

Practice Quiz 0

CS50 — Fall 2010

Prepared by: Doug Lloyd '09

October 11, 2010

Below are several questions that are of the sort you will find on the quiz. While this is by no means a comprehensive review of all topics, it will help you study for the quiz. *Of course, you should also attend section, and study the course materials.*

1. Explain why some people might find the following image funny¹:



2. Consider the following segment of code:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    char *myWord = "fish";
    char *myOtherWord = "mouse";
    int myInt = 12345678;
    double myDouble = 3.1415926535;
    printf("%d\n", sizeof(myWord));
    printf("%d\n", sizeof(myWord[1]));
    printf("%d\n", sizeof(myOtherWord));
    printf("%d\n", sizeof(myInt));
    printf("%d\n", sizeof(myDouble));
    printf("%d\n", sizeof(sizeof(myDouble)));
}
```

What will the output of this program be, if compiled on a 64-bit machine?

¹Courtesy of <http://xkcd.com/138/>

3. Calculate in binary:

```
    00101101
+   01101011
-----
```

4. What is hexadecimal (hex)? How does it relate to binary?
5. 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.
6. Write a `for` loop that calculates the sum of the numbers 1 through 10 and stores the result in an integer variable `sum`. Then convert it to a `while` loop.
7. We would like to write the function `mult_binomials()`, which takes two binomials of degree 1 (each represented as a one-dimensional array of 2 ints), and outputs a polynomial of degree 2, represented as a one-dimensional of 3 ints, which is their product. For example, we would represent:

```
4x + 7 as int binomial[2] = {4, 7};
2x + 0 as int binomial[2] = {2, 0};
5 as int binomial[2] = {0, 5};
3x2 + 7x + 5 as int product[3] = {3, 7, 5};
```

...et cetera. Fill in the values for `product[0]` (the x^2 term), `product[1]` (the x term), and `product[2]` (the constant term):

```
int *mult_binomials(int A[2], int B[2]) {

    int *product = (int *) malloc(3 * sizeof(int));

    product[0] = _____;
    product[1] = _____;
    product[2] = _____;

    return product;
}
```

8. Why is it okay that the return type of the function in Question 7 is `int *` instead of an `int` array? Is there a difference? Why or why not?
9. What is the major pointer error being made in Question 7?
10. Explain what typecasting is, and why it might be useful.
11. How would you compile a program saved as `example.c` that uses the function `GetInt()` from the CS50 library such that the resulting file would be named `a.out`? What if we wanted to call it `example1`?
12. What does the line

```
#include <stdio.h>
```

do in a program? Name one function that is declared in `stdio.h`.

13. How would you declare some “container” of a type called `student` containing all of 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 double-precision floating point number to represent their GPA, and
 - A character (A, B, C, D) to represent their class year (freshman, sophomore, junior, senior).
14. Given that we have one of these “containers” described in Question 13 called `student1`, instantiate `student1` for John Harvard, a junior whose ID number is 1636 and who has a 3.78 GPA.

Questions 15-17. Assume that the definition of a `student` is in the file `definition.h`

15. How would you statically create an array of 100 students in your program?
16. It can be wasteful to statically declare this array. Now we have a program `make_students` that allows the user to input the number of student records they want to create. Fill in the line of code that **dynamically** allocates space for these students:

```
#include <stdio.h>
#include <cs50.h>
#include "definition.h"

int main(int argc, char *argv[]) {
    printf("How many records do you want? ");
    int records = GetInt();

    -----;

    return 0;
}
```

17. Convert the program in Question 16 into one that instead is run using `./make_student <number>`. That is, the user inputs on the command line how many records to create. You may assume that `<number>` will consist only of numeric characters. Make sure to do all error checking.
18. Say that at the top of our program we have the following two lines of code:

```
#include <time.h>
#include "clock.h"
```

Why is `time.h` in angle brackets and `clock.h` in quotes? What does this say about the locations of the header files `time.h` and `clock.h`?

19. If we used the following function in our code somewhere, why could we run the risk of a major error?

```
void divide_and_remainder(int x, int y) {
    int q = x/y;
    int r = x%y;

    printf("%d divided by %d is: %d with a remainder of %d\n", x, y, q, r);
    return;
}
```

20. How many times will “hello, world” print to the screen when the following code is executed?

```

for(i = 0; i < 10; i++)
    for(j = 3; j < 303; j += 3)
        printf("hello, world\n");

```

21. David 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 22
    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 = 'F';
    }

    return letterGrade;
}

```

Assuming that the function `round()` takes as its input a quiz score between 0 and 100, and outputs that number rounded down to the nearest 10 (e.g. `round(74)` returns 70 and `round(80)` returns 80), explain why everyone in the class is really upset about their grade.

22. Why is the line marked Question 22 in the code above not equivalent to `quizScore`? After all, in general, $\frac{n}{10} * 10 = n$.
23. Explain what the following for loop does and how it accomplishes it:

```

for (c = 'A'; c <= 'Z'; c++)
{
    printf("%c", c);
}

```

24. What does the following function calculate:

```

int secret(int x, int y) {
    if(y == 0)
        return 1;
    else
        return x * secret(x, y-1);
}

```

25. Sort the following "classes" of functions in the order of generally fastest to generally slowest: *logarithmic, exponential, factorial, constant, polynomial, linear*.

26. Write a function `div_by_3()` which takes one argument, `k`, and returns `true` if `k` is divisible by 3, and `false` otherwise.
27. Write a function `div_by_n()` which takes two arguments, `k` and `n`, and returns `true` if `k` is divisible by `n`, and `false` otherwise.
28. Write the few lines of C code that would print out the multiplication table from 1 to 10 in the following format.

```

1    2    3    4    5    ....  10
2    4    6    8    10   ....  20
....
10   20   30   40   50   ....  100

```

29. Write a function `letter_appears()` which takes two arguments, `word` (a string) and `c`, (a character) and returns the position in `word` that `c` first appears, or 0 if `c` never appears in `word`.
30. What is the worst-case runtime of the function in Question 29?
31. Write a function `gcd()` which takes two arguments, `x` and `y`, both ints, and returns their greatest common divisor. Do not use recursion.
32. Write a function `lcm()` which takes two arguments, `a` and `b`, both ints, and returns their least common multiple.
33. A recursive version of `gcd()` appears below:

```

int gcd(int x, int y) {
    if(y == 0)
        return x;
    else
        return gcd(y, x % y);
}

```

First, convince yourself that the above function works. Then, calculate the worst-case runtime.

34. Write a function called `swap()` that takes two arguments as integers, and swaps their values locally without returning anything. If this function is called from `main()` as:

```

...
swap(a,b);
...

```

where `a` and `b` are integers declared in `main()`, what will happen to the values of `a` and `b`? What is one solution to this problem?

35. If `main()` calls the function `domath()`, and `domath()` calls the function `pow()`, what function will be on the **top** of the stack right before `pow()` returns: `main()`, `domath()`, or `pow()`?
36. Give at least two reasons why `#define` statements are useful.
37. How is `#define` different from `#include`? What, basically, does `#include` do?
38. What does `strlen(a)` return if `a = "hello, world!"`? How many bytes does it take to store that string?
39. Write the lines of code that loop through a string, `s`, and print each character out on a line by itself. For example, if `s = "hello"`, then your program should print:

h
e
l
l
o

40. If a program, `commandline`, is executed as follows from the prompt:

```
> ./commandline pick apple cheese
```

what will be contained in the memory location designated by `argv[1][1]`?

41. Write code that will print out the contents of the following array:

```
int numbers[5][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}, {13, 14, 15} };
```

(a) In the order: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

(b) In the order: 1 4 7 10 13 2 5 8 11 14 3 6 9 12 15

42. Write a function that will test to see if an array of integers is in sequential (ascending) order.

43. Write a function that will compare two strings. Why can't you just compare strings using `==`?

44. What does this program do?

```
#include <stdio.h>
int main(int argc, char* argv[]) {
    printf("%d ", argc);
    if (argc) main(argc - 1, argv);
    else printf("\n");
    return 0;
}
```

45. Assuming you have access to a function `randomInt()`, which generates a pseudorandom nonnegative integer in the range $[0, 2^{32} - 1]$, write a function `rollDie()` that takes one argument `n`, an integer which represents the number of sides of the die, and employ the `randomInt()` function to make `rollDie()` simulate rolling a die with `n` faces.

46. What is the difference between the `.` (dot) and `->` (arrow) operators, with respect to structs?

47. Imagine I execute the following lines of code:

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

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

48. What is the printout of this program? k, pk, ppk, &k, &pk, or &ppk?

```
#include <stdio.h>
int main() {
    int k = 3;
    int *pk;
    int **ppk;

    pk = &k;
    ppk = &pk;

    printf("%x", (*( &(*ppk) )));
}
```

49. From what part of memory is memory obtained from malloc() allocated?

50. Why do we have a pointer type void * if we cannot dereference any pointers of that type?

51. What is the difference between a float and a double?

52. 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):

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

Code	a	b	c	pa	pb	pc
b = a + c;						
a /= c;						
*pc = *pa * a;						
pb = *pb;						
*pc *= *pb;						
a = *pb * *pc;						
pc = &*pa;						

53. What do we call it if we do not free all malloced memory? What is another major error that we can make involving the use of free?

54. Write a recursive function sumOddDigs() that takes as its argument an integer n, and returns the sum of all of the digits of n that are odd. For example, sumOddDigs(424) would return 0, sumOddDigs(425) would return 5, and sumOddDigs(535) would return 13. Convert any negative integers into positive integers before you do any calculations.

55. Write a recursive function sumParityDigs() that takes as its arguments an integer n and a boolean value even. If even is true, sumParityDigs() returns the sum of all of the digits of n that are even. If even is false, sumParityDigs() returns the sum of all of the digits of n that are odd. Convert any negative integers into positive integers before you do any calculations.

56. Predict the error that will occur when you try to compile the following program:

```
#define SIX 6
void main(int argc, char **argv) {
    printf("%d", SIX++);
    return;
}
```

57. What is the value of each variable when these lines of code are executed, knowing that “false” is numerically 0 and “true” is numerically 1?

```
int i = 1, m = 1, n = 2;
i = --m || n++;
```

58. What do we know the following more commonly as?

```
struct mystery3 {
    int a;
    struct mystery3 *b;
    struct mystery3 *c;
};
```

59. Assume we have some program `threeargs` which we run from the command line as `./threeargs ready set go`. What is printed if this is the only content of its `main()`? (Hint: Consider the relationship between pointers and arrays)

```
printf("%c", *+++argv);
```

60. In the Game of Fifteen (in this case, restrict yourself only to the Game of Eight), we provided you with the files `3x3.txt` and `4x4.txt`. Let's walk through a hypothetical scenario for a moment. Imagine that, each time you play the game, a second program runs along with it, `capture_moves`. What this program does is, every time you type in a tile number to move, and that tile number is legal, it records it in a text file. When the game is complete, we close the text file, which is saved as `mymoves.txt`. Then, much like `3x3.txt` and `4x4.txt`, you can run `mymoves.txt` to recreate the moves that you made in that game. Provided below is some skeleton code for `capture_moves`. Help us fill in the blanks to get this program up and running! Imagine that the function `convertToCharacter()` takes an integer in the range $\{1, \dots, 8\}$ and outputs the respective characters in the range $\{'1', \dots, '8'\}$. For simplicity, you may assume that this program has access to the function `won()`, from the Game of Fifteen, as well as access to the variable `tile`.

```
#include <string.h>
#include <stdio.h>
int main(int argc, char *argv[]) {
    ___(1)___ *textFile = ___(2)___("mymoves.txt", "w");

    if(textFile != ___(3)___) {
        while(!won()) {
            fputc(convertToCharacter(tile), ___(4)___);
            fputc('\n', ___(4)___);
        }
    } else
        return ___(5)___;

    fclose(___ (6) ___);
    return 0;
}
```


61. **Challenge question.** Explain what is happening in this program:

```
#include <stdio.h>
#include <string.h>

void baz(char *qux, int bar);

int main(int argc, char *argv[])
{
    if(argc != 2)
        return 1;

    baz(argv[1], strlen(argv[1]));
}

void baz(char *qux, int bar)
{
    char foo1[bar+1];
    strcpy(foo1, qux);
    char foo2[bar+1], *xyzzzy1, *xyzzzy2;

    xyzzzy1 = foo1 + bar - 1;
    xyzzzy2 = foo2;

    while(xyzzzy1 >= foo1)
        *xyzzzy2++ = *xyzzzy1--;

    *xyzzzy2 = '\0';

    printf("%s %s\n", foo1, foo2);

    return;
}
```