Computer Science & Engineering Department
I. I. T. Kharagpur

Software Engineering: CS20006/CS20202

Assignment – 2: Classes and Object-oriented Programming *Marks: 30*

Assign Date: *7th February, 2022* Submit Date: *23:55, 14th February, 2022*

Instructions: Please solve the questions using pen and paper and scan the images. Every image should contain your roll number and name.

1. Write a singleton class `PrimalityTest` which provide the facility perform primality testing in a calling program. The class maintains a buffer of known primes from the previous tests. You have to implement the following functions:

   `newTest()`: This static function will be accessed in the main to get an instance of `PrimalityTest`. It is also responsible for resizing the buffer.

   `test()`: This function is responsible for the primality test using the buffer. It should also populate the necessary primes in the buffer, and report error if buffersize is not enough for testing the given input.

   Constructor should allocate necessary memory and initialize members, while destructor should perform the necessary cleanup.

   Use the following template for writing your functionality.

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

class PrimalityTest {
    static PrimalityTest* _myTest;
    int nStored;      // Number of Stored Primes
    int *primes;      // Pointer to Buffer
    int bufsize;      // Size of the buffer
    PrimalityTest(int bufsize);                    // Implement
    ~PrimalityTest();                              //Implement
public:
    static PrimalityTest& newTest(int bufsize = 100);  //Implement
    void test(int n);                                  //Implement
};

PrimalityTest* PrimalityTest::_myTest = NULL;


int main() {
    PrimalityTest::newTest().test(2958);
    PrimalityTest::newTest().test(823);
    PrimalityTest::newTest().test(83479);

    return 0;
}
```

   Implement the member functions along with comments providing documentation for of internal functionality. For each function, 50% marks will be for correctness, 25% marks for conciseness and efficiency of code, and 25% marks for code clarity and documentation. **[4 + 4 + 4 = 12]**

2. Design a C++ program for managing the shopping basket of an online customer. You have to implement the following classes: `Customer`, `ProductItem`, `Order`, and `ShoppingBasket`. `Customer` and `ProductItem` are self explanatory (please see the attached code). `Order` has a customer and a list of product items. `ShoppingBasket` for a customer is a list of orders.

   The following functionality should be implemented:

- For new instances of all classes, a unique id (integer) should be generated based on the current number of instances of that class. The counting of instances should be done by the class.
- An order can have a fixed size list of products, along with counts. Products cannot be removed from an order. Also, orders can be finalized after which, they cannot be changed.
- `Operators` + and * should be over overloaded for `ProductItems` and `Orders`, so as to implement addition of certain counts of a product. Please see the code for usage syntax.
- The orders in a shopping basket can be added or removed. The operators + and - should be overloaded appropriately to perform addition and deletion (please see code for usage).

Your program should have the following features:

- All the functionality should be implemented through member functions of classes.
- All data should be private within the classes.
- There should minimal copy of objects. Also, use of pointers should be minimised to facilitate code readability.

Your tasks are:

- Provide appropriate definition of member variables with explanations to be provided in code comments (Fill in the blanks).
- Provide implementations for the member function prototypes provided in the code below. Implement other member functions if necessary.
- Provide appropriate initializations for static member variables. Note that you cannot define any global variables.

Find the code below for reference. For each class, 50% marks will be for correctness, 25% marks for conciseness and efficiency of code, and 25% marks for code clarity and documentation. **[4 + 4 + 4 + 6 = 18]**

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

class Customer {
    _____ name;
    _____ id;
    _____ NumCustomer;
    public:
    Customer(string name = "NA");
    ~Customer();
    friend ostream& operator << (ostream &os, const Customer & cust);
    friend istream& operator >> (istream &is, Customer & cust);
};

class ProductItem {
    _____ title;
    _____ id;
    static int NumProductItem;
    float price;
    int copies;
    public:
    ProductItem(string title = "NA", float price = 0);
    ~ProductItem();
    friend ostream& operator << (ostream &os, const ProductItem & pi);
    friend istream& operator >> (istream &is, ProductItem & pi);
    ProductItem& operator* (int a);
    ProductItem& operator= (ProductItem & pi);
};

class Order {
    _____ id;
    _____ NumOrder;
    Customer c;
    vector<ProductItem> prods;
    public:
    Order(Customer &in_c);
    ~Order();
    int getid();
    friend ostream& operator << (ostream &os, const Order & o);
    friend istream& operator >> (istream &is, Order & o);
    Order & operator+ (ProductItem &p);
    Order & operator= (Order &o);
};

class ShoppingBasket {
    _____ id;
    _____ NumBasket;
    Customer c;
    list<Order> orders;
    public:
    ShoppingBasket(Customer &in_c);
    ~ShoppingBasket();
    friend ostream& operator << (ostream &os, const ShoppingBasket & sb);
    friend istream& operator >> (istream &is, ShoppingBasket & sb);
    ShoppingBasket & operator+ (Order &p);
    ShoppingBasket & operator- (int orderid);
    ShoppingBasket & operator= (ShoppingBasket &sb);
};

int main() {
    //create a customer
    Customer *c = new Customer("Nikhil");
    // create a product
    ProductItem *p = new ProductItem("Something");
    // create am Order
    Order *o = new Order(*c);

    // add 10 copies of p to order o
    Order &oref = *o;
    oref = oref + *p * 10;

    // create a shopping basket
    ShoppingBasket *s = new ShoppingBasket(*c);
    ShoppingBasket &shop = *s;
    shop = shop + oref;
    shop = shop - oref.getid();
    return 0;
}
```