CS-2303 System Programming Concepts WPI, A-term 2017  
Professor Mike Ciaraldi Quiz #1 (20 points)  
Quiz date: Thursday, August 31, 2017 ANSWER KEY

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

NAME:

WPI E-mail ID:

This is a closed-book, open-notes quiz. You can use the class slides and other handouts (on paper) and anything you have written yourself. Note: The phrase “written yourself” means that you composed the text yourself, not whether you physically typed or hand-wrote it. **Copies of previous tests are not permitted, even if you hand-copied them.**

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.

In other words: The answer you write not only has to be a true statement, it has to actually answer the question. It cannot merely restate the question in the form of a statement. For example, if the question asks you to explain how a **for** loop works, it is not sufficient to say, “It loops by repeating the body of the loop.”

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.

This quiz will end at 8:20 am. *It is* STRONGLY *suggested that you read the entire quiz before attempting to answer any questions.*

1. Doxygen (5 points)

You use the Doxygen system by inserting specially-formatted block comments into your source code.

* 1. How does the **compiler** know to skip over these blocks of text, as distinguished from any other text in the file? (3 points)

*Answer: Each Doxygen comment starts with the character sequence “****/\*\*****”. The compiler treats any block of text beginning with “****/\*****” as a comment; the extra star does not matter to the compiler. Note: Just saying that the block starts with “****/\*\*****” is not a sufficient answer, because the compiler only looks at the first two characters before deciding.*

* 1. How does the **Doygen utility** recognize the comments it is supposed to process, as distinguished from all the other comments? (2 points)

*Answer: The Doxygen utility only processes those comments which start with* *the character sequence “****/\*\*****”. It ignores any comments which begin with “****/\*****” but do not have the second star.*

1. Formatted input (5 points)

Suppose you run the following program fragment:

**int j = 1, k = 5; double a = -7.6;**

**j = scanf(“%d,%f”, &a, &k);**

And you type the following input:

**35.6, 127**

* 1. What will be the values of j, k, and a after calling scanf()? (2 points)

*Answer:* ***j == 2, a == 35.6, k == 127***

*This is what I was looking for. In fact, there is a mistake in the question, which the compiler would have caught. The format string should have been “%f,%d” because the double came before the int. The return value of scanf() is the number of successful conversions.* ***I***

* 1. Why did we have to put the ampersand (‘&’) symbol before k and a, in order for the function to work properly? Be sure your answer tells what the ampersand does, and why this is necessary. (3 points)

*Answer: This causes the function to use pointers to k and a (in other words, to use their addresses in memory, rather than their current values). This is so the function knows where to store the values it has read from the input.*

1. For and Switch (5 points)

Demonstrate your understanding of the **for** and **switch** statements by telling what will be printed by the following program. Be sure to indicate line breaks.

**#include <stdio.h>**

**int main() {**

**int i; // Loop counter**

**for (i = 1; i <= 6; i++) {**

**switch (i) {**

**case 0:**

**printf("A\n");**

**break;**

**case 1:**

**printf("B\n");**

**break;**

**case 2:**

**printf("C\n");**

**break;**

**case 3:**

**case 4:**

**printf("D\n");**

**case 5:**

**printf("E\n");**

**break;**

**default:**

**printf("F\n");**

**break;**

**} // end of switch**

**} // end of for loop**

**return 0; // Indicate successful program exit.**

**}**

*Explanation: The loop counter i goes from 1 to 6, inclusive. So, case 0 never gets used. Case 1 prints “B” and then breaks to the end of the switch statement. Case 2 prints “C” and then breaks to the end of the switch statement. Case 3 does not do anything unique, but because there is no break, the program falls through to case 4. Case 4 prints “D” and falls through to case 5. Case 5 prints “E” and breaks to the end of the switch statement. For any other value of i, the program prints “F” and then breaks to the end of the switch statement.*

*Answer:*

***B***

***C***

***D***

***E***

***D***

***E***

***E***

***F***

1. Formatted output (5 points)

Suppose you have some variables declared like this:

**int i = 12;**

**int j = 34;**

**double a = 123.4752;**

**double b = 96.431;**

* 1. Write a call to the function printf() which will print the values of i and j so the output looks like the following. Be sure the output skips to a new line after printing. (1 point)

**1234**

*Answer:* ***printf(“%d%d\n”, i, j);***

*Note: You do not use pointers here, because printf() needs the value in the variables, not their addresses.*

* 1. Write a call to the function printf() which will print the values of i and j so the output looks like the following (one blank between the numbers). Be sure the output does not skip to a new line after printing. (1 point)

**34 12**

*Answer:* ***printf(“%d %d”, j, i);*** OR ***printf(“%d%3d”, j, i);***

* 1. Write a call to the function printf() which produces the same output as in part b, except that there will always be exactly one space between the numbers no matter how many digits each one has. For example, even if i == 12345. (2 points)

*Answer:* ***printf(“%d %d”, j, i);***

* 1. Write a call to the function printf() which prints the value of a so the output looks like the following. Be sure the output skips to a new line after printing. (1 point)

**123.48**

*Answer:* ***printf(“%6.2f\n”);*** *Note: The most important part is the “.2”, because it specifies that exactly two decimal places will be output, rounding if necessary. The “6” specifies the minimum number of characters to output, including the decimal point and all the digits on both sides of the decimal point. It could be left out or could be any other number. If it is left out, or <= 6, then a == 123.4752 will cause 6 characters to be printed, because printf() will not skip digits to the left of the decimal point. If it is > 6, enough blanks will be output before the digits to produce the specified minimum length.*