|  |  |
| --- | --- |
|  | **Practical** |
| **Program:1.1** | A bank wants to create a simple system to manage customer bank accounts. The system should allow customers to perform basic banking operations such as depositing money, withdrawing money, and checking their account balance.Each bank account will need to have an account holder's name, a unique account number, and a balance. Deposits should increase the account balance, while withdrawals should only be allowed if there are sufficient funds in the account. If an attempt is made to withdraw more money than is available, an error message should be displayed. Customers should also have the ability to view their account balance whenever required. The system must be designed using Object-Oriented Programming principles, focusing on creating a simple and efficient solution to manage the accounts effectively. The system should ensure that all account details are secure and accessible only through authorized methods. |
| **Code** | #include<iostream> using namespace std;    class BankAccount { private:  string accountHolder; int accountNumber; float balance;    public:  BankAccount(string name, int accNum, float initialBalance = 0.0) { accountHolder = name; accountNumber = accNum; balance = initialBalance;  }    void deposit(float amount) { balance += amount; cout << "Amount Deposited: $" << amount << endl;  } |
|  | void withdraw(float amount) { if (amount <= balance) { balance -= amount; cout << "Amount Withdrawn: $" << amount << endl;  } else { cout << "Error: Insufficient balance!" << endl;  }  }    void displayBalance() { cout << "Account Holder: " << accountHolder << endl; cout << "Account Number: " << accountNumber << endl; cout << "Balance: $" << balance << endl;  }  };    int main() {  BankAccount acc1("preet", 101, 1000.0); acc1.deposit(500); acc1.withdraw(200); acc1.displayBalance(); acc1.withdraw(2000);    cout << "Name: preet" << endl << "ID: 24CE091" << endl; return 0;  } |
| **Output** |  |

|  |  |
| --- | --- |
|  | **Practical** |
| **Program:1.2** | A small retail store is facing challenges in managing its inventory effectively. The store sells a variety of products, each identified by a unique product ID, a name, the available quantity instock, and the price per unit. To streamline their operations, the store needs a basic system tomanage this inventory efficiently. The system must provide the ability to add new products to the inventory, ensuring that each product has its ID, name, quantity, and price properly recorded. Additionally, the system should allow the store staff to update the quantity of any existing product, such as when new stock arrives or when items are sold. Another essential feature of the system is the calculation of the total value of all products in the inventory, which is determined by multiplying the quantity of each product by its price and summing these values for all products. |
| **Code** | #include<iostream> using namespace std;    class Product { private:  int productID; string productName; int quantity; float price;    public:  Product(int id, string name, int qty, float p) { productID = id; productName = name; quantity = qty; price = p;  }    void updateQuantity(int qty) { quantity += qty;  } |
|  | float calculateValue() { return quantity \* price;  }    void displayDetails() { cout << "ID: " << productID << ", Name: " << productName  << ", Quantity: " << quantity << ", Price: $" << price << endl;  }  };    int main() {  Product p1(101, "Laptop", 10, 50000); p1.displayDetails(); p1.updateQuantity(5); p1.displayDetails(); cout << "Total Inventory Value: $" << p1.calculateValue() << endl;    cout << "Name: PREET" << endl << "ID: 24CE091" << endl; return 0;  } |
| **Output** |  |