Problem Set: OOP

1. Suppose, you are working on a software project that involves implementing a geometry library in C++. The library should have a “Shape” class hierarchy to represent different types of shapes, such as “Rectangle” and “Circle”. The “Shape” class has the following specifications:

   Member variables named “name” and “color”, Member functions named setname(),setcolor(), displayinfo().
   The derived class “Rectangle” inherited from “Shape” class has member variables named “length” and “width” and member functions named “setDimensions()”, “calculateArea()” (an override member function) Formula: length * width; “displayInfo()”(an override member function). Similarly the derived class “Circle” that has member variable “radius” and member functions “setRadius()”,“calculateArea()” Formula: 3.14 * radius * radius; (an override member function) “displayInfo()”(an override member function).

   You need to demonstrate method overloading by implementing multiple versions of the “calculateArea()” function in the “Shape” class based on the number and types of parameters. Also, you need to demonstrate method overriding by overriding the “calculateArea()” function in the “Rectangle” and “Circle” classes to provide their specific implementation of calculating the area.

   Once you have implemented the classes, you should create objects of each class, set their attributes using the member functions, and call the “displayInfo()” function on them to display the information of the shapes, including their calculated areas, on the screen. Print appropriate messages to indicate which class's member function is being called during the execution of the program. Apply polymorphism to the given criteria by writing cpp code.

2. Problem Statement: Managing Bank Accounts

   Develop a C++ program to manage bank accounts, utilizing friend functions and constructors. The program should allow users to create and manipulate bank accounts, including functionalities such as depositing and withdrawing funds, checking account balances, and transferring funds between accounts. Implement a BankAccount class with private attributes such as account number, account holder name, and balance. Utilize a friend function to facilitate fund transfers between accounts while ensuring appropriate access to private member variables. Additionally, design constructors to initialize bank accounts with default or user-specified values. Develop a user-friendly interface in the main program to interact with the bank account functionalities, enabling users to perform transactions seamlessly. Ensure robust error handling to manage invalid inputs and account operations effectively.