Quiz 0

Answer Key

Answers other than the below may be possible.

Multiple Choice.

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

True or False.

6. T or F
7. T
8. F
9. T
10. T or F

Itching for Week 0?

11.

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

12.

```
#include <stdio.h>

int
main(void)
{
    int n = 10;
    while (n > 0)
    {
        printf("%d", n);
        n--;
    }
    printf("Blastoff!");
}
```

Role Reversal.

13. This program always prints `zero`, no matter its input. Removing the semicolon on the same line as `if` fixes the problem.

14. This program always prints `one`, no matter its input. Changing the `=` to `==` in the program's condition fixes the problem.

Big OMG.

15.

Algorithm	***O***	**Ω**
sorting an array with Merge Sort	$n \log n$	$n \log n$
sorting an array with Selection Sort	n^2	n^2
inserting into a **sorted** linked list	n	1
searching a **sorted** array with Binary Search	$\log n$	1
searching a **sorted** linked list with Linear Search	n	1

Bit Wise.

16.

type	**bits**
`char`	8
`char *`	32
`double`	64
`double *`	32
`int`	32
`int *`	32
`unsigned int`	32
`long long`	64

Temp Variables.

17. Because both `9` and `5` are of type `int`, `(9 / 5)` evaluates to `1`, since the remainder is truncated. And so the formula currently implemented is F = 1 × C + 32 = C + 32. To fix the problem, it suffices to rewrite `9` as `9.0`, rewrite `5` as `5.0`, or cast either to a `float`.

This is CS50.

18. Whereas the `#include` (a pre-processor directive) informs `gcc` that a program might call functions declared in `cs50.h`, the `-lcs50` (a linker flag) tells `gcc` to incorporate the bits that result from compiling `cs50.c` (wherever they are) into the program.

Quickies.

19. Pseudocode is an amalgam of English- and code-like syntax used to define algorithms or outline programs in a language-neutral way.

20. Whereas `\n` moves a cursor down to a new line, `\r` moves a cursor to the beginning of the current line. In the context of files, Linux uses `\n` to end lines, Mac OS uses `\r`, and Windows uses `\r\n`.

21. Because an `int` is, by default, `signed`, it's intended to represent negative and positive numbers alike. While half of its range is allocated toward non-negative numbers (0 through $2^{31} - 1$), the rest is reserved for negative numbers.

22. You can trigger a segfault by indexing into an array far beyond its bounds, by dereferencing an invalid (*e.g.,* garbage or `NULL`) pointer, and even by calling a function recursively too many times (the result of which is that the stack overruns the heap).

User Error.

23. The user must have inputted something other than a floating-point number (potentially preceded and followed by whitespace), the result of which is that the input does not match the format string's expectation of a leading `%f`, and so neither `f` nor `c` get filled with a value.

24. The user must have inputted a floating-point number (potentially preceded and followed by whitespace), followed by one or more non-numeric characters, the result of which is that the input matches the whole format string, and so both `f` and `c` get filled with values.

Programmer Error.

25. Because the parameter, `c`, to `capitalize` is passed by value, the function receives a copy of whatever character is passed in. That character is indeed capitalized within the scope of that function, but as soon as the function returns, the capitalized character is lost (because it's popped off the stack). This problem can be fixed by having `capitalize` return the capitalized character, as in the below.

```
char
capitalize(char c)
{
    if (islower(c))
        return toupper(c);
    else
        return c;
}
```

With `capitalize` so defined, the caller must now retain this return value, as in the below.

```
c = capitalize(c);
```

Alternatively, `capitalize` can be declared as expecting a pointer, as in the below.

```
void
capitalize(char *c)
{
    if (islower(*c))
        *c = toupper(*c);
}
```

With `capitalize` so defined, the caller must now call the function as follows.

```
capitalize(&c);
```

Bottle(s).

26. Because the `s` in line 15 inside of a branch, its scope is limited to lines 14 through 16; although `s` is assigned a value, that value goes unused. Similarly does the `s` in line 19 only exist between lines 18 and 20; its value, too, goes unused. And so in line 21, which outside of those scopes, `s` appears to be undeclared.

27. This program is intended to print `bottle` if the user inputs `1`, else `bottles` if the user inputs `0` or an integer greater than `1`.

Pointer Fun without Binky.

28. `GetString` returns the address in memory of a string inputted by a user. And so if `GetString` is called twice, it returns two different (*i.e.*, unequal) addresses, one for each string, even if both happen to comprise the same characters.

29.

statement	what would be printed
`printf("%d", i);`	1
`printf("%d", j);`	3
`printf("%d", *p);`	3
`printf("%d", k);`	7

30.
```
int
strlen(char *s)
{
    int n = 0;
    if (s == NULL)
        return n;
    for (int i = 0; s[i] != '\0'; i++)
        n++;
    return n;
}
```

(Proto)typical Alex.

31.

prototype	function

`bool isalnum(char c)`	[redacted]
	`{`
	`  if (isalpha(c) || isdigit(c))`
	`  return true;`
	`  else`
	`  return false;`
	`}`

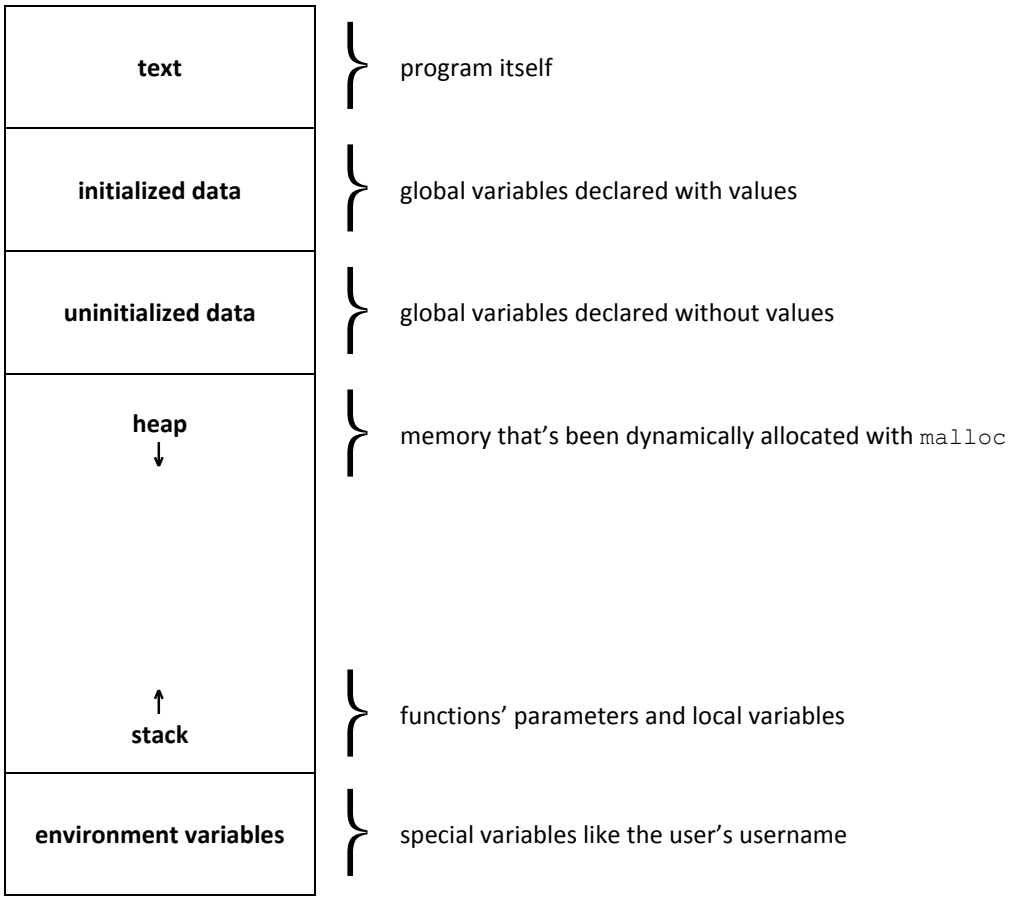
`char tolower(char c)`	[redacted]
	`{`
	`  if ('A' <= c && c <= 'Z')`
	`  return c - 'A' + 'a';`
	`  else`
	`  return c;`
	`}`

`bool isupper(char c)`	[redacted]
	`{`
	`  if ('A' <= c && c <= 'Z')`
	`  return true;`
	`  else`
	`  return false;`
	`}`

`bool isdigit(char c)`	[redacted]
	`{`
	`  if ('0' <= c && c <= '9')`
	`  return true;`
	`  else`
	`  return false;`
	`}`

Ah, memories.

32.



Argh.

33.

statement	what would be printed
`printf("%s", argv[0]);`	`./a.out`
`printf("%c", argv[0][1]);`	`/`
`printf("%c", argv[1][0]);`	`f`
`printf("%s", &argv[1][1]);`	`oo`
`printf("%d", strlen(argv[0]));`	`7`
`printf("%d", strlen(&argv[0][2]));`	`5`
`printf("%d", argc);`	`4`

From Floor to Ceiling.

34.
```
int
floor(float x)
{
    return (int) x;
}
```

35.
```
int
ceil(float x)
{
    if (x - (int) x > 0.0)
        return (int) (x + 1.0);
    else
        return (int) x;
}
```

PC LOAD LETTER.

36.
```
double
steal(double interest)
{
    return interest - round(interest * 100) / 100;
}
```