WEEK 8

1# Write a C++ program that uses functions to perform the following operations:
To insert a sub string into a given main string from a given position. ii. To delete n characters from a given position in a given string

```
#include <iostream>
#include <string>
using namespace std;
void insrt_str(string &mn_str, string &s_str, int n)
{
  if (n < 0 || n > mn_str.length())
  {
    cout << "Invalid inserting position";
    return;
  }
  else
    mn_str.insert(n, " " + s_str);
}
void del_str(string &str, int no_char, int pos)
{
  str.erase(pos, no_char);
}
int main()
{
  int select;
  string main_str, sub_str;
  int pos_i;
  string str_d;
  int pos_d, n_d;
  cout << "Enter a string\n";
  getline(cin, main_str);
  cout << "what do you want to do?\nEnter 1 for Insertion operation and Enter 2 for Deletion"
<< endl;
  cin >> select;
  switch (select)
  {
  case 1:
    cout << "Enter a substring that you want to insert: ";
    cin.ignore();
    getline(cin, sub_str);
    cout << "Enter the position at which string is inserted: ";
    cin >> pos_i;
    insrt_str(main_str, sub_str, pos_i);
    cout << main_str << endl;
    break;
  case 2:
    cout << "Enter the position and number of characters to be deleted: ";
    cin >> pos_d >> n_d;
```

```
    del_str(main_str, n_d, pos_d);
    cout << main_str;
    break;
  }
```

2# Write a C++ program to determine if the given string is a palindrome or not.

```
#include <bits/stdc++.h>
using namespace std;
string reverse(const string &myStr)
{
    string reversedStr = myStr;
    int length = myStr.length();

    for (int i = length - 1; i >= 0; i--)
    {
        reversedStr[length - 1 - i] = myStr[i];
    }
    return reversedStr;
}
int main()
{
    string mainStr;
    cout << "Enter a string: ";
    getline(cin, mainStr);
    cout << "The stored string is: " << mainStr << endl;
    string reverseStr = reverse(mainStr);
    cout << "The reversed string is: " << reverseStr << endl;
    if (mainStr == reverseStr)
    {
        cout << "Given string is a palindrome";
    }
    else
    {
        cout << "Given string is not a palindrome";
    }
    return 0;
}
```

3# Write a C++ program to find a string within a sentence and replace it with another string.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string mainString, toReplace, replacement;
    cout << "Enter a string: ";
    getline(cin, mainString);
    cout << "Enter string that you want to Replace (part of main string): ";
    getline(cin, toReplace);
```

```
    cout << "Replace with: ";
    getline(cin, replacement);
    // Find the position of the substring to be replaced
    size_t pos = mainString.find(toReplace);
    if (pos != string::npos)
    {
        // Replace the substring with the new string
        mainString.replace(pos, toReplace.length(), replacement);
        cout << "Modified string: " << mainString << endl;
    }
    else
    {
        cout << toReplace << " not found in the main string." << endl;
    }
    return 0;
}
```

4# Write a C++ program that reads a line of text and counts all occurrence of a particular word.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int count = 0;
    string mainString, sString;
    cout << "Enter a string: ";
    getline(cin, mainString);
    cout << "Enter word/string to find counting: ";
    getline(cin, sString);
    size_t f = mainString.find(sString);
    while (f != string::npos)
    {
        count++;
        f = mainString.find(sString, f + 1);
    }
    cout << "The word " << sString << " appears " << count << " times in the string\n";
}
```

5# Write a C++ program that displays the position or index in the string S where the string T begins, or 1 if S doesn't contain T.

```
#include <iostream>
using namespace std;
int main()
{
    string S, T;
    cout << "Enter a string S: ";
    getline(cin, S);
    cout << "Enter a word T: ";
    getline(cin, T);
```

```
    size_t pos = S.find(T);
    long len = T.length();
    int final = pos + len;

    if (pos != string::npos)
    {
        cout << T << " is from postion " << pos + 1 << " to " << final;
    }
    else
    {
        cout << "1";
    }
}
```

WEEK 9

1# Write C programs that use both recursive and non-recursive functions to find:

a) The factorial of a given integer.

b) To find the greatest common divisor of two given integers.

```
#include <iostream>
using namespace std;
// factorial by both recursive and non-recursive
long long int fact_rec(int a)
{
    if (a == 1 || a == 0)
    {
        return 1;
    }
    else
    {
        return a * fact_rec(a - 1);
    }
}

long long int fact_NRec(int a)
{
    if (a < 0)
    {
        cout << "Factorial of negative number is not possible";
        return 0;
    }

    int b = 1;
    cout << "Factorial is   ";
    for (int i = 1; i <= a; i++)
    {
        b *= i;
    }
    return b;
}

int main()
{
    int x;
    long int factAns, factAns2;
    cout << "Enter the number for which you want the factorial number\n";
    cin >> x;
    factAns = fact_NRec(x);
    factAns2 = fact_rec(x);
    cout << factAns << endl;
    cout << factAns2;
#include <iostream>
using namespace std;
```

```
// Non-recursive function to calculate the GCD of two numbers using Euclidean algorithm
int gcdNonRecursive(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
int main()
{
    int num1, num2;
    cout << "Enter two integers: ";
    cin >> num1 >> num2;

    int result = gcdNonRecursive(num1, num2);
    cout << "GCD of " << num1 << " and " << num2 << " (using iteration) is: " << result <<
endl;
    return 0;
}
#include <iostream>
using namespace std;
// Recursive function to calculate the GCD of two numbers using Euclidean algorithm
int gcdRecursive(int a, int b)
{
    if (b == 0)
        return a;
    return gcdRecursive(b, a % b);
}
int main()
{
    int num1, num2;
    cout << "Enter two integers: ";
    cin >> num1 >> num2;
    int result = gcdRecursive(num1, num2);
    cout << "GCD of " << num1 << " and " << num2 << " (using recursion) is: " << result <<
endl;

    return 0;
}
```

2# Write C programs that use both recursive and non-recursive functions to solve towers of Hanoi problem.

```
#include <iostream>
#include <stack>
//Towers of Hanoi by recursive and non-recursive

using namespace std;
```

```
struct Move
{
    int n;
    char source, auxiliary, destination;
};
void towerOfHanoiNonRecursive(int n, char source, char auxiliary, char destination)
{
    stack <Move> moves;
    Move initialMove = {n, source, auxiliary, destination};
    moves.push(initialMove);

    while (!moves.empty())
    {
        Move currentMove = moves.top();
        moves.pop();
        n = currentMove.n;
        source = currentMove.source;
        auxiliary = currentMove.auxiliary;
        destination = currentMove.destination;
        if (n == 1)
        {
            cout << "Move disk 1 from " << source << " to " << destination << endl;
        }
        else
        {
            // Push the next moves onto the stack in reverse order
            moves.push({n - 1, auxiliary, source, destination});
            moves.push({1, source, auxiliary, destination});
            moves.push({n - 1, source, destination, auxiliary});
        }
    }
}

int main()
{
    int numDisks;
    cout << "Enter the number of disks: ";
    cin >> numDisks;
    towerOfHanoiNonRecursive(numDisks, 'A', 'B', 'C');
    return 0;
}
#include <iostream>
using namespace std;

// tower of hanoi problem using recursion
void pm(int start, int end);
void hanoi(int n, int start, int end)
{
    if (n == 1)
    {
        pm(start, end);
    }
    else
```

```
    {
        int other = 6 - (start + end);
        hanoi(n - 1, start, other);
        pm(start, end);
        hanoi(n - 1, other, end);
    }
}
void pm(int start, int end)
{
    cout << start << "->" << end << endl;
}

int main()
{
    int n;
    cout << "Enter number of discs: ";
    cin >> n;
    cout << "We assume the number of rods be 3\n";
    hanoi(n, 1, 3);
    return 0;
}
```

3# Write a C++ program to print the transpose of a given matrix using function.

```
#include <iostream>
using namespace std;
// transpose of matrix

// Function to compute the transpose of a matrix
void transposeMatrix(int original[][100], int transposed[][100], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transposed[j][i] = original[i][j];
        }
    }
}

int main() {
    int rows, cols;
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> cols;
    int originalMatrix[100][100];
    int transposedMatrix[100][100];
    if ((rows < 0) || (cols < 0)){
        return -1;
    }

    // Input the original matrix
    cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < rows; i++) {
```

```
        for (int j = 0; j < cols; j++) {
            cin >> originalMatrix[i][j];
        }
    }

    // Compute the transpose of the matrix
    transposeMatrix(originalMatrix, transposedMatrix, rows, cols);

    // Print the transpose matrix
    cout << "Transpose of the matrix:" << endl;
    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            cout << transposedMatrix[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

4# Write a C++ program to swap two number by both call by value and call by reference mechanism, using two functions swap_value() and swap_reference respectively, by getting the choice from the user and executing the user's choice by switch-case.

```
#include <iostream>
using namespace std;
// swap two numbers

// Function to swap two numbers by value
void swap_value(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}

// Function to swap two numbers by reference
void swap_reference(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}

int main()
{
    int num1, num2, choice;
    cout << "Enter two numbers: ";
    cin >> num1 >> num2;
    cout << "Choose swapping mechanism:" << endl;
    cout << "1. Call by Value" << endl;
    cout << "2. Call by Reference" << endl;
    cin >> choice;
```

```
    switch (choice)
    {
    case 1:
        // Swap by value
        swap_value(num1, num2);
        cout << "Swapped numbers (Call by Value): " << num1 << " " << num2 << endl;
        break;
    case 2:
        // Swap by reference
        swap_reference(num1, num2);
        cout << "Swapped numbers (Call by Reference): " << num1 << " " << num2 << endl;
        break;
    default:
        cout << "Invalid choice. Please enter 1 or 2." << endl;
    }

    return 0;
}
```

5# Write a C++ program to display all array elements using recursion.

```
#include <iostream>
using namespace std;

// Function to display array elements using recursion
void displayArray(int arr[], int size, int index)
{
    if (index == size)
    {
        return; // Base case: stop when the index reaches the array size
    }
    cout << arr[index] << " ";
    displayArray(arr, size, index + 1);
}

int main()
{
    int size;
    int* arr = new int[size];
    cout << "Enter size for the array: ";
    cin >> size;
    for (int i = 0; i < size; i++)
    {
        cout << "-> ";
        cin >> arr[i];
    }
    cout << "Array elements using recursion: ";
    displayArray(arr, size, 0);
    cout << endl;
    return 0;
}
```

6# Write a C++ program to find sum of elements of array using recursion.

```cpp
#include <iostream>
using namespace std;
// Function to find the sum of elements in an array using recursion
int arraySum(int arr[], int size, int index)
{
    if (index < 0)
    {
        return 0;
    }
    return arr[index] + arraySum(arr, size, index - 1);
}
void displayArray(int arr[], int size, int index)
{
    if (index == size)
    {
        return;
    }
    cout << arr[index] << " ";
    displayArray(arr, size, index + 1);
}

int main()
{
    int size;
    int* arr = new int[size];
    cout << "Enter size for the array: ";
    cin >> size;
    for (int i = 0; i < size; i++)
    {
        cout << "-> ";
        cin >> arr[i];
    }
    cout << "Array elements: ";
    displayArray(arr, size, 0);
    cout << endl;
    int sum = arraySum(arr, size, size - 1);
    cout << "Sum of array elements using recursion: " << sum << endl;
    return 0;
}
```

7# Write a C++ program to find maximum and minimum elements in array using recursion.

```cpp
#include <iostream>
using namespace std;

// Function to find the maximum element in an array using recursion
int findMax(int arr[], int size) {
    if (size == 1) {
        return arr[0];
    }
    int maxInRest = findMax(arr, size - 1);
    return (arr[size - 1] > maxInRest) ? arr[size - 1] : maxInRest;
}
```

```cpp
// Function to find the minimum element in an array using recursion
int findMin(int arr[], int size) {
    if (size == 1) {
        return arr[0];
    }
    int minInRest = findMin(arr, size - 1);
    return (arr[size - 1] < minInRest) ? arr[size - 1] : minInRest;
}
int main() {
    int size;
    cout << "Enter size for the array: ";
    cin >> size;
    int* arr = new int[size];
    cout << "Enter elements for the array: ";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }
    int maxElement = findMax(arr, size);
    int minElement = findMin(arr, size);
    cout << "Maximum element in the array: " << maxElement << endl;
    cout << "Minimum element in the array: " << minElement << endl;
    delete[] arr;
    return 0;
}
```

WEEK 10

1# Write a C++ program that uses functions to perform the following operations:

i. Reading a complex number
ii. Writing a complex number
iii. Addition and subtraction of two complex numbers
iv. Multiplication of two complex numbers. Note: represent complex number using a structure.

```
#include <iostream>
using namespace std;

struct Complex {
    float real;
    float imag;
};
void readComplexNumber(Complex &num) {
    cout << "Enter real part: ";
    cin >> num.real;
    cout << "Enter imaginary part: ";
    cin >> num.imag;
}
void writeComplexNumber(const Complex &num) {
    cout << "Complex Number: " << num.real;
    if (num.imag >= 0) cout << " + " << num.imag << "i";
    else cout << " - " << -num.imag << "i";
    cout << endl;
}
Complex addComplexNumbers(const Complex &num1, const Complex &num2) {
    Complex result;
    result.real = num1.real + num2.real;
    result.imag = num1.imag + num2.imag;
    return result;
}
Complex subtractComplexNumbers(const Complex &num1, const Complex &num2) {
    Complex result;
    result.real = num1.real - num2.real;
    result.imag = num1.imag - num2.imag;
    return result;
}
Complex multiplyComplexNumbers(const Complex &num1, const Complex &num2) {
    Complex result;
    result.real = num1.real * num2.real - num1.imag * num2.imag;
    result.imag = num1.real * num2.imag + num1.imag * num2.real;
    return result;
}
int main() {
    Complex num1, num2, sum, difference, product;
    cout << "Enter first complex number:\n";
    readComplexNumber(num1);
    cout << "Enter second complex number:\n";
    readComplexNumber(num2);
```

```
    sum = addComplexNumbers(num1, num2);
    difference = subtractComplexNumbers(num1, num2);
    product = multiplyComplexNumbers(num1, num2);
    cout << "\nSum of the complex numbers:\n";
    writeComplexNumber(sum);
    cout << "\nDifference of the complex numbers:\n";
    writeComplexNumber(difference);
    cout << "\nProduct of the complex numbers:\n";
    writeComplexNumber(product);
    return 0;
}
```

2# Write a C++ program to compute the monthly pay of 100 employees using each employee's name, basic pay. The DA is computed as 52% of the basic pay. Gross-salary (basic pay + DA). Print the employees name and gross salary

```
#include <iostream>
#include <string>
using namespace std;
struct Employee {
    string name;
    float basicPay;
};
void calculateGrossSalary(Employee &employee) {
    const float DA_PERCENTAGE = 52;
    float DA = (DA_PERCENTAGE / 100) * employee.basicPay;
    float grossSalary = employee.basicPay + DA;
    cout << "Employee Name: " << employee.name << endl;
    cout << "Gross Salary: " << grossSalary << endl;
}
int main() {
    int numEmployees;
    cout << "Enter Employee Number: ";
    cin >> numEmployees;
    Employee employees[numEmployees];
    for (int i = 0; i < numEmployees; ++i) {
        cout << "Enter name of employee #" << (i + 1) << ": ";
        cin >> employees[i].name;
        cout << "Enter basic pay of employee #" << (i + 1) << ": ";
        cin >> employees[i].basicPay;
    }
    cout << "\nEmployee Gross Salaries:\n";
    for (int i = 0; i < numEmployees; ++i) {
        calculateGrossSalary(employees[i]);
    }
    return 0;
}
```

3# Create a Book structure containing book_id, title, author name and price. Write a C++ program to pass a structure as a function argument and print the book details.

```cpp
#include <iostream>
#include <string>
using namespace std;
struct Book {
    int book_id;
    string title;
    string author;
    float price;
};
void printBookDetails(const Book &book) {
    cout << "Book ID: " << book.book_id << endl;
    cout << "Title: " << book.title << endl;
    cout << "Author: " << book.author << endl;
    cout << "Price: $" << book.price << endl;
}
int main() {
    Book myBook;
    // Assign values to the structure members
    myBook.book_id = 1;
    myBook.title = "How to be Rich";
    myBook.author = "John Doe";
    myBook.price = 2010;
    // Call the function and pass the structure as an argument
    printBookDetails(myBook);
    return 0;
}
```

4# Create a union containing 6 strings: name, home_address, hostel_address, city, state and zip. Write a C++ program to display your present address

```cpp
#include <iostream>
#include <string>
using namespace std;
struct Address {
    string name;
    string home_address;
    string hostel_address;
    string city;
    string state;
    string zip;
};
int main() {
    Address myAddress;
    myAddress.name = "John Doe";
    myAddress.home_address = "123 Main Ghar";
    myAddress.hostel_address = "456 College Ave";
    myAddress.city = "Stadt";
    myAddress.state = "Kanada";
    myAddress.zip = "12345";

    // Displaying the present address
```

```
    cout << "Name: " << myAddress.name << endl;
    cout << "Home Address: " << myAddress.home_address << endl;
    cout << "Hostel Address: " << myAddress.hostel_address << endl;
    cout << "City: " << myAddress.city << endl;
    cout << "State: " << myAddress.state << endl;
    cout << "Zip: " << myAddress.zip << endl;
    return 0;
}
```

5# Write a C++ program to define a structure named D.O.B., which contains name, day, month and year. Using the concept of nested structures display your name and date of birth.

```
#include <iostream>
#include <string>
using namespace std;
struct Date {
    int day;
    int month;
    int year;
};
struct DOB {
    string name;
    Date birthdate;
};
int main() {
    DOB myInfo;
    myInfo.name = "Sarim";
    myInfo.birthdate.day = 25;
    myInfo.birthdate.month = 02;
    myInfo.birthdate.year = 2002;
    // Displaying name and date of birth
    cout << "Name: " << myInfo.name << endl;
    cout << "Date of Birth: " << myInfo.birthdate.day << "/"
         << myInfo.birthdate.month << "/" << myInfo.birthdate.year << endl;
    return 0;
}
```

WEEK 11

1# Write a program in C++ to display your name, Branch, Year on to the computer screen without using classes and object. All information should be displayed in the separate line.

```
#include <iostream>
using namespace std;
int main() {

    cout << "Xeroriano Fenix" << endl;
    cout << "Computer Science" << endl;
    cout << "1st Year" << endl;
    return 0;
}
```

2# Write a menu driven program in C++ to perform all basic arithmetic operation addition, subtraction,
multiplication, and division of two given values. Program receives two values and required operation to be performed from the keyboard and display particular result of the required operation.

```
#include <iostream>
using namespace std;
int main() {
    char choice;
    float num1, num2, result;
    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;
    cout << "Choose an operation:" << endl;
    cout << "1. Addition (+)" << endl;
    cout << "2. Subtraction (-)" << endl;
    cout << "3. Multiplication (*)" << endl;
    cout << "4. Division (/)" << endl;
    cout << "Enter your choice: ";
    cin >> choice;
    switch(choice) {
        case '+':
            result = num1 + num2;
            cout << "Result: " << result << endl;
            break;
        case '-':
            result = num1 - num2;
            cout << "Result: " << result << endl;
            break;
        case '*':
            result = num1 * num2;
            cout << "Result: " << result << endl;
            break;
```

```
        case '/':
            if(num2 != 0) {
                result = num1 / num2;
                cout << "Result: " << result << endl;
            } else {
               cout << "Error! Division by zero is not allowed." << endl;
            }
            break;
        default:
            cout << "Invalid choice!" << endl;
    }
    return 0;
}
```

3# Write a menu driven program in C++ that receives 4-digit integer value the keyboard and perform following operations:

i) Reverse of that no.
ii) sum of number with its reverse.
iii) sum of alternative digits (1 digit+3 digit and 2 digit+4 digit)

```
#include <iostream>
using namespace std;
int main()
{
    int number, originalNumber, reverseNumber = 0, digit, sum = 0, alternateSum1 = 0,
alternateSum2 = 0;
    cout << "Enter a 4-digit number: ";
    cin >> number;
    if (number < 1000 || number > 9999)
    {
        cout << "Invalid input. Please enter a 4-digit number." << endl;
        return 1; // Exit the program with an error code
    }
    originalNumber = number;
    while (number != 0)
    {
        digit = number % 10;
        reverseNumber = reverseNumber * 10 + digit;
        number = number / 10;
    }

    // Calculate the sum of the number with its reverse
    sum = originalNumber + reverseNumber;

    // Calculate the sum of alternate digits separately
    int position = 1;
    while (originalNumber != 0)
    {
        digit = originalNumber % 10;
        if (position % 2 == 0)
        {
            alternateSum1 += digit;
```

```
        }
        else
        {
            alternateSum2 += digit;
        }
        originalNumber = originalNumber / 10;
        position++;
    }
    cout << "Reverse of the number: " << reverseNumber << endl;
    cout << "Sum of the number with its reverse: " << sum << endl;
    cout << "Sum of alternate digits at even positions: " << alternateSum1 << endl;
    cout << "Sum of alternate digits at odd positions: " << alternateSum2 << endl;

    return 0;
}
```

4# Write a menu driven program in C++ to receive integer number and convert equivalent binary, octal, hexadecimal number.

```
#include <iostream>
#include <bitset>
using namespace std;
int main() {
    int num;
    char choice;
    do {
        cout << "Enter an integer number: ";
        cin >> num;
        cout << "Binary equivalent: " << bitset<32>(num) << endl;
        cout << "Octal equivalent: 0" << oct << num << endl;
        cout << "Hexadecimal equivalent: 0x" << hex << num << endl;

        cout << "Do you want to continue? (y/n): ";
        cin >> choice;
    } while (choice == 'y' || choice == 'Y');
    return 0;
}
```

5# Write a menu driven program in C++ to perform all basic arithmetic operation addition, subtraction, multiplication, and division of two given values using function and switch case. Program receives two values and required operation to be performed from the keyboard and display particular result of the required operation.

```
#include <iostream>
using namespace std;
float add(float a, float b)
{
    return a + b;
}
float subtract(float a, float b)
{
    return a - b;
```

```
}
float multiply(float a, float b)
{
    return a * b;
}
float divide(float a, float b)
{
    if (b != 0)
    {
        return a / b;
    }
    else
    {
        cout << "Error! Division by zero is not allowed.";
        return 0;
    }
}

int main()
{
    float num1, num2, result;
    char operation;
    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;
    cout << "Choose an operation:" << endl;
    cout << "1. Addition (+)" << endl;
    cout << "2. Subtraction (-)" << endl;
    cout << "3. Multiplication (*)" << endl;
    cout << "4. Division (/)" << endl;
    cout << "Enter your choice (1/2/3/4): ";
    cin >> operation;
    switch (operation)
    {
    case '1':
        result = add(num1, num2);
        cout << "Result: " << result;
        break;
    case '2':
        result = subtract(num1, num2);
        cout << "Result: " << result;
        break;
    case '3':
        result = multiply(num1, num2);
        cout << "Result: " << result;
        break;
    case '4':
        result = divide(num1, num2);
        cout << "Result: " << result;
        break;
    default:
        cout << "Invalid choice!";
```

```
    }
    return 0;
}
```

6# Define a class Bank Account to represent a bank account. Include the following members: Data Members:

o Name of the depositor o Account Number o Type of account o Balance amount in the account Member Functions: o To assign initial value o To deposit an amount o To withdraw an amount after checking

```
#include <iostream>
#include <string>
using namespace std;
class BankAccount
{
private:
    string depositorName;
    string accountNumber;
    string accountType;
    double balance;
public:
        void setInitialValues(string name, string accNumber, string accType, double
initialBalance)
    {
        depositorName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = initialBalance;
    }
    void deposit(double amount)
    {
        if (amount > 0)
        {
            balance += amount;
            cout << "Deposit successful. Current balance: " << balance << endl;
        }
        else
        {
            cout << "Invalid deposit amount." << endl;
        }
    }
        void withdraw(double amount)
    {
        if (amount > 0 && amount <= balance)
        {
            balance -= amount;
            cout << "Withdrawal successful. Current balance: " << balance << endl;
        }
        else
        {
```

```
            cout << "Invalid withdrawal amount or insufficient balance." << endl;
        }
    }
};
int main()
{
    BankAccount account;
    account.setInitialValues("John Doe", "1234567890", "Savings", 1000.0);
   account.deposit(500.0);
    account.withdraw(200.0);
    account.withdraw(2000.0);
    return 0;
}
```

WEEK 12

1# Write a program in C++ to demonstrate default constructor. Create a class having two data members in the private section. Define a default constructor to initialize these data members to initial value and display these values with the help of member function

```
#include <iostream>
using namespace std;
class MyClass
{
private:
    int num1;
    int num2;

public:
    MyClass()
    {
        num1;
        num2;
    }
    void setValues(int n1, int n2)
    {
        num1 = n1;
        num2 = n2;
    }
    void displayValues()
    {
        cout << "Value of num1: " << num1 << endl;
        cout << "Value of num2: " << num2 << endl;
    }
};
int main()
{
    MyClass obj;
    int input1, input2;
    cout << "Enter value for num1: ";
    cin >> input1;
    cout << "Enter value for num2: ";
    cin >> input2;
    obj.setValues(input1, input2);
    obj.displayValues();
    return 0;
}
```

2# Write a program in C++ to demonstrate parameterized/constructor overloading constructor. Create a class calculator that contains four data members in it. Initialize data members with different values using parameterized constructor and perform various arithmetic operation over these values and display result on to the computer screen.

```
#include <iostream>
using namespace std;
```

```
class Calculator
{
private:
    int num1, num2, num3, num4;
public:
    Calculator(int a, int b, int c, int d)
    {
        num1 = a;
        num2 = b;
        num3 = c;
        num4 = d;
    }
    void performOperations()
    {
        cout << "Addition: " << num1 + num2 << endl;
        cout << "Subtraction: " << num1 - num2 << endl;
        cout << "Multiplication: " << num1 * num2 << endl;
        cout << "Division: " << num1 / num2 << endl;
        cout << "Modulus: " << num3 % num4 << endl;
    }
};
int main()
{
    int a, b, c, d;
    cout << "Enter 4 inputs for calculation: ";
    cin >> a >> b >> c >> d;
    Calculator calc(a, b, c, d);
    calc.performOperations();
    return 0;
}
```

3# Create a class called Triangle that stores the length of the base and height of a right triangle in two private instance variables. Include a constructor that sets these values. Define two functions. The first is hypo( ), which returns the length of the hypotenuse. The second is area ( ), which returns the area of the triangle

```
#include <iostream>
#include <cmath>
using namespace std;

class Triangle
{
private:
    double base;
    double height;
public:
    Triangle(double b, double h)
    {
        base = b;
        height = h;
    }
```

```
    double hypo()
    {
        return sqrt(base * base + height * height);
    }
    double area()
    {
        return 0.5 * base * height;
    }
};
int main()
{
    int b, h;
    cout << "Enter base: ";
    cin >> b;
    cout << "Enter height: ";
    cin >> h;
   Triangle myTriangle(b, h);
    cout << "Hypotenuse: " << myTriangle.hypo() << endl;
    cout << "Area: " << myTriangle.area() << endl;
    return 0;
}
```

4# Create a class for counting the number of objects created and destroyed within various block using constructor and destructors.

```
#include <iostream>
using namespace std;
class ObjectCounter
{
private:
    static int count;
public:
    ObjectCounter()
    {
        count++;
    }
    ~ObjectCounter()
    {
        count--;
    }
    static void displayCount()
    {
        cout << "Number of objects: " << count << endl;
    }
};
int ObjectCounter::count = 0;
int main()
{
    {
        ObjectCounter obj1;
        ObjectCounter obj2;
        ObjectCounter::displayCount();
```

```
    }


        ObjectCounter obj3;
        ObjectCounter::displayCount();
    }
    ObjectCounter::displayCount();
    return 0;
}
```