

## Sample Midterm

**Problem 1.** Consider the following two classes.

```
1
2 class Order
3 {
4     public:
5         Order();
6         Order(string i; double p; int v);
7
8         void ReadOrder();
9
10        string get_item() const;
11        double get_price() const;
12        int get_volume() const;
13
14    private:
15        string item;
16        double price;
17        int volume;
18 }
19
20
21
22 class Customer
23 {
24     public:
25         Customer(string n)
26
27     private:
28         string name;
29         vector<Order> orders;
30 }
```

- (a) Use these two classes to explain the concept of data encapsulation.
- (b) In the class `Order`, what is the difference between the constructor `Order()`, and `Order(string prod; double p; int v)`?
- (c) Write the implementation of the constructor `Order(string prod; double p; int v)`.
- (d) We want our program to handle any number of customers. How would you store all the customer objects?

Continued on next page →

- (e) Write a short program segment that let's the user enter new customers and then stores them.
  
- (f) The goal is to write a program segment that promotes all customers with orders totalling \$100,000 to VIP customers.  
What additional data fields or member functions would you require for the class `customer` ?  
Keep in mind that we want to keep the data encapsulated.
  
- (g) Using your answer to the previous question, write a program segment that promotes all customers with orders totalling \$100,000 to VIP customers.
  
- (h) Write a short program segment that prints the names of all VIP customers.

**Problem 2.** Consider the following program.

```
1
2 #include <iostream>
3 using namespace std;
4
5 class pClass {
6
7     public:
8
9         int num;
10
11         pClass *left;
12         pClass *right;
13
14         pClass(int n);
15
16 };
17
18 pClass::pClass(int n)
19 {
20     num = n;
21     left = NULL;
22     right = NULL;
23 }
24
25 //=====
26
27 int main() {
28
29     pClass *p1 = new pClass(1);
30     pClass *p2 = new pClass(2);
31     pClass *p3 = new pClass(3);
32     pClass *p4 = new pClass(4);
33
34     p1->left = p2;
35     p1->right = p3;
36
37     p2->left = p4;
38     p2->right = p3;
39
40     p3->left = p2;
41     p3->right = p4;
42
43     p4->right = p4;
44     p4->left = p1;
45
46     pClass *current = p1;
47     pClass *temp = NULL;
48
49     while (current != NULL)
50     {
51         cout << "I am at" << current->num << "\n";
```

Continued on next page →

```
52     if (current->left != NULL)
53     {
54         temp = current->left;
55         current->left = NULL;
56         current = temp;
57     }
58     else if (current->right != NULL)
59     {
60         temp = current->right;
61         current->right = NULL;
62         current = temp;
63     }
64     }
65     else //both left and right are NULL
66     {
67         current = NULL;
68     }
69
70 }//while
71
72 }
```

- (a) Draw a picture indicating what objects the pointers p1, p2, p3, and p4 point to, as well as how the objects are connected through pointers, just before the program enters the while-loop.
- (b) Trace the program and determine its output.