Terence Henriod

Test 1 Challenges

CS 302: Data Structures

October 7, 2013

|  |  |
| --- | --- |
| The Question: | Define the **scope** of a variable. |
|  |  |
| My Assertion: | I assert that my answer wrongly had a point deducted. My original answer was: “The scope of a variable is the frame of reference through which a variable may be referenced. This frame of reference may be limited to the time in which the instantiating function is being called.1, 2 (Except in cases like static variables)” (Parts that were crossed out on my test are underlined) |
|  |  |
| Reasoning: | I believe that I had credit reduced simply because I used phrasing like “frame of reference” and not “context of a function wherein a variable was instantiated.” I was hesitant to make such a statement because variables do exist in the context of a function, but many exist in the context of a class, as well.1, 3 Static variables, for example, can exist in the scope of a class in general, and not necessarily the lifetime of an object or the lifetime of a function.3 Regular member variables of any instance of a class are referenced in the scope of that object, and not in the scope of a function.2  Perhaps I should have made it clear that the “frame of reference” was the context of a function or class, but I was concerned that if I spent too much time defining the broadest sense of the definition of scope that I would receive a reduction in credit. However I felt it was intellectually dishonest to restrict the definition of the scope of a variable to just that of a function.  I believe that all 3 of the following quotes from the cited textbook are relevant to this matter, and support my original answer. I have annotated the components of my answer and my additional reasoning with numbers that correspond to these quotes:   1. “The scope of a variable is the part of the program where the variable may be used… The first rule of scope you should learn is that a variable cannot be used in any part of the program before the definition.” 2. “**NOTE:** When a program is running and it enters the section of code that constitutes a variable’s scope, it is said that the variable *comes into scope*. This simply means the variable is now visible and the program may reference it. Likewise, when a variable *leaves scope*, it may no longer be used.” 3. “When a value is stored in a static member variable, it is not stored in an instance of the class… You can think of static member variables and static member functions as belonging to the class instead of to an instance of the class.”   See: Starting Out With C++: From Control Structures through Objects, by Gaddis; pages 59, 213, and 800, respectively. (The text for this class does not seem to have a suitable discussion about scope) |

|  |  |
| --- | --- |
| The Question: | Write the function to insert after the cursor. |
|  |  |
| My Assertion: | (This one is a bit silly, but I thought I’d give it a shot)  In answering this question I had to write isFull(). I mistakenly wrote that the function should always return true, as opposed to false. I believe that I should get credit for this, even though it is not correct, on a technicality. |
|  |  |
| Reasoning: | In the lab manual (C++ Data Structures: A Laboratory Course, by Brandle, Geisler, Robergé, and Whittington), on page 63 it states: “… we recommend that your implementation of the isFull function assume that there is always more memory available and always return true.” I think this provides a technicality that should allow me to receive credit anyway for this answer because: (1) If professionals get paid to write a textbook with this mistake, should a student be held to a higher standard? and (2) There is a set of directions saying to implement the function this way (even if it is incorrect) that may have been studied in preparation for the test.. |

|  |  |
| --- | --- |
| The Question: |  |
|  |  |
| My Assertion: |  |
|  |  |
| Reasoning: |  |