Simulated Quiz 0

out of 70 points

CS50 — Fall 2012

Prepared by: Doug Lloyd '09

October 8, 2012

This practice Quiz is designed as a simulation of what you can expect to see on real Quiz, based on the types and number of questions that have appeared on the Quiz in the past. This simulated Quiz is not necessarily a reflection of exactly the material that you will see on the real Quiz this year, nor do the relative weights of the questions necessarily reflect how questions will be weighted this year. This simulated Quiz will have absolutely no impact upon your grade and is meant only as a study aid.

Do not turn this page over until you are ready to begin.

This quiz is "closed-book." However, you may utilize during the quiz one two-sided page (8.5" x 11") of notes, typed or handwritten, and a pen or pencil, nothing else.

Unless otherwise noted, assume that any problems herein refer to C.

You have 75 minutes to work on this simulated Quiz.

Multiple Choice.

For each of the following questions or statements, circle the letter (a, b, c, or d) of the one response that best answers the question or completes the statement; you need not explain your answers.

0. (0 points.) Which House is the best House?

   (a) Eliot.

   (b) ~~Lowell.~~ Eliot.

   (c) ~~Currier.~~ Eliot.

   (d) ~~Mather.~~ Eliot.

1. (1 point.) What line of code would print the third letter of the second command line argument, assuming it existed?

   (a) `printf("%c", argv[3][2]);`

   (b) `printf("%c", argv[2][3]);`

   (c) `printf("%c", argv[1][2]);`

   (d) `printf("%c", argv[2][1]);`

2. (1 point.) How many bytes in an `int`?

   (a) 1

   (b) 4

   (c) 8

   (d) 32

3. (1 point.) Assuming `x` is set to 2 in the line directly above this code, how many asterisks will print?

```
while(x <= 10)
{
   printf("*");
   x += 2;
}
```

   (a) 4

   (b) 5

   (c) 6

   (d) 7

4. (1 point.) How many bytes are required to store the string "`This is CS50!`" (not including the surrounding quotation marks) in memory?

   (a) 11

   (b) 12

   (c) 13

   (d) 14

True or False.

For each of the statements below, circle T if the statement is true and F if the statement is false.

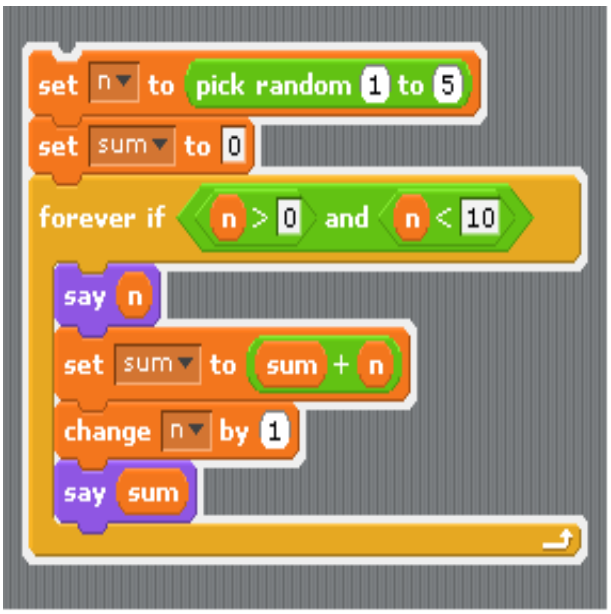
5. T F (1 point.) It is possible for the best case runtime of bubble sort to be $\Omega(n)$.
6. T T (0 points.) You kind of felt like folk dancing during that bubble sort demonstration video.
7. T F (1 point.) '`\0`' is a special character that is used to demarcate the end of every array.
8. T F (1 point.) `void` functions can only return variables of type `void`.
9. T F (1 point.) The ASCII value of '`A`', in binary, is 01000001.

Back to Basics?

10. (2 points.) Perform the calculation below in binary. Then, convert your answer from binary to decimal.

```
  0 1 0 1 1 0 1 0
+ 0 0 1 1 0 0 1 1
-----------------
```

11. (5 points.) Consider the below set of Scratch blocks:



This set of blocks basically chooses a random number in the range [1, 5], and calculates the sum of the numbers from the chosen number up to 10, printing out a couple of things along the way. Translate these Scratch blocks to a C function called `sum()`, whose declaration appears on the next page. No need to do any `#includes` or print out newlines (unless you want to). Know that C has a function `rand()` which you can use to generate a random number, though `rand()`s return value could be anywhere in the range [0, `INT_MAX`], so you'll need to do some work to put it in the proper range.

```
void sum(void)
{









}
```

Point(er) Me in the Right Direction.

12. (2 points.) Explain, in no more than two sentences, why some people (hopefully you among them!) might find the following image funny:

13. (7 points.) Fill in the following table regarding the values of variables and pointers. You may assume that the following declarations are made, and that `a` is stored at memory address 0x4 (which is decimal 4), `b` is stored at memory address 0x8 (decimal 8), and `c` is stored at memory address 0xC (decimal 12). You may assume that each row, everything begins anew. That is, you don't need to carry the values of the variables from row to row. You'll notice that we've plucked off a little more than half of the cells for you.[1]

```
int a = 3, b = 5, c = 6;
int *pa = &a, *pb = &b, *pc = &c;
```

Code	a	b	c	pa	pb	pc
`a /= c;`		5		0x4	0x8	0xC
`*pc = *pa * a;`	3	5		0x4	0x8	0xC
`pb = *pb;`	3		6	0x4		0xC
`*pc *= *pb;`	3			0x4		
`pc = &*pa;`		5			0x8	

Big O. Oh Boy.

14. (3 points.) In the space below, write a function consistent with the below declaration and that runs in $O(1)$.

```
int yourFunction(int array[], int arraySize)
{




}
```

15. (6 points.) Complete the following table regarding asymptotic notation. As indicated, fill in worst-case runtime (O) and best-case runtime (Ω). If you need to state an assumption to complete your answer, do so. Two of the eight cells have been plucked off for you.

Algorithm	O	Ω	Assumptions or Prerequisites, if any
Binary Search			List is already sorted
`strlen()`, where the string is of length n			
Merge Sort	$O(n \log n)$		
A recursive factorial function with the call `fact(n)`		$\Omega(1)$	

[1] "Yeah," you're thinking. "Some help you are."

A Buggy Life.

```
1  void main(void) {
2     int m = GetInt();
3
4     if(m) {
5        for(int i = 0; i <= m; i++) {
6           if(i = 13)
7              break;
8           if(i % 2) {
9              char *s = "i is odd here\n";
10             printf("%d: %s", i, s);
11          }
12          else {
13             s = "i is even here\n";
14             printf("%d: %s", i, s);
15          }
16       }
17       printf("I was only able to count to %d!\n", i);
18    }
19    return 0;
20 }
```

16. (4 points.) The above code (whose lines have been numbered for the sake of discussion) will not compile. When `make` is run, on which lines does the compiler say there's a problem, and why? (You may assume we have properly `#include`'d cs50.h and stdio.h.)

17. (2 points.) Suppose Tommy's Caesar cipher program works perfectly most of the time, but when he types the following at the command line:

    ```
    > ./caesar
    ```

    the response he gets back at the terminal is:

    ```
    Segmentation fault (core dumped).
    ```

    What is the most likely explanation for why this happened?

18. (2 points.) The code below is supposed to only spit out even numbers. If it gets an even number as its input, it returns it, and otherwise, it returns double the number (since 2 times any odd number is an even number). But for some reason, when we run this through GDB, the function is returning odd and even numbers! Tell us why, and propose a solution to fix it.

```
int toEven(int x)
{
   // Even numbers, divided by two, have no remainder
   if(x % 2 == 0);
      return x;

   // If the above isn't satisfied, we know we have an odd number
   return (2 * x);
}
```

Tell Me About It!

19. (2 points.) Describe the difference between a `while` loop and a `do-while` loop. Give an example of an instance where a `while` loop would be more useful and another where a `do-while` loop would be more useful and explain why.

20. (2 points.) What is a stack frame? How are they created, and how are they destroyed?

21. (2 points.) Sort the following "classes" of functions in the order of generally fastest to generally slowest: *logarithmic, exponential, factorial, constant, polynomial, linear.*

22. (2 points.) Imagine we execute the following lines of code:

```
string input = GetString();
string input_copy = input;
input_copy[0] = 'X';
```

Why is it that, if we look at `input[0]`, it is also 'X', even though our line of code modified `input_copy`?

23. (2 points.) Give at least two reasons why `#define` statements are useful.

Just a Number in a File.

24. (3 points.) In the space below, declare a "container" of a type called `student` containing the following information:

- A string (**without** using CS50's `string` type) for a student's last name,
- An integer to represent that student's ID number,
- A floating point number to represent their GPA, and,
- A character (A, B, C, D) to represent their class year (freshman, sophomore, junior, senior).

Curses! It Recurses!

25. (3 points.) What exactly does the below function do?

```
void mystery(char c)
{
   if(c < 'A' || c > 'Z')
      return;
   else {
      mystery(c - 1);
      printf("%c", c);
      return;
   }
}
```

26. (2 points.) Rewrite the above function iteratively, instead of recursively, so that the behavior is identical.

```
void mystery_iterative(char c)
{









}
```

A Few Hours From Now.

27. (2 points.) Nate has written a program that takes a student's quiz score and assigns a letter grade to it. The code in the `giveGrade` function looks like this:

```
char giveGrade(int quizScore)
{
   int newScore = (quizScore/10) * 10; // Question 28
   char letterGrade;

   switch(newScore) {
      case 100: case 90:
         letterGrade = 'A';
      case 80:
         letterGrade = 'B';
      case 70:
         letterGrade = 'C';
      case 60:
         letterGrade = 'D';
         break;
      default:
         letterGrade = 'E';
   }

   return letterGrade;
}
```

Explain in no more than two sentences why everyone in the class is really upset about their grade. Propose, in no more than two sentences, a fix.

28. (1 point.) Why is the line marked `Question 28` in the code above not equivalent to `quizScore`? After all, as a general rule, $\frac{n}{10} * 10 = n$.

Hanging by a String.

29. (8 points.) Recall that `strcmp` is a function that, according to its man page, "compares the two strings `s1` and `s2`. It returns an integer less than, equal to, or greater than zero if `s1` is found, respectively, to be less than, to match, or be greater than `s2`." Unfortunately, it looks like we've misplaced `string.h`, where `strcmp` is usually declared, and so you must rewrite this function yourself. Complete the implementation of `strcmp` below. Suffice it to say, your solution may NOT call the built-in version of `strcmp`. However, we've begun work on rebuilding `string.h` and the function `strlen` is at your disposal, should you wish to use it. If either `s1` or `s2` happens to be `NULL`, your implementation must return a nonzero value (consider `NULL` to be the first possible string, alphabetically). If **BOTH** `s1` and `s2` are `NULL`, your function may return zero. Note also that we've declared `s1` and `s2` as `const char *`s, which means they may not be modified in any way.

```
int strcmp(const char *s1, const char *s2)
{









}
```


dere, all done
i thought u said dis was hard?
ICANHASCHEEZBURGER.COM