Simulated Quiz 1 - Answer Key

CS50 — Fall 2012

Prepared by: Doug Lloyd '09

November 10, 2012

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

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

0. b
1. c
2. d
3. a

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

4. F
5. F
6. T
7. T or F

Magic Numbers

8. Score 1 point for every two correct cells.

Binary	Octal	Decimal	Hexadecimal
10101	25	21	15
101000	50	40	28
110010	62	50	32
1010000	120	80	50

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

```
function sum()
{
   $n = (rand() % 5) + 1;
   $sum = 0;
   while($n < 10)
   {
      print($n . "\n");
      $sum += $n;
      $n++;
      print($sum . "\n");
   }
}
```

HTML-o World

10. `http://www.davidmalan.com/result?input=CS50`. Score 2 points for an answer in this vein, 0 otherwise.

A Buggy Life 2.

11. `fp` has only been opened for reading. Because the file handle is "read-only", any attempt to write to it will fail. The fix is to either open `fp` for reading and writing (`r+`) or to open a second file pointer to handle writing. 1 point for explaining why it's broken, 1 point for proposing the fix.

12. Because the function prints BOTH children before the parent node, everything to the left (less than) is printed, then everything to the right (greater than) is printed, and then finally the parent itself is printed. The fix is to print the parent node before the right child. 1 point for explaining why it's broken, 1 point for proposing the fix.

13. A still-reachable 352 bytes in 1 block is a classic signature that a file pointer was not `fclose`'d. Adding an `fclose()` call in an appropriate location will fix this memory leak. 1 point for explaining why it's broken, 1 point for proposing the fix.

14. It means there is an unclosed parenthesis somewhere up above line 63 that, when the interpreter gets to line 63, the interpreter cannot precede because of the syntax problem. This kind of bug really can only be solved by carefully stepping through the code above and making sure all parentheses are properly closed. One of them is assuredly not. 1 point for explaining why it's broken, 1 point for explaining how to root out the problem.

Think Fast!

15. Panel three of the comic is an example of a SQL injection attack, whereby a malicious user enters input that is designed to edit or make a change to the database. This could be fixed by using a function like `htmlspecialchars()` or `mysql_real_escape_string()`. 1 point for identifying the injection attack, 1 point for describing how it could be fixed.

16. Because PHP uses associative arrays, we are able to use actual strings–such as the words being looked up–as indices of the array. Because of this, the need for a hash table is eliminated because lookup is instantaneous. 2 points for an explanation along these lines, 0 points otherwise.

17. The reason for using the fixed-width integers provided in `stdint.h` is to provide for greater cross-portability from system to system. On some older systems, for example, an `int` is actually just 2 bytes while a `long` is 4. On most modern systems though, the difference between the two is merely vestigial and both are 4 bytes. Using fixed width integers guarantees the variable to take up exactly the same amount of memory on all systems. 2 points for an explanation along these lines, 0 points otherwise.

18. If the algorithm runs in $O(n)$, we know it must be inserting at the *end* of each bucket's list. If it runs in $O(1)$, we know it must be inserting at the *beginning* of each bucket's list, instead. 1 point for identifying the reason behind each case.

19. For every file we compress using Huffman's algorithm, we need to package the compressed bits with extra metadata containing at a minimum the frequency table so that when the file is decompressed, the Huffman tree can be recreated. That metadata was the Huffeader in Problem Set 6. For very small files (less than about 1.5KB), that metadata will take up a substantial chunk of memory relative to the rest of the file, so our "compressed" file actually is larger than the decompressed file. 2 points for an explanation along these lines, 0 points otherwise.

20. Using `$_GET` in your code is best for when you want to pass data that doesn't have any sort of security implications or when you want to be able to bookmark the "landing" page. Using `$_POST` is better for passing sensitive information such as a password. The difference between them is that while `$_GET` passes variables/information to the "landing" page via the URL, `$_POST` passes the variables behind the scenes, invisibly, in the HTTP headers. 1 point each for giving: (a) a use of `$_GET`, (b) a use of `$_POST`, and (c) explaining the difference between them.

Ch-ch-ch-changes

21. `chmod 651 file.php`. Score 1 point for using `chmod` and 1 point for identifying that 651 is the proper argument to set the permissions as described.

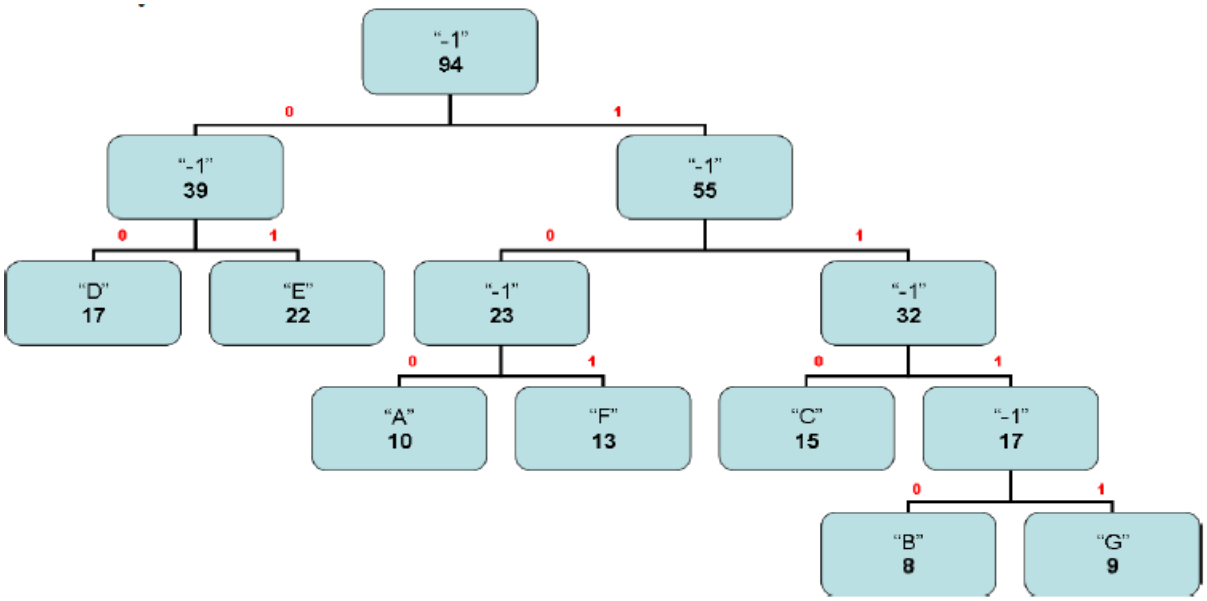
Feeling Listless

22.
```
typedef struct _dllist {
    char x;
    int y;
    struct _dllist *prev;
    struct _dllist *next;
} dllist;
```

    1 point for a structure containing two data fields (each node contains a character field and an integer field), 1 point for properly referring to `struct _dllist` (or equivalent) and not `dllist` for self-referential pointers, and 1 point for proper syntax overall.

I'll Huff, and I'll Puff...

23. The Huffman tree looks like this:



And so the proper encodings are:

A	100	E	01
B	1110	F	101
C	110	G	1111
D	00		

The minimum score for this problem, if attempted, is 4 points. Discretionarily deduct from the remaining 8 points if there are errors.

'Tis the Season.

24.
```
CREATE TABLE xmas
(
   days TINYINT,
   gifts VARCHAR(50)
)
```

Score 2 points if perfect, 1 point if generally correct but some small syntax errors, 0 points for unattempted or completely incorrect.

25. The following goes within the TODO:

```
for($i = 0; $i < 12; $i++)
{
   $result = query("INSERT INTO xmas (days, gifts) VALUES (?, ?)", $days[i], $gifts[i]);
   if($result === false)
   {
      apologize("Error inserting into the database!");
      exit(1);
   }
}
```

Score 1 point for proper looping, 2 points for a proper PDO query, 1 point for apologizing, and 1 point for generally good syntax (using `$` to prefix variables). Using `foreach` or `while` is fine.

26. The following goes within the TODO:

```
<?php

   $rows = query("SELECT * FROM xmas");
   if($rows === false)
   {
      apologize("Error querying database!");
      exit(1);
   }

   foreach($rows as $row)
   {
      print("<tr>");
      print("<td>" . $row["days"] . "</td>");
      print("<td>" . $row["gifts"] . "</td>");
      print("</tr>");
   }

?>
```

Score 1 point for proper looping (using `foreach` is the only proper looping construct in this case), 2 points for a proper query to the database, and 2 points for code within the `foreach` that successfully prints out the database as a table. Not closing rows or columns would NOT be a successful print out.

27. `UPDATE xmas SET gifts = 'Will you marry me?' WHERE days = 12`

Score 2 points if correct, 0 points otherwise.