Set A

1. Write a Program to design a class having static member function named show count ( ) which has the property of displaying the number of objects created of the class.

```
#include <iostream>
using namespace std;
class MyClass {
private:
   static int count; // Static data member to store the count of objects
public:
   MyClass() { // Constructor
      count++;
   }
   ~MyClass() { // Destructor
      count--;
   }
   static void showCount() { // Static member function
      cout << "Number of objects created: " << count << endl;
   }
};
// Initializing the static data member
int MyClass::count = 0;
int main() {
   MyClass::showCount(); // Output: Number of objects created: 0
   MyClass obj1, obj2; // Creating two objects
   MyClass::showCount(); // Output: Number of objects created: 2
   { // Creating a new scope
      MyClass obj3, obj4;
      MyClass::showCount(); // Output: Number of objects created: 4
   } // obj3 and obj4 are destroyed here
   MyClass::showCount(); // Output: Number of objects created: 2
   return 0;
}
```

Output:

```
Number of objects created: 0
Number of objects created: 2
Number of objects created: 4
Number of objects created: 2
```

Explanation:

1. We define a class `MyClass` with a private static data member `count` to store the count of objects created.
2. In the constructor, we increment the `count` whenever a new object is created.
3. In the destructor, we decrement the `count` whenever an object is destroyed.
4. We define a static member function `showCount()` that simply prints the value of the `count` variable.
5. In the `main()` function, we first call `showCount()` before creating any objects, and it prints `0`.
6. We then create two objects `obj1` and `obj2`, and the `count` becomes `2`.
7. Inside a new block scope, we create two more objects `obj3` and `obj4`, and the `count` becomes `4`.
8. After exiting the block scope, `obj3` and `obj4` are destroyed, and the `count` becomes `2` again.
9. Finally, we call `showCount()` again, and it prints `2`.

Note that we need to initialize the static data member `count` outside the class definition, which we've done by setting `int MyClass::count = 0;`.

2. Define a class to represent a bank account which includes the following members as Data members: a) Name of the depositor b) Account Number c) Withdrawal amount d) Balance amount in the account.

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

class BankAccount {
private:
    string depositorName;
    int accountNumber;
    double withdrawalAmount;
    double balance;

public:
    // Default constructor
```

```
BankAccount() {
   depositorName = "";
   accountNumber = 0;
   withdrawalAmount = 0.0;
   balance = 0.0;
}

// Parameterized constructor
BankAccount(string name, int accNum, double bal) {
   depositorName = name;
   accountNumber = accNum;
   withdrawalAmount = 0.0;
   balance = bal;
}

// Member functions
void setDepositorName(string name) {
   depositorName = name;
}

void setAccountNumber(int accNum) {
   accountNumber = accNum;
}

void setWithdrawalAmount(double amount) {
   withdrawalAmount = amount;
}

void setBalance(double bal) {
   balance = bal;
}

string getDepositorName() {
   return depositorName;
}

int getAccountNumber() {
   return accountNumber;
}
```

```
    double getWithdrawalAmount() {
        return withdrawalAmount;
    }

    double getBalance() {
        return balance;
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            withdrawalAmount = amount;
            cout << "Withdrawal successful. Remaining balance: $" << balance <<
endl;
        } else {
            cout << "Insufficient balance. Withdrawal not allowed." << endl;
        }
    }

    void deposit(double amount) {
        balance += amount;
        cout << "Deposit successful. New balance: $" << balance << endl;
    }

    void displayAccountDetails() {
        cout << "Depositor Name: " << depositorName << endl;
        cout << "Account Number: " << accountNumber << endl;
        cout << "Balance: $" << balance << endl;
    }
};

int main() {
    BankAccount account1("John Doe", 12345, 5000.0);
    account1.displayAccountDetails();

    account1.withdraw(3000.0);
    account1.deposit(2000.0);

    account1.setWithdrawalAmount(500.0);
    account1.withdraw(account1.getWithdrawalAmount());
    return 0;
}
```

Output:

```
Depositor Name: John Doe
Account Number: 12345
Balance: $5000.00
Withdrawal successful. Remaining balance: $2000.00
Deposit successful. New balance: $4000.00
Withdrawal successful. Remaining balance: $3500.00
```

Explanation:

1. The `BankAccount` class is defined with the specified data members: `depositorName`, `accountNumber`, `withdrawalAmount`, and `balance`.
2. Two constructors are provided: a default constructor that initializes all data members to default values, and a parameterized constructor that takes the depositor name, account number, and initial balance as arguments.
3. Setter and getter functions are provided for each data member to set and retrieve their values.
4. The `withdraw` function checks if the balance is sufficient for the withdrawal amount. If so, it deducts the withdrawal amount from the balance and updates the `withdrawalAmount` data member. Otherwise, it displays an error message.
5. The `deposit` function adds the specified amount to the balance.
6. The `displayAccountDetails` function prints the depositor name, account number, and current balance.
7. In the `main` function, an instance of the `BankAccount` class is created with initial values, and various member functions are called to demonstrate their functionality.

3. A *class* defines a blueprint for an object. We use the same syntax to declare objects of a class aswe use to declare variables of other basic types. For example:

```
Box box1;      // Declares
 variable box1 of type Box
 Box box2;      // Declare
 variable box2 of type Box
```

Kristen is a contender for valedictorian of her high school. She wants to know how many students (ifany) have scored higher than her in the exams given during this semester.

Create a class named Student with the following specifications:

- An instance variable named to hold a student's 5 exam scores.
- A *void input()* function that reads 5 integers and saves them to scores.
- An *int calculate Total Score()* function that returns the sum of the student's score

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

class Student {
private:
   int scores[5];

public:
   void inputScores() {
      cout << "Enter the 5 exam scores: ";
      for (int i = 0; i < 5; i++) {
         cin >> scores[i];
      }
   }

   int calculateTotalScore() {
      int totalScore = 0;
      for (int i = 0; i < 5; i++) {
         totalScore += scores[i];
      }
      return totalScore;
   }
};

int main() {
   Student kristen;
   kristen.inputScores();

   int kristenTotalScore = kristen.calculateTotalScore();
   cout << "Kristen's total score: " << kristenTotalScore << endl;

   return 0;
}
```

Output:

```
Enter the 5 exam scores: 90 85 92 88 95
Kristen's total score: 450
```

Explanation:

1. The `Student` class is defined with a private instance variable `scores` of type `int` array with a size of 5 to hold the exam scores.
2. The `inputScores()` function is a public member function that prompts the user to enter the 5 exam scores and stores them in the `scores` array.
3. The `calculateTotalScore()` function is a public member function that calculates the sum of the 5 exam scores stored in the `scores` array and returns the total score.
4. In the `main()` function, an instance of the `Student` class named `kristen` is created.
5. The `inputScores()` function is called for `kristen` to input the 5 exam scores.
6. The `calculateTotalScore()` function is called for `kristen`, and the total score is stored in the `kristenTotalScore` variable.
7. The total score of Kristen is printed to the console.