C-DAC MUMBAI
Object Oriented Programming Using C++
Assigment-2

Q1. Create a class called Student with the following private data members:
1. name (string): to store the name of the student.
2. rollNumber (int): to store the roll number of the student.
3. marks (float): to store the marks obtained by the student.
4. grade (char): to store the grade calculated based on the marks.
Implement getter and setter member functions for each data member
Create a member function calculateGrade() that calculates the grade based on the following
grading scale:
A: 90-100
B: 80-89
C: 70-79
D: 60-69
F: Below 60
Implement a menu-driven program in the main() function with the following options:
1. Accept Information
2. Display information
3. Calculate Grade
4. Exit the program.

-->>

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

class Student {
private:
    string name;
    int rollNumber;
    float marks;
    char grade;

public:
    void setName(string n) { name = n; }
    void setRollNumber(int r) { rollNumber = r; }
    void setMarks(float m) { marks = m; }
    string getName() { return name; }
    int getRollNumber() { return rollNumber; }
    float getMarks() { return marks; }
    char getGrade() { return grade; }

    void calculateGrade() {
        if (marks >= 90) grade = 'A';
        else if (marks >= 80) grade = 'B';
        else if (marks >= 70) grade = 'C';
        else if (marks >= 60) grade = 'D';
        else grade = 'F';
    }

    void display() {
        cout << "Name: " << name << ", Roll Number: " << rollNumber
             << ", Marks: " << marks << ", Grade: " << grade << endl;
    }
};

int main() {
    Student s;
    int choice;

    while (true) {
        cout << "\nMenu:\n1. Accept Information\n2. Display Information\n3.
Calculate Grade\n4. Exit\nEnter choice: ";
        cin >> choice;

        if (choice == 1) {
            string name;
            int rollNumber;
            float marks;
            cout << "Enter Name: ";
            cin >> name;
            cout << "Enter Roll Number: ";
            cin >> rollNumber;
            cout << "Enter Marks: ";
            cin >> marks;
            s.setName(name);
            s.setRollNumber(rollNumber);
            s.setMarks(marks);
        } else if (choice == 2) {
            s.display();
        } else if (choice == 3) {
            s.calculateGrade();
            cout << "Grade calculated successfully.\n";
        } else if (choice == 4) {
            break;
        } else {
            cout << "Invalid choice!\n";
        }
    }

    return 0;
}
```

Q2. Create a C++ program for a simple banking system. You need to implement a class called

```
1. BankAccount with the following data members:
2. accountNumber (int): The account number of the bank account.
3. accountHolderName (string): The name of the account holder.
4. balance (double): The current balance in the account.
The BankAccount class should have the following member functions:
1. Getter and Setter Methods:
2. deposit method: A method that allows the user to deposit money into the account.
It
should take an amount as a parameter and update the balance accordingly.
3. withdraw method: A method that allows the user to withdraw money from the
account. It should take an amount as a parameter and update the balance. Make sure
to check if there is sufficient balance before allowing the withdrawal.
4. displayAccountDetails method: A method that displays the account details (
account number, account holder name, and balance).
Now, create a menu-driven program in the `main` function that allows the user to
perform the
following operations:
1. Deposit money into an existing account.
2. Withdraw money from an existing account.
3. Display the account details.
4. Exit the program.

-->>

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

class BankAccount {
private:
    int accountNumber;
    string accountHolderName;
    double balance;

public:
    void setAccountDetails(int accNum, string name, double bal) {
        accountNumber = accNum;
        accountHolderName = name;
        balance = bal;
    }

    void deposit(double amount) {
        balance += amount;
        cout << "Deposited: " << amount << ". New Balance: " << balance << endl;
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            cout << "Withdrawn: " << amount << ". Remaining Balance: " << balance <<
endl;
        } else {
            cout << "Insufficient balance!\n";
        }
    }

    void displayAccountDetails() {
        cout << "Account Number: " << accountNumber
             << ", Account Holder: " << accountHolderName
             << ", Balance: " << balance << endl;
    }
};

int main() {
    BankAccount account;
    int choice;

    account.setAccountDetails(1945, "Yugandhar Deshmukh", 0);

    while (true) {
        cout << "\nMenu:\n1. Deposit Money\n2. Withdraw Money\n3. Display Account
Details\n4. Exit\nEnter choice: ";
        cin >> choice;

        if (choice == 1) {
            double amount;
            cout << "Enter amount to deposit: ";
            cin >> amount;
            account.deposit(amount);
        } else if (choice == 2) {
            double amount;
            cout << "Enter amount to withdraw: ";
            cin >> amount;
            account.withdraw(amount);
        } else if (choice == 3) {
            account.displayAccountDetails();
        } else if (choice == 4) {
            break;
        } else {
            cout << "Invalid choice!\n";
        }
    }

    return 0;
}
```

Q3. Imagine you are tasked with creating a program to simulate a toll booth. The toll booth
keeps track of the number of vehicles that have passed through it and the total

amount of money
collected. You need to implement a class TollBooth with appropriate data members and

member functions to accomplish this.
Here are the details for the TollBooth class:
1. Data Members: - totalVehicles: An integer to keep track of the total number of vehicles that have
passed through the toll booth. - totalRevenue: A double to store the total revenue collected from tolls.
2. Member Functions:
1. void reset(): Resets both the totalVehicles and totalRevenue to zero.
2. void vehiclePayingToll(int vehicleType, double tollAmount): Accepts an integer vehicleType and a double tollAmount. The vehicleType represents the type of car (1 for standard car, 2 for truck, 3 for bus). The function should increment totalVehicles
by 1 and add tollAmount to totalRevenue based on the car type.
3. int getTotalVehicles() : A getter method that returns the total number of vehicles
passed through the booth.
4. double getTotalRevenue() : A getter method that returns the total revenue collected.
3. Menu-Driven Program:
Write a menu-driven program in main() that allows the user to interact with the TollBooth
class: - Display a menu with the following options:
1. Add a standard car and collect toll
2. Add a truck and collect toll
3. Add a bus and collect toll
4. Display total cars passed
5. Display total revenue collected
6. Reset booth statistics
7. Exit - Implement the logic for each menu option using appropriate member functions of the
TollBooth class. - Continue displaying the menu until the user chooses to exit. - Define a fixed toll amount for each type of car (e.g., $2 for standard cars, $5 for trucks, $10
for buses).

-->>

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

class TollBooth {
private:
    int totalVehicles;
    double totalRevenue;

public:
    TollBooth() : totalVehicles(0), totalRevenue(0.0) {}
    void reset() {
        totalVehicles = 0;
        totalRevenue = 0.0;
        cout << "Booth statistics reset \n";
    }

    void vehiclePayingToll(int vehicleType) {
        double tollAmount = 0;

        if (vehicleType == 1) tollAmount = 20;  // Standard car
        else if (vehicleType == 2) tollAmount = 50;  // Truck
        else if (vehicleType == 3) tollAmount = 100;  // Bus

        totalVehicles++;
        totalRevenue += tollAmount;
        cout << "Toll collected: Rs " << tollAmount << endl;
    }

    int getTotalVehicles() { return totalVehicles; }
    double getTotalRevenue() { return totalRevenue; }
};

int main() {
    TollBooth booth;
    int choice;

    while (true) {
        cout << "\nMenu:\n1. Add Standard Car\n2. Add Truck\n3. Add Bus\n4. Display
Total Cars Passed\n5. Display Total Revenue\n6. Reset Booth Statistics\n7.
Exit\nEnter choice: ";
        cin >> choice;

        if (choice == 1) {
            booth.vehiclePayingToll(1);
        } else if (choice == 2) {
            booth.vehiclePayingToll(2);
        } else if (choice == 3) {
            booth.vehiclePayingToll(3);
        } else if (choice == 4) {
            cout << "Total Vehicles: " << booth.getTotalVehicles() << endl;
        } else if (choice == 5) {
            cout << "Total Revenue: Rs " << booth.getTotalRevenue() << endl;
        } else if (choice == 6) {
            booth.reset();
        } else if (choice == 7) {
            break;
        } else {
            cout << "Invalid choice!\n";
        }
    }

    return 0;
}
```

Q4. You are tasked with creating an Employee Payroll Management System in C++. Your program should allow the user to perform the following tasks through a menu-driven interface:
1. Add a new employee: - Create a class Employee with the following private data members: - int empID (Employee ID) - string empName (Employee Name) - double empSalary (Employee Salary) - Include appropriate getter and setter methods for these data members. - Ensure that the Employee ID is unique for each employee.
2. Calculate the gross salary for an employee: - Create a member function calculateGrossSalary in the Employee class. - The gross salary should be calculated based on the following rules: - If the employee's salary is less than or equal to 5000, add a 10% bonus. - If the employee's salary is greater than 5000 but less than or equal to 10000, add a
15% bonus. - If the employee's salary is greater than 10000, add a 20% bonus. - Display the gross salary for the chosen employee.
3. Display the employee details: - Create a member function displayEmployeeDetails in the Employee class to display
all the details of an employee (ID, Name, Salary, and Gross Salary).
4. Update employee information: - Allow the user to update the employee's name and salary using setter methods.
5. Exit the program.

-->>

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

class Employee {
private:
    int empID;
    string empName;
    double empSalary;
    double grossSalary;

public:
    void setEmployeeDetails(int id, string name, double salary) {
        empID = id;
        empName = name;
        empSalary = salary;
    }

    void calculateGrossSalary() {
        if (empSalary <= 5000)
            grossSalary = empSalary + (0.1 * empSalary);
        else if (empSalary <= 10000)
            grossSalary = empSalary + (0.15 * empSalary);
        else
            grossSalary = empSalary + (0.2 * empSalary);

        cout << "Gross Salary: " << grossSalary << endl;
    }

    void displayEmployeeDetails() {
        cout << "ID: " << empID << ", Name: " << empName << ", Salary: " <<
empSalary << ", Gross Salary: " << grossSalary << endl;
    }

    void updateEmployeeDetails(string name, double salary) {
        empName = name;
        empSalary = salary;
        cout << "Details updated successfully \n";
    }
};

int main() {
    Employee emp;
    int choice;

    emp.setEmployeeDetails(101, "Yugandhar", 75000);

    while (true) {
        cout << "\nMenu:\n1. Add/Update Employee\n2. Calculate Gross Salary\n3.
Display Employee Details\n4. Exit\nEnter choice: ";
        cin >> choice;

        if (choice == 1) {
            string name;
            double salary;
            cout << "Enter Name: ";
            cin >> name;
            cout << "Enter Salary: ";
            cin >> salary;
            emp.updateEmployeeDetails(name, salary);
        } else if (choice == 2) {
            emp.calculateGrossSalary();
        } else if (choice == 3) {
            emp.displayEmployeeDetails();
        } else if (choice == 4) {
            break;
        } else {
            cout << "Invalid choice!\n";
        }
    }
    return 0;
}
```