CS-2303 System Programming Concepts WPI, A-term 2017  
Professor Mike Ciaraldi Quiz #2 (20 points)  
Quiz date: Thursday, September 7, 2017 ANSWER KEY

|  |  |  |
| --- | --- | --- |
| Question | Possible | Points |
| 1 | 5 |  |
| 2 | 5 |  |
| 3 | 5 |  |
| 4 | 5 |  |
| Maximum | 20 |  |

NAME:

WPI E-mail ID:

READ THESE INSTRUCTIONS.

This is an closed-book quiz, open-notes quiz. You can consult any of the class handouts, and anything you wrote yourself. You cannot consult tests from previous versions of the course. You are not allowed to use any electronic or communications device during the quiz, without special permission.

This quiz is worth 20 points.

Answer questions in the spaces provided on the quiz itself. Take the number of points assigned to each question and the amount of space provided for your answer as a measure of the length and difficulty of the expected solution.

Be sure to answer the question which is actually being asked, in a way which demonstrates that you understand the meaning of the question and the answer. For example, if the question asks you to say what happens and why, you have to tell both to get full credit. *Remember, the graders know the answers to these questions. What you write has to show that you understand the question and its answer.*

A *program fragment* is a section of code which is part of a complete program. You can make any reasonable assumptions about the rest of the program.

All questions apply to C unless otherwise indicated.

This quiz will end at 8:20 am. *It is* STRONGLY *suggested that you read the entire quiz before attempting to answer any questions. Also, re-read all the parts of any one question before you start answering it.*

1. **Function prototypes**. When the compiler sees that you are calling a function, it needs to know certain information about that function in order to make sure you are calling it correctly and to properly generate the machine code. So it typically looks at the function prototype for the function with that name. What are two pieces of information contained in a function prototype which the compiler needs in this case? [5 points]

*Answer: The number and type of the parameters. The return type of the function. I did not accept “the name of the function” as an answer, because that is in the question itself. That would be like asking, “If I know someone’s name, what information can I find out about him or her?” and then accepting “Well, you can find out the name.” Note: From the prototype, the compiler can tell what type each parameter is supposed to be, but it cannot tell their significance. For example, if there are two int parameters, the prototype does not their meanings, so it cannot check for that.*

1. **Arrays**. [5 points]
   1. When you pass an array as a parameter into a function, what actually gets copied to the function? [2 points]

*Answer: The address of the start of the array. This is also called the base of the array. It is the same as the address of element 0 of the array. So any of these would be considered a correct answer.*

* 1. When you pass an array to a function as a parameter, why do you generally have to also pass the size of the array as a separate parameter? Note: This is not asking why the function cares about the size; it is asking why you need a separate parameter. [2 points]

*Answer: Because there is no way for the function to find out the size, since all the function knows is the address of the start of the array. Note: Writing “because the function would not know otherwise” is not specific enough; that answer could be guessed just from the way the question was worded.*

* 1. Why do you not need to pass a separate parameter in Java? [1 point]

*Answer: Because in Java an array is an object. If the function can access an array object, it can access both its contents and its size.*

1. **Lifetime**. [5 points]
   1. What is the lifetime of a local variable of a function? In other words, when is memory allocated for it, and when does it go away? [3 points]

*Answer: Local variables are created when you enter a function or block, and destroyed when you exit that function or block. Note: The memory gets allocated at run time, not at compile time. So, writing :when the function is declared” would not be a correct answer.*

* 1. Where are local variables stored in memory? Hint: This is important for recursive functions. [2 point]

*Answer: On the stack. That is all you had to write, but here is why the stack is appropriate: Each time a recursive function is called, you get a fresh copy of the local variables – recursion would not work otherwise!*

1. **Scope**. [5 points]
   1. If you declare a variable local to a function, where is it visible? [2 points]

*Answer: Inside that function and any blocks within that function. I accepted “inside the function” because the blocks are indeed inside the function, but it is more correct to mention both, so I will expect that if I ask again.*

* 1. What happens if you declare a global variable, and also declare a local variable with the same name? [1 point]

*Answer: Within the function, the local variable “masks” the global one, so only the local is visible.*

* 1. If a variable is declared outside of any function, but is marked with the keyword “static”, where is it visible? [2 point]

*Answer: Only to functions in the same C source file. There was some question about whether this static variable is visible to the linker. The name of the variable is not visible to the linker. However, the linker does get informed how much static memory is needed by each .o file.*