Practice Quiz 1

CS50 — Fall 2010

Prepared by: Doug Lloyd '09

November 15, 2010

Questions

1. Declare a function that takes an integer as an input and returns `true` if the number is even and `false` if the number is odd.
2. Write the function you declared in Question 1.
3. Modify the function you declared and wrote in Questions 1 and 2 to instead accept a pointer to an integer instead of the integer itself.
4. Modify the function you declared and wrote in Questions 1 and 2 to be a complete program that asks a user to input an integer, tells the user whether it is odd or even, and then repeats this process, until the user inputs `0`.
5. Modify the program you wrote in Question 4 to be a complete program that takes one command-line argument, determines it to be an integer and, if it is an integer, returns `true` if the number is even and `false` if it is odd.
6. Consider the program below, which deals with variable scope:

   ```
   #include <stdio.h>

   int x = 10;

   int f(int x) {
      return x * x;
   }
   int main() {
      printf("The result is %d and x is %d here\n", f(4), x);
   }
   ```

   What is printed at the end of this program?
7. Explain the major difference between selection sort and insertion sort.
8. Write a pseudocode algorithm for binary search.
9. What is the result of `11010100 XOR 10011001`?
10. What is the result of `01001010 AND 10100110`?

11. Write a function `alphabeticalWord()` that takes one argument, `s`, a string, and returns `true` if the letters in `s` are in alphabetical order, and `false` otherwise. You should ignore case.

12. How many steps will it take for a binary search algorithm to look through the array below to find the element 32?

4	6	9	11	14	18	21	25	29	32	37

13. What does the following function do?

```
void mystery(char *foo) {
   if(*foo == '\0')
      printf("\n");
   else {
      printf("%c", *foo);
      mystery(foo+sizeof(char));
   }
}
```

14. State the time complexity of the algorithm given in Question 13 using asymptotic notation.

15. Compare the function in Question 13 to the following one:

```
void *newMystery(char *foo) {
   if(*foo == '\0') {
      printf("\n");
      return;
   }
   else {
      newMystery(foo + 1);
      printf("%c", *foo);
      return;
   }
}
```

    Do they have the same effect?

16. Bubble sort and bi-directional bubble sort (cocktail sort) both run in $O(n^2)$ time, even though when run side-by-side, the latter appears to move quicker. Explain why both run with the same time complexity.

17. Write an algorithm that prompts a user to repeatedly enter a series of `double`s until they type in some sentinel value, and return to them the average of these values. Describe the efficiency of your algorithm using big O notation.

18. Let us consider the following array of integers:

5	8	2	1	6	7	3	4

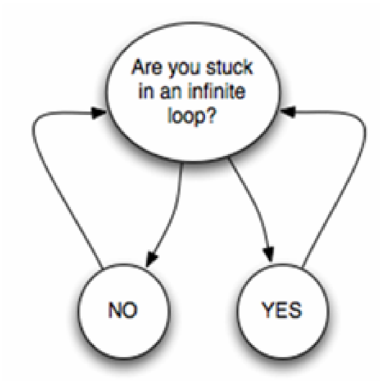
    And the uni-directional bubble sort algorithm, where we bubble the largest values to the right on each pass through, and then return to the front of the array to do the same, until the array is sorted. If we use this algorithm, what does the array look like after one pass?

19. What will the array in Question 18 look like after two passes of the algorithm?

20. What will the array in Question 18 look like after three passes of the algorithm?

21. Write a recursive function that computes the n^{th} Fibonacci number. (The sequence begins $0, 1, 1, 2, 3, 5, 8...$)

22. Explain the following image[1]:



In particular: What is an infinite loop? Write two different lines of C–one using a `while` loop and one using a `for` loop–that create infinite loops.

23. Imagine we have the following functions declared in `pilingUp.c`:

```
void anotherLevel() {
  anotherLevel();
  return;
}

void main() {
  anotherLevel();
  return;
}
```

If we run the program, will it ever terminate? If so, why? If not, why not?

24. How do we compile `pilingUp.c` so that the program's name is `levels`?

Questions 25-27. Compare the speed of the listed "big O" classes, and indicate which one is (generally) faster.

25. Logarithmic time or linear time?

26. Exponential time or polynomial time?

27. Linearithmic (loglinear) time or quadratic time?

28. Say my program contains the following lines of code:

```
int i;
int *ptr;
i = 5;
/* ??? */
```

What line should replace the question marks, so that `ptr` holds the address of `i`?

[1] From http://blog.stevienova.com/wp-content/uploads/LiveWriter/Areyoustuckinaninfiniteloop_AC8D/image.png

29. If we have this line in our program:

    ```
    int array[] = {1,2,3,4};
    ```

    What is the value of the following expression?

    ```
    *(&(*(array+1)))
    ```

30. How do we declare, dynamically, an array of 40 `doubles`?

31. How many different keys for the Vigenère cipher can be produced with an alphabet of k characters?

32. Point out at least four things wrong with the program that appears below:

    ```
    #include <stdio.h>

    #define NUM_INTS 22;

    int main() {
       int i;
       int *myInts = malloc(NUM_INTS * sizeof(int))

       for(i = 0; i <= NUM_INTS; ++i)
          myInts[i] = (i*i);
       for(j = NUM_INTS; j > 0; j--)
          printf("%d\n", myInts[i-1]);

       return 0;
    }
    ```

33. Define a structure for a type that can hold the number of students enrolled, the name of the instructor, and the CUE rating (just the overall score) for a course. Then, demonstrate how to declare an instance of this structure.

34. If we had an instance of the structure declared in Question 33, how would we access the "instructor" field?

35. If, instead, we had a pointer to an instance of the structure declared in Question 33, how would we access the "number of students" field?

36. If you are selling your hard drive on eBay, why might it be a good idea to do a complete format, rather than a "quick format"? (Even if you don't know the precise difference between the two, try to reason based on your knowledge of what happened with David's Flash card!)

37. Demonstrate how to open an ASCII file named "input.txt" for reading.

38. Demonstrate how to open an ASCII file named "output.txt" for writing bytes.

39. In `whodunit.c`, from Problem Set 5, what was the underlying representation of our `BYTE` type? That is, what filled in the blank in the type definition below?

    ```
    typedef ________________ BYTE;
    ```

40. How would you define a new enumerated type called `months` that, understandably enough, enumerates all of the months of the year?

41. What will the following code segment return?

```
#include <stdio.h>

int x = 10;
int y = 20;

int f(int x, int z) {
   y += x;
   x += y;
   return x + z;
}

int main() {
   return f(x+4, y);
}
```

42. How do I dispense $1.43, using as few coins as possible, using the greedy algorithm, if I have the following coin denominations: 1, 5, 10, 20, 25, 100, and 200 cents.

43. Using the same coin denominations as listed in Question 42, how do I dispense $1.43, again using as few coins as possible, using an exhaustive search algorithm?

44. How do I dispense $2.26, using as few coins as possible, using the greedy algorithm, if I have the following coin denominations: 1, 2, 5, 15, 25, 50, 70, and 150 cents.

45. Using the same coin denominations as listed in Question 44, how do I dispense $2.26, again using as few coins as possible, using an exhaustive search algorithm?

46. In `string.h`, we conveniently have the function `strncpy()`, which allows us to copy one string to another. However, the designers forgot to actually define `strncpy()` in `string.c`. Implement, without actually using `strncpy()`, `strcpy()`, `memcpy()`, or any other comparable function, the definition of `strncpy()`. Your definition must conform to the following declaration:

```
strncpy(char *dest, const char *src, int num);
```

where `dest` is the destination string, `src` is the source string, and `num` is the number of characters to copy. You need not worry about `free()`ing any `malloc()`ed memory, should you choose to take that route.

47. Write a function `prime()` that takes one integer parameter `z`, and returns `true` if `z` is a prime number and `false` otherwise. You may use a recursive or an iterative implementation.

48. Draw the Huffman tree for a file with character frequencies as listed in the table below. Assume that ties are broken by prioritizing by ASCII value, where an internal node's ASCII value is -1. What ends up being the encoding of each letter?

A	10	E	22
B	8	F	13
C	15	G	9
D	17		

49. In PHP, we are allowed to use "associative arrays". In an associative array, we can map a string, for example, to another strong. That is, we can index our associative array by saying `array["key"] = "value"`. What is one advantage of an associative array over a numerically-indexed array? What is one disadvantage?

50. What is the prefix property, and how do we know that a coding scheme that adheres to the prefix property is "immediately decodable"?

Questions 51-54. In answering these questions, consider the code at the bottom of the page.

51. On lines 9 and 10, what do the "`r`" and "`w`", respectively, do?
52. On lines 9 and 10, what do the calls to `fopen(...)` return?
53. Explain the flow of information in lines 21 and 22.
54. In line 27, note that `destinationFile` is being opened again. Why do this instead of just keeping it open in the first place?
55. Write a function `deleteList()` that takes a linked list, deallocates all of its memory, and sets the head to `NULL`.
56. Write a function `reverseList()` to reverse a singly-linked list.

```
 1  #include <stdio.h>
 2
 3  int main() {
 4     FILE *sourceFile;
 5     FILE *destinationFile;
 6     char buffer[1000];
 7     int n;
 8
 9     sourceFile = fopen("file.c", "r");
10     destinationFile = fopen("file2.c", "w");
11
12     if(sourceFile == NULL) {
13        printf("Error: cant access file.c.\n");
14        return 1;
15     }
16     else if(destinationFile == NULL) {
17        printf("Error: cant create file for writing.\n");
18        return 2;
19     }
20     else {
21        n = fread(buffer, 1, 1000, sourceFile);
22        fwrite(buffer, 1, n, destinationFile);
23
24        fclose(sourceFile);
25        fclose(destinationFile);
26
27        destinationFile = fopen("file2.c", "r");
28        n = fread(buffer, 1, 1000, destinationFile);
29        printf("%s", buffer);
30
31        fclose(destinationFile);
32        return 0;
33     }
34  }
```

57. The following is an excerpt from the function `insert()`, which attempts to insert a node called `newptr` (already instantiated and initialized) into a linked list pointed to by `first`. This linked list is composed of nodes that point to student structures. But, while debugging, we find that this code frequently hangs at the command line after the first few insertions. Why? You can assume that there are no duplicates in the list.

```
// check for empty list
if(first == NULL)
   first = newptr;

// else check if student belongs at list's head
else if(newptr->student->id < first->student->id) {
   newptr->next = first;
   first = newptr;
}

// else try to insert student in middle or tail
else {
   node *predptr = first;
   while(1) {
      if(predptr->next == NULL) {
         predptr->next = newptr;
         break;
      }

      // check for insertion in middle
      else if(predptr->next->student->id > newptr->student->id) {
         predptr->next = newptr;
         newptr->next = predptr->next;
         break;
      }

      // update pointer
      predptr = predptr->next;
   }
}
```

58. Implement a function `void delete(node *n)` to remove a node from a doubly-linked list. You can assume pointers to the head and tail of the list are global. Remember to account for the cases in which the node you are deleting is the first or last node!

59. Given any binary tree, write a function to count the number of nodes in the tree.

60. In the worst case, separate chaining lookup can take n steps. Give an example of when this would happen.

61. Think about the BIG BOARD from Problem Set 6. Many students who implemented hash tables on Problem Set 6 got different runtimes from one another. Why?

62. Assuming a completely random hash function which can take on m different values, what is the probability of a collision when we enter $n < m$ elements in the table?

63. Write a function to count the number of strings stored in a hash table. Assume the hash table is an array of size `LENGTH` and that it contains pointers to linked lists. The pointers are of type `node *`.

64. Given a binary search tree of integers, the following function is supposed to print out the numbers in sorted, increasing order. But when we run it, the numbers do not appear to be in order at all! What is wrong with it?

```
void printTree(treenode *node) {
   if(node == NULL)
      return;

   printTree(node->left);
   printTree(node->right);
   printf("%d ", node->data);
}
```

65. The following function is intended to locate the node containing an integer specified by the user within a linked list pointed to by the global pointer variable `first`. This code won't compile. Why?

```
void find() {
   // prompt user for number
   printf("Number to find: ");
   int n = GetInt();

   // get list's first node
   node *ptr = first;

   // try to find number
   while(ptr != NULL) {
      if(ptr.n == n) {
         printf("\nFound %d\n", n);
         sleep(1);
         break;
      }
      ptr = ptr.next;
   }
   return;
}
```

66. What assumptions about a bubble sort algorithm can we make if the list is sorted after n operations?

67. Same question as Question 66, except the list is sorted after n^2 operations.

68. What assumptions about a piece of code that inserts into a hash table with separate chaining can we make if the runtime of the algorithm is $O(n)$?

69. Same question as Question 68, except the runtime of the algorithm is $O(1)$.

70. We can reduce the likelihood of collisions by increasing the size of the hash table. What can be a drawback of making the hash table so large that the likelihood of collisions is almost zero?

71. Why is `sizeof()` not a function?

72. Consider the code below:

```
typedef struct _uselessList {
   struct _uselessList *next;
} node;

int main() {
   node *first = (node *) malloc(sizeof(node));
   node *second = (node *) malloc(sizeof(node));
   node *third = (node *) malloc(sizeof(node));
   first->next = second;
   second->next = third;
   free(first);
   return 0;
```

What is wrong with this code?

73. Consider the code below, which prompts a user for strings and enters them into a hash table. Draw a diagram of the hash table that results after a user inputs the following strings in the following order: `lemon`, `clementine`, `peach`, `Cherry`, `apple`, `kiwi`, `banana`,

```
typedef struct _list {
   char *word;
   struct _list *next;
} list;

int main() {
   list *hashtable[10];
   char *word = (char *)malloc(MAX_LENGTH * sizeof(char));

   for(int i = 0; i < 10; i++)
      hashtable[i] = NULL;

   int total, i;

   while(1) {
      printf("Please enter a string");
      word = GetString();
      if(strcmp(word, "") == 0))
         break;
      hashvalue = word[0] % 10;
      list *newptr;
      if((newptr = (list *)malloc(sizeof(list))) == NULL)
         return 1;
      newptr->next = hashtable[hashvalue];
      hashtable[hashvalue] = newptr;
      strcpy(newptr->word, word);
   }
   return 0;
}
```

74. The following code is a security nightmare. Identify the problems and rewrite the code to have the same functionality, but with greater security.

```
#include <stdio.h>

int main(int argc, char **argv) {
   char buf[256], buf2[256];
   printf("Please enter a string\n");
   scanf("%s", buf);
   strcpy(buf2, argv[1]);
   printf("The first command line argument you supplied was \n");
   printf(buf2);
   printf(" the string you entered was ");
   printf(buf);
   return 0;
}
```

75. David has put the following HTML code in his webpage:

```
<IMG SRC=blueflax.jpg ALT="Blue Flax" WIDTH=300 HEIGHT=175 ALIGN=left>
```

Rewrite this as valud XHTML.

76. Cansu is debating where to place the following line of code in her new valid XHTML web page:

```
<link href="css/styles.css" rel="stylesheet" type="text/css" />
```

Where should she put it?

(a) Within `<body>` tags
(b) Within `<a>` tags
(c) Within `<head>` tags
(d) Within `<rel>` tags

77. What does `chmod` do? Feel free to consult the `man` page to look up your answer.

78. When we run `ls -l` on a directory, we may see a line such as the following:

```
-rwx------ 1 username student 1000 2010-08-24 17:27 file.php
```

That means that `file.php` is readable, writable, and executable by you and no one else. If (and don't actually do this) we wanted to modify `file.php` so that it was:

- Readable and writable, but not executable, by you.
- Readable and executable, but not writable, by your group.
- Executable, but neither writable nor readable, by the world.

What octal numbers would fill in the blank in: `chmod ___ file.php`?

79. If, after executing the command as described in Question 78, we ran `ls -l` again, what would replace the question marks in the line:

```
-????????? 1 username student 1000 2010-08-24 17:27 file.php
```

Questions 80-85. In answering these questions, use the table below, called `users`, which is in your SQL database. Write the SQL command asked for in each question to make the appropriate manipulation of the table.

username	**password**	**fullname**
`cs50stud`	`h4ck3r`	`CS50 Stud`
`malan`	`djmftl!`	`David Malan`

80. Grab `malan`'s password.
81. Add `mtucker` to the table with the password "`l33t`" and the name "`Mike Tucker`".
82. Change `malan`'s password to "`n3rd`".
83. Grab the full records of everyone in the table at this point, and arrange them in reverse alphabetical order by username.
84. Return the number of records in the database.
85. Delete `cs50stud` from the table.
86. Say we want to recreate the `users` table (same exact rows and columns) using HTML. Complete the following code:

```
<? $result = mysql_query("SELECT * FROM users"); ?>
<tr><td>username</td><td>password</td><td>fullname</td></tr>
<? while($row = mysql_fetch_array($result)) { ?>
<tr>
<! TODO! REPLACE THIS CODE! -->
</tr>
<? } ?>
```

Questions 87-96. Provide basic definitions for the following terms that demonstrate your understanding.

87. Mutually-recursive functions.
88. Ω
89. Interpreted language (as opposed to a compiled language).
90. Loosely typed.
91. Abstraction.
92. Constructor.
93. FIFO data structure.
94. Binary search tree.
95. Closures.
96. Persistent data.
97. Write a line of code in JavaScript that changes the text of all `div`s with the ID `colorflip` to green when the user clicks a button on the page.
98. Inside of which HTML element (tag) do we "declare" our intent to use JavaScript?

Questions 99-100. Consider the following HTML snippet:

```
<form action="register.php" method="post">
   <table border="0" style="text-align: left;">
      <tr>
         <td>Username:</td><td><input name="username" type="text" /></td>
      </tr>
      <tr>
         <td>Password:</td><td><input name="password" type="text" /></td>
      </tr>
   </table>
   <input type="submit" value="Register!" />
</form>
```

99. If we wanted to store the username in some variable `$username`, what line of PHP code would we write in `register.php` to accomplish this?

100. The password field of this form has an inherent weakness. How would you improve that line of code to make the entered password a little more secure?

101. The government looks to you to write the login page to a top-secret website. (See where a Harvard education can get you?) You decide to use PHP and a MySQL table to store login information. Here's an excerpt of the code you write:

    ```
    $username = $_GET["username"];
    $password = $_GET["password"];
    $result = mysql_query("SELECT * FROM users WHERE username='$username' AND password='$password'");
    ```

    Give two reasons why this code is very weak.

102. Write, in a file called funkystyles.css, code that makes the head element have a purple background, yellow text, and size 30 Times font, and that makes the body element have a green background, red text, and size 20 Verdana font, all of which is surrounded by an orange, 16-pixel wide, dotted border.

103. Why do we term PHP a metalanguage? What is the primary usage of PHP in web design?

104. What are `$_GET`, `$_POST`, and `$_SESSION`?

105. Write a PHP function `dollarBills()`, which assumes an integer input and returns the fewest number of bills it takes to make that amount. (Denominations are 1, 2, 5, 10, 20, 50, & 100)

106. Visit http://www.macloo.com/examples/html/basiccode.htm, and draw the Document Object Model for that website, based on the code that you see.