bool isPrime(int num)
{
    if (num <= 1)
        return false;

    for (int i = 2; i <= num / 2; i++)
    {
        if (num % i == 0)
            return false;
    }

    return true;
}

void main()
{
    int range;
    printf("Enter the range: ");
    scanf("%d", &range);

    printf("Prime numbers within the range: ");
    for (int i = 2; i <= range; i++)
    {
        if (isPrime(i))
            printf("%d ", i);
    }
    printf("\n");
}
```