COEN 79

Object-Oriented Programming and Advanced Data Structures

Assignment #3

Name: **Andy Qin** Date: **October 17th, 2020**

2 points per question, unless noted

1. The sequence's insert member function normally puts a new item before the current item. What does insert do if there is no current item?

If there is no current item, the insert function will insert the item to the front of the list.

This happens either the index is 0 and count is 0 or that the index is bigger than or equal to the count

2. (4 pts) Modify the following code to generate the given output. Do not modify the `main` function.

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

class box {

public:
// Constructor definition
box(double l = 2.0, double b = 2.0, double h = 2.0) {
length = l;
breadth = b;
height = h;
}

double volume() {
return length * breadth * height; }

private:
double length;
double breadth;
double height;
};

int main(void) {
box Box1(3.3, 1.2, 1.5); // Declare box1

box Box2(8.5, 6.0, 2.0); // Declare box2

return 0;
}
```

```
Output:
Number of box objects created so far: 1
Number of box objects created so far: 2
```

In the private variables, we add a new static int called "Count", which represents the amount of box objects created. It is static so we can keep track the count everytime a new box is created(so it will not reset) and can be declared globally(see addition below)

```
private:
double length;
double breadth;
double height;
Static int Count;
```

In the constructor, we add a change in "Count", and a println message(see addition below)

```
box(double l = 2.0, double b = 2.0, double h = 2.0) {
length = l;
breadth = b;
height = h;
Count++; // increment count each time a new box is created
Cout << "Number of box objects created so far: " << Count << endl; // print the message with count
}
```

Since we do not want reset count when create a new box, we initialize its value outside the class:

```
Int box :: count = 0; // initialize count to 0;
```

3. (6 pts) In the following code, indicate if the selected lines are legal or illegal:

```
#include <iostream>

   class small
   {
   public:
       small() {size = 0;};
       void k() const;
       void h(int i);
       friend void f(small z);

   private:
       int size;
   };
   void small::k() const
   {
       small x, y;
       x = y; // LEGAL/ILLEGAL?
```

Legal, this is part of the value semantics: assignment operator maybe used

```
       x.size = y.size; // LEGAL/ILLEGAL?
```

Legal, since k() is a member function, it is allowed to access private variables. Since y.size = 0 and x.size = 0, x.size is not changed.

```
       x.size = 3; // LEGAL/ILLEGAL?
```

legal, x is not the original object using the k() when called, so nothing is changed to the original data;

```
   };
```

```
void small::h(int i)
{

};

void f(small z)
{
    small x, y;
    x = y; // LEGAL/ILLEGAL?
    Legal, this is part of the value semantics
    x.size = y.size; // LEGAL/ILLEGAL?
    Legal, it is a friend non-member function with no const qualifier
    x.size = 3; // LEGAL/ILLEGAL?
    legal, since the function is a non-constant friend function, private
    variables can be altered.
    x.h(42); // LEGAL/ILLEGAL?
    Legal, member functions are allowed to use.
};


int main() {

    small x, y;
    x = y; // LEGAL/ILLEGAL?
    Legal, this is part of the value semantics
    x.size = y.size; // LEGAL/ILLEGAL?
    Illegal, can not access private variables outside
    x.size = 3; // LEGAL/ILLEGAL?
    Illegal, can not access private variables outside
    x.h(42); // LEGAL/ILLEGAL?
    Legal, objects are allowed to use member functions

    return 0;
}
```

4. (4 pts) We create an array of `fruit` in the `main` function. How can we make sure that for all the items in array `fruit_ptr` the values of `weight` and `color` are equal to 1 and 2, respectively? Please show your solution. Do not modify the main function.

```
class fruit {
private:
int weight;
int color;
}

main() {
fruit * fruit_ptr;
fruit_ptr = new fruit[100];
}
```

Answer:
We need to write a default constructor for fruit (fruit(int w = 1, int c = 2)), and we want to initialize weight to w = 1 and color to c = 2 by default.

When new fruit[100] is called, the default constructor is called for each fruit in the dynamic array, thus will initialize all weight to 1 and color to 2

5. Explain why *heap* variables are essentially global in scope. Please present an example as well.

Variables created on the Heap can be accessed by any function, in anywhere of the program, thus are essentially global in scope.

Example: 1. int *ptr;
2. ptr = new int(5);
this new keyword allocates memory for the int pointer ptr on the heap. Thus making it a dynamic pointer of type int of initial value of 5.

6. Is it possible to use the keyword "`this`" inside a friend function? Please explain your answer.

Answer:

My guess is that you can not use "this" inside a friend function. Since friend function is not a member function, this program will not recognize the default class of "this" pointer, therefore can not determine the data type of the function.

So if you are trying to do "this.some_private_variable" or "this.some_function()", the program does not know which object/class you are talking about.

7. (4 pts) Does the following code compile? Does it run? Is there any problem with the code? If yes, how do you fix it?

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

class Computer {
int Id;

public:
Computer(int id) { this -> Id = id; }
void process() { cout << "Computer::process()"; }
};

class Employee {
Computer* c;

public:
Employee() { c = new Computer(123); }
~Employee() {}
void foo() {
cout << "Employee::foo()";
c -> process();
}
};

int main() {
Employee ob;
ob.foo();
return 0;
}
```

There is nothing wrong with the code, at least no compilation error, as I tested on my computer.