**C++ Programming**

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

**Question 1: Problem Statement:**

**Create a class BankAccount that models a simple bank account. The class should support the following:**

* **Private members: accountNumber, accountHolder, balance**
* **Static member: interestRate (common for all accounts)**
* **Constructors:**
  + **Default constructor that initializes values to zero/default**
  + **Parameterized constructor to initialize all members**
  + **Copy constructor to create a new object from an existing one**
* **A static function to update the interest rate**
* **A public function display() to show all details**
* **Demonstrate all of the above in main() by:**
  + **Creating multiple objects**
  + **Updating static interest rate**
  + **Copying an account using copy constructor**

**Expected Concepts:**

* **Constructor overloading**
* **Access specifiers**
* **Static members & functions**
* **Copy constructor**
* **cin / cout usage**

**Code:**

#include <iostream>

#include <string>

using namespace std;

class BankAccount {

private:

int accountNumber;

string accountHolder;

double balance;

static float interestRate;

public:

BankAccount() : accountNumber(0), accountHolder("Unknown"), balance(0.0) {}

BankAccount(int accNo, const string& holder, double bal) {

accountNumber = accNo;

accountHolder = holder;

balance = bal;

}

BankAccount(const BankAccount& other) {

accountNumber = other.accountNumber;

accountHolder = other.accountHolder;

balance = other.balance;

}

static void setInterestRate(float rate) {

interestRate = rate;

}

void display() const {

cout << "Account Number: " << accountNumber << endl;

cout << "Account Holder: " << accountHolder << endl;

cout << "Balance: ₹" << balance << endl;

cout << "Interest Rate: " << interestRate << "%" << endl;

cout << "-------------------------" << endl;

}

};

float BankAccount::interestRate = 3.5;

int main() {

BankAccount acc1(1001, "Rahul Sharma", 15000.0);

BankAccount acc2(1002, "Sneha Mehta", 23000.0);

cout << "Initial Accounts:\n";

acc1.display();

acc2.display();

BankAccount::setInterestRate(4.2);

cout << "After Updating Interest Rate:\n";

acc1.display();

acc2.display();

BankAccount acc3 = acc2;

cout << "Copied Account:\n";

acc3.display();

BankAccount acc4;

cout << "Default Constructed Account:\n";

acc4.display();

return 0;

}

**Question 2: Design a Student class to manage basic student details. The class should include:**

* **Private members: rollNo, name, marks**
* **Inline member functions to:**
  + **Set student details**
  + **Display student details**
* **An overloaded functioncalculateGrade():**
  + **No arguments → use existing marks**
  + **One float argument → use this as the marks for grade**
* **Grade logic:**
  + **>=90: A**
  + **>=75: B**
  + **>=60: C**
  + **Else: D**

**Expected Concepts:**

* **Inline functions**
* **Function overloading**
* **Conditional logic**
* **Use of access specifiers**
* **Use of cin, cout**

**Code:**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

int rollNo;

string name;

float marks;

public:

void setDetails(int r, const string& n, float m) {

rollNo = r;

name = n;

marks = m;

}

void displayDetails() const {

cout << "Roll No: " << rollNo << endl;

cout << "Name: " << name << endl;

cout << "Marks: " << marks << endl;

}

void calculateGrade() const {

cout << "Grade: " << getGrade(marks) << endl;

}

void calculateGrade(float customMarks) const {

cout << "Grade (based on custom marks " << customMarks << "): " << getGrade(customMarks) << endl;

}

private:

char getGrade(float m) const {

if (m >= 90)

return 'A';

else if (m >= 75)

return 'B';

else if (m >= 60)

return 'C';

else

return 'D';

}

};

int main() {

Student s1;

int roll;

string name;

float marks;

cout << "Enter student roll number: ";

cin >> roll;

cin.ignore();

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter marks: ";

cin >> marks;

s1.setDetails(roll, name, marks);

cout << "\nStudent Info:\n";

s1.displayDetails();

s1.calculateGrade();

s1.calculateGrade(82.5);

return 0;

}