OOP345 [redacted] Midterm Exam – Sample

Part A: Short Answer (15%)

1. Explain the difference between stack memory and heap memory in the context of C++. (2%)
2. Describe the concept of "composition" in object-oriented programming, and give an example in C++. (2%)
3. What is a pointer in C++? How does it differ from a reference? Provide a one-line code example for each. (3%)
4. What does the keyword "extern" signify? (2%)
5. In C++, explain the difference between **public**, **private**, and **protected** access specifiers. Provide a situation where you might use each. (3%)
6. Describe the significance of **virtual** functions in C++ with respect to inheritance and polymorphism. (3%)

Part B: Debugging (25%)

Below is a piece of code with several errors:

```
#include <iostream>

class MyClass {
private:
   int* data;
   static int count;
public:
   MyClass() : data(new int[5]) { count++; }
   ~MyClass() { delete data; }
   void setData(int index, int value) { data[index] = value; }
   int getData(int index) const { return data[index]; }
   static int getCount() { return count; }
};

int MyClass::count = 0;

int main() {
   MyClass obj1, obj2;
   obj1.setData(5, 10);
   std::cout << "Count: " << MyClass::getCount() << std::endl;
   return 0;
}
```

Identify and explain the errors, and suggest fixes.

Part C: Walkthrough (20%) [Note: Some code snippets will be MCQs and some won't be]

Given the following code snippet, determine the output:

```
#include <iostream>

class Base {
public:
   virtual void show() { std::cout << "Base\n"; }
};

class Derived : public Base {
public:
   void show() override { std::cout << "Derived\n"; }
};

int main() {
   Base* b = new Derived();
   b->show();
   delete b;
   return 0;
}
```

Part D: Programming (40%)

You are tasked with implementing a simple **Rectangle** class. The requirements are as follows:

1. The class should have two private member variables for width and height.
2. Implement appropriate constructors to initialize the dimensions.
3. Implement the copy and move constructors.
4. Implement a member function **area()** that returns the area of the rectangle.
5. Create a member function **scale(double factor)** which scales the rectangle's dimensions by the given factor.
6. Implement a static member function **totalRectangles()** which returns the total number of rectangle objects created.

Write the C++ code for the above requirements.

Answers:

Part A:

1. Stack memory is used for static memory allocation and heap memory for dynamic memory allocation. Variables allocated on the stack are stored directly to memory and access to this memory is very fast. Variables allocated on the heap have their memory allocated at runtime and accessing this memory is a bit slower. Stack has size limits, heap does not.
2. Composition is a "has-a" relationship between classes, where one class contains an instance of another class. E.g., A **Car** has an **Engine**.
3. A pointer holds the memory address of a value. A reference is an alias for a variable. Example: **int* ptr;** and **int& ref = someVar;**.
4. The **mutable** keyword is used to specify that a member of an object can be modified even if the object is declared as **const**.
5. **public** members are accessible from anywhere. **private** members are only accessible within the class. **protected** members are accessible within the class and its derived classes. Use **public** for interfaces, **private** for encapsulation, and **protected** for inheritance.
6. **virtual** functions are used in base classes to ensure that the derived class's function is called, even when a base class pointer/reference is used. They are fundamental for runtime polymorphism.

Part B:

1. **delete data;** should be **delete[] data;** as data is an array.
2. In **obj1.setData(5, 10);**, the index 5 is out of bounds as the array size is 5.
3. The **count** variable should be incremented in the copy constructor as well, if the intention is to count all objects.

Part C:

Output:

Derived

Part D:

```
class Rectangle {
private:
   double width, height;
   static int total;

public:
   Rectangle(double w, double h) : width(w), height(h) { total++; }
   Rectangle(const Rectangle& r) : width(r.width), height(r.height) { total++; }
   Rectangle(Rectangle&& r) : width(r.width), height(r.height) { r.width = 0; r.height = 0; total++; }
   double area() const { return width * height; }
```

```
    void scale(double factor) { width *= factor; height *= factor; }
    static int totalRectangles() { return total; }
};

int Rectangle::total = 0;
```