Midterm 1 Practice

CSci 3081

Spring 2016

Here are some practice questions for the upcoming midterm. The actual exam will contain more questions. However, these questions are representative of some types of questions and topics that *might* appear on the actual exam.

Do as many problems below as time permits (there might not be time for all the problems, so feel free to be selective). This is practice and we will not ask you to hand anything it, but treat the practice like an actual exam, for example, show your work on nontrivial problems. Also, when writing code, you do not need to include comments, but otherwise use good style, including (i) not writing complicated code when there is a less complicated solution, and (ii) making sure your code is easy to understand.

We will not be posted solutions to these questions. However, the last part of lab will be for the TA(s) to answer questions and discuss solutions.

Problem (1) Suppose you have the following C++ code:

```
int x = 1, y = 10, z = 100;
int *px = &x, *py = &y, *pz = &z;
```

And suppose the function `f` is as follows:

```
int f(int &a, int b, int *c) {
   a = a * 2;
   b = b * 3;
   *c = (*c) * 4;
   return a + b + *c;
}
```

(a) What value is returned by the function call `f(x, y, pz)`?

(b) What are the values of `x`, `y`, and `z` after completion of the function call in part (a)?

Problem (2) This questions involves the Project Iteration 1 solution code.

(a) Look up the code for the classes `Mask`, `TPen`, `Tool`, and `ToolFactory` (in an actual exam, this code would be given to you as part of the exam handout). Then draw a UML

diagram showing those four classes and the relationships between them. You do not need to include class attributes and operations, or any multiplicities in the relationships. But make sure each type of relationship is clear — you may include roles or short comments on the relationships if useful.

(b) Draw a UML diagram for the `Tool` class. Use the same level of detail as you did for Writing Assignment 3.

Problem (3) One of McConnell's design guidelines was to use a consistent level of abstraction in class interfaces. Create an example that violates this guideline by creating a class having at least two different member functions at different levels of abstraction. You do not need to give code for this class, but (i) give a class name and the function names, (ii) give a brief description of what the class and operations do, and (iii) give a brief description of how the class interface violates the design guideline.

Problem (4) Recall the `Duck` example from Lab 2. (You may look up the code online if you wish; in an actual exam, any needed code would be included in the exam handout.) Suppose you wanted to give each Duck object its own color. This includes objects of the Duck subtypes such as `MallardDuck` objects, `DecoyDuck` objects, etc. Assume there should be an initial color for each class (e.g., yellow for the `DecoyDuck`). However, a user should have the ability to change the color for each object after the object is created; for example, a `DecoyDuck`'s yellow might fade as the duck ages.

You decide to have a member variable of type `ColorData` (from the project support code) to store the color information. Where would you put this `ColorData` member variable? For example, would it be part of the `Duck` base class? Part of each subclass but not part of the `Duck` base class? Somewhere else? Explain clearly where you would place it, and briefly justify your decision.

Problem (5) Consider a `NumberRange` class that has the following:

- It has two member variables: `m_range` is a pointer to an array of ints, and `m_size` the number of elements in that array.
- It has a single constructor that takes two integer arguments `a` and `b`, with `a` guaranteed to be greater than or equal to `b`. The constructor sets `m_size` to the number of integers between (and including) `a` and `b`. It then allocates an array of integers to store the values `a, a+1, a+2, ..., b`, with `m_range` pointing to the array. For example, if `a` is 5 and `b` is 9, then the constructor would allocate an array and fill it with the values 5, 6, 7, 8, 9 in that order.
- It has a destructor that does any needed clean-up.
- It has a member function `print()` that prints out the values in the array, in order, one per line.

Write the C++ code for both the `.h` and `.cpp` files for this class.