Simulated Quiz 0 - Answer Key

CS50 — Fall 2012

Prepared by: Doug Lloyd '09

October 8, 2012

Many of these answers are sample answers. Thus, more answers are possible.

Multiple Choice. Score 1 point if right, 0 points otherwise.

0. a, b, c, or d
1. c
2. b
3. b
4. d

True or False. Score 1 point if right, 0 points otherwise.

5. T
6. T
7. F
8. F
9. T

Back to Basics?

10. Score 1 point for correct sum, 1 point for correct decimal conversion.

```
  1 1 1     1
  0 1 0 1 1 0 1 0
+ 0 0 1 1 0 0 1 1
-----------------
  1 0 0 0 1 1 0 1 (141)
```

11. Score 0 points for unattempted, 3 points for broken, 5 points for perfect.

```
void sum(void)
{
   int n = (rand % 5) + 1;
   int sum = 0;
   while((n > 0) && (n < 10))
   {
      printf("%d\n", n);
      sum += n;
      n++;
      printf("%d\n", sum);
   }
   return;
}
```

Point(er) Me in the Right Direction.

12. Pointers are nothing more than addresses, and give us direct access to the memory located at those addresses. The "answers" being given by the character on the right are also addresses, so he is providing pointers in the computer science sense, not the typical English sense. Score 2 points for an answer in this vein, 0 otherwise.

13. Score 1 point for the first correct cell, and 1 point more for every two more correct:

Code	a	b	c	pa	pb	pc
`a /= c;`	0	5	6	0x4	0x8	0xC
`*pc = *pa * a;`	3	5	9	0x4	0x8	0xC
`pb = *pb;`	3	5	6	0x4	0x5	0xC
`*pc *= *pb;`	3	5	30	0x4	0x8	0xC
`pc = &*pa;`	3	5	6	0x4	0x8	0x4

Big O. Oh Boy.

14. Score 0 points for unattempted, 1 point for function not in $O(1)$ (any operation on `array`), 3 points for perfect.

```
int yourFunction(int array[], int arraySize)
{
   return arraySize;
}
```

15. Score 1 point for each cell in the O and Ω columns correctly filled in.

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

A Buggy Life.

16. Score 1 point each:

    - Line 6 – assignment when intention is to use equality operator, compiler will warn to place parentheses around assignment used as truth value
    - Line 13 – use of undeclared identifier `s`, which is out of scope (only exists lines 9 and 10)
    - Line 17 – use of undeclared identifier `i`, which is out of scope (only exists lines 5 through 16)
    - Line 19 – returning a value from a `void`-type function

17. Tommy most likely doesn't check that `argc` is equal to 2 before he takes `atoi(argv[1])`, since his program works perfectly as long as there are 2 command line arguments. Score 2 points for this explanation, 1 point for any other explanation that would cause a segfault, 0 points otherwise.

18. The semicolon right after the `if` statement renders it useless, and so despite the indentation, the next line is executed for ALL inputs, and the last line is never executed. This, among other solutions, works:

    ```
    int toEven(int x)
    {
       return (x % 2) ? (2 * x) : x;
    }
    ```

    Score 1 point for explaining why it's broken, and another 1 point for any working solution.

Tell Me About It!

19. A do-while loop guarantees that the body of the loop is run at least once. A while loop offers no such guarantee. A do-while loop is often more useful for user input, because you want to grab input at least once. A while loop is cleaner in most other cases. For instance, if you wanted to implement a "forever" loop, you'd generally do:

    ```
    while(1) { }
    ```

    instead of a do-while. This is simply for clarity while reading. Score 1 point for explaining the difference, and 1 point for describing any valid use of each.

20. A stack frame is a chunk of memory which is given to every new instance of a function by the processor when that function is called. It is the memory which that function uses to do its work. Once a function returns a value or completes execution, the stack frame is destroyed, meaning the memory that function once was using is released back to the system. Score 2 points for touching on all of the three questions asked, 1 point for touching on two of them, 0 points otherwise.

21. constant, logarithmic, linear, polynomial, exponential, factorial. Score 2 points for perfect order, 1 point for one or two switched, 0 points otherwise.

22. Recall that string is just CS50's alias for a `char *`. A `char *` of course is a pointer to a character (or, in this case, an array of characters). A pointer is just an address. So when we assign one pointer's value to another pointer's value, both pointers have the same address. Ergo, they are pointing to the exact same thing. So, any change made to `input` will also show up as a change to `input_copy`, because they are pointing to the same location in memory. To achieve the effect likely desired here, one would have to use `strcpy()` or its more secure cousin `strncpy()`, to obtain two different pointers to two different addresses (and therefore two different strings). Score 2 points for an explanation that touches on most of these points, 1 point for half-right, 0 points otherwise.

23. Score 1 point for each valid explanation. For example: They improve readability of our code, and they also eliminate magic numbers.

Just a Number in a File.

24. Score 1 point for using the `struct` keyword, and 1 additional point for each set of two fields properly declared:

```
struct student {
   char *lastname;
   int idnum;
   float gpa;
   char classyear;
}
```

Curses! It Recurses!

25. Score 3 points for properly analyzing that this function prints the alphabet in order from 'A' to c, the parameter passed in. If that parameter is not a capital letter, the function prints nothing. Score 0 points otherwise.

26.
```
void mystery_iterative(char c)
{
   if(c < 'A' || c > 'Z')
      return;
   for(char i = 'A'; i <= c; i++)
      printf("%c", c);

   return;
}
```

    Score 1 point for keeping the error checking, and 1 point for a loop that does indeed print the alphabet in order.

A Few Hours From Now.

27. There are no break statements, meaning that our cases will "fall through" until they hit one! So, everyone who scored 60 or better will get a D, and everyone else will get an F. Simply inserting break statements at the end of each case would fix this problem. Score 1 point for explaining why it's broken, and 1 point for proposing any fix that would work.

28. This is what happens with C's integer arithmetic. The trailing decimal portion will be truncated. So,for example, 89/10, which would otherwise be 8.9, gets truncated to just 8. Multiplying this by 10 puts us up to 80. It helps here though, as it allows us to use a switch, instead of an if-else combination, as would otherwise be necessary. Score 1 point for a correct explanation, 0 points otherwise.

Hanging by a String.

29.
```
int strcmp(const char *s1, const char *s2)
{
   int pos = 0;

   while(s1[pos] == s2[pos])
   {
      if(s1[pos] == '\0')
         return 0;
      pos++
   }

   // if we're here, we have unequal characters, so return their difference:
   return (s1[pos] - s2[pos]);
}
```

Score 2 points for returning the correct values at the end (s1 is lesser: -1, the strings are equal: 0, s1 is greater: 1), and a maximum of 6 points for proper internal mechanics of the function, with discretionary deductions coming from that 6 for design inefficiencies.