

Question X (30 marks)

For simplicity, all preprocessing (<include>) directives and using directives are ignore in this question. You do not need to provide them in your answers as well.

- (a) Mary is new to O-O programming. She has written a simple program that models shops (class Shop) and customers (class Customer), and how the shops earn profits from customers.

In the main function she adds 2 customer objects and 2 shop objects, sets the initial money amount of each customer, and calls the earn function of the shops to earn and get money from the customers. Finally she shows the customers' remaining total money and the shops' total profits.

Below are the source files written by Mary. Mary's expected program outputs are shown in the comments next to the output statements in the main function.

Main.cpp

```
int main()
{
    Customer c1,c2;
    Shop s1,s2;

    c1.set(200); //c1 has $200 initially
    c2.set(200); //c2 has $200 initially

    s1.earn(c1, 30); //s1 earns and get $30 from c1
    s2.earn(c1, 40); //s2 earns and get $40 from c1
    s2.earn(c2, 50); //s2 earns and get $50 from c2

    cout << c1.getAmount() << endl; //expected output: 130
    cout << c2.getAmount() << endl; //expected output: 150
    cout << s1.getProfit() << endl; //expected output: 30
    cout << s2.getProfit() << endl; //expected output: 90

    return 0;
}
```

Customer.h

```
class Customer {
private:
    int totalMoney;
public:
    void set(int money);
    void spend(int value);
    int getAmount();
};
```

Shop.h

```
class Shop {
private:
    int totalProfit = 0;
public:
    void earn(Customer c, int value);
    int getProfit();
};
```

Customer.cpp

```
void Customer::set(int money) {
    totalMoney = money;
}

void Customer::spend(int value) {
    totalMoney -= value;
}

int Customer::getAmount() {
    return totalMoney;
}
```

Shop.cpp

```
void Shop::earn(Customer c, int value) {
    totalProfit += value;
    c.spend(value);
}

int Shop::getProfit() {
    return totalProfit;
}
```

Mary can compile the source files and build the executable successfully. However, when she runs the program, the outputs are wrong. They are not the same as her expectations.

(i) Explain the mistake that Mary has made. (3 marks)

(ii) What are the outputs of Mary's program? (3 marks)

(iii) How could the problem be fixed? Write down all code that has to be revised. (4 marks)

- (iv) Mary's program can further be improved by adding a `Customer` constructor that initializes the customer's total money amount. Write down all code that has to be revised. (4 marks)

- (b) Design the class `Group` that models a group of customers joined together for group-purchasing.

Note:

- For simplicity, we assume that a purchasing group has at most 10 customers only.
Also, the money to pay for each group purchase is divisible by the count of group members so that the payment by each member is a whole number.
- The same customer should be able to purchase individually and should be able to join 1 or more purchasing groups. For example, if the customer has \$100 initially, then after spending \$40 individually, and \$35 in one group purchase, and \$12 in another group purchase, he should have \$13 left.
- Other than the `Group` class, you also need to
 - (a) Add an `earn` function in the `Shop` class that handles earning from group purchase
 - (b) Rewrite the `main` function for testing. It should contain at least the following steps:
 - create 2 customer objects: `c1` and `c2`.
 - create 1 purchasing group `g1` and have both `c1` and `c2` joining `g1`
 - create 1 shop object: `s1`
 - the shop `s1` earns \$300 from `g1` (i.e. each of `c1` and `c2` spends \$150)
 - the shop `s1` earns \$30 from `c1`
 - display the total profit of `s1`
 - display the total remaining money of `c1`
- You may assume that the revised code given in part (iii) and (iv) have already been applied.

Write your code for the `Group` class (`Group.h` and `Group.cpp`), as well as the new `earn` function in `Shop.cpp`, and the new `main` function. (16 marks)