Quiz 0

Answer Key

Answers other than the below may be possible.

Multiple Choice.

0. b
1. b

***O*(MG).**

2.

	Ω	***O***
Bogo Sort	n	∞
Bubble Sort	n	n^2
Insertion Sort	n	n^2
Linear Search	1	n
Merge Sort	$n \log n$	$n \log n$
Selection Sort	n^2	n^2

Phew, Scratch.

3.
```
#include <stdio.h>

int main(void)
{
    for (int i = 50; i >= 0; i--)
    {
        printf("%i\n", i);
    }
}
```

4.
```
#include <stdio.h>

void cough(void);

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        cough();
    }
}

void cough(void)
{
    printf("cough\n");
}
```

Itsa Mario again.

5.
```
#include <stdio.h>

int main(void)
{
    for (int i = 1; i <= 7; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

CS50 Library 2.0.

6.
```
int GetPositiveInt(void)
{
    int n;
    while (true)
    {
        n = GetInt();
        if (n >= 1)
        {
            break;
        }
        else
        {
            printf("Retry: ");
        }
    }
    return n;
}
```

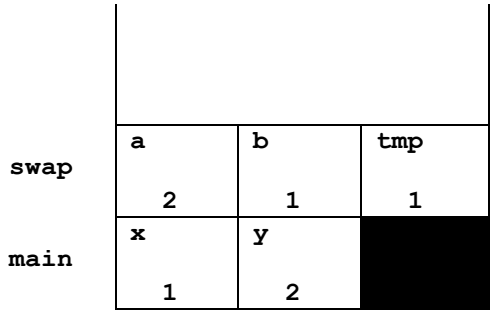
7.
```
int RandomInt(int a, int b)
{
    return drand48() * (b - a) + a;
}
```

Swapfest.

8.

	x	y	a	*a	b	*b	tmp
1 →	1						
2 →	1	2					
3 →			0x123	1	0x127	2	
4 →			0x123	1	0x127	2	1
5 →			0x123	2	0x127	2	1
6 →			0x123	2	0x127	1	1
7 →	2	1					

9.



#include?

10.
```
int atoi(char* s)
{
    if (s == NULL)
    {
        return 0;
    }
    int value = 0;
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (s[i] < '0' || s[i] > '9')
        {
            return 0;
        }
        value *= 10;
        value += s[i] - '0';
    }
    return value;
}
```

11.
```
int strlen(char* s)
{
    if (s == NULL)
    {
        return 0;
    }
    int length = 0;
    for (char* t = s; *t != '\0'; t++)
    {
        length++;
    }
    return length;
}
```

Switching gears.

12.
```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n = GetInt();
    if (n == 1 || n == 2)
    {
        printf("small\n");
    }
    else if (n == 3)
    {
        printf("medium\n");
    }
    else if (n == 4 || n == 5)
    {
        printf("large\n");
    }
}
```

Did you mean: *recursion*

13. `5+4+3+2+1+0=15`

14. Because sigma calls itself recursively, each time passing itself `m - 1`, sigma won't hit its base case if initially passed `-5`, since m will not hit `0` (at least not until `m` happens to wrap around from `-2147483648` to `0` because of overflow). And so the recursive calls' stack frames will eventually overrun the heap, inducing a segfault.

This is not 50.

15. $1 \cdot 32 + 1 \cdot 16 + 1 \cdot 1 = 49$

Making progress.

16. During preprocessing, directives like `#include` are processed, the result of which is to insert the contents of header files (e.g., `stdio.h`) and any declarations therein into the file being pre-processed. During compiling, the preprocessed C code is translated into assembly instructions, possibly with optimizations applied. During assembling, the assembly instructions are translated into machine code and stored in an object (`.o`) file. During linking, the object code for `main` is combined with (i.e., linked against) object code from other files (e.g., C's standard library), the result of which is an executable file.

Making sense.

17. Because `50` and `100` are both of type `int`, the expression `50 / 100` evaluates to an `int` as well, in which case everything after the decimal is discarded, and so `0.5` becomes `0`. It is that `0` that's then implicitly cast to a `float`, stored in `dollars`, and printed to `2` decimal places, and so the `0` is printed as `0.00`.

18. `./think`

Short answers.

19. Selection Sort doesn't know which of its not-yet-sorted elements is smallest until it's traversed them all, since the smallest might be the last. To sort n elements, then, Selection Sort must look at n elements, then $n - 1$ elements, then $n - 2$ elements, and so forth, which adds up to $n(n + 1)/2$, which is on the order of n^2.

20. When `GetString` gets a `string` from a user, it allocates memory for that `string` on the heap and returns a pointer thereto. But only recently did we learn that we should really be calling `free` on that pointer when done with the `string` in order to return the memory to the operating system. By not calling `free`, we've been leaking (allocating and never de-allocating) memory.

21. Some algorithms are, by their own definition, recursive (e.g, Merge Sort), in which case it's straightforward (and thus convenient) to implement them in code recursively. An iterative implementation, by contrast, might take more time to write and might even prove less readable.

22. Recursive implementations of algorithms tend to consume more memory than necessary if recursive calls' frames pile up on the stack (without being optimized away by a compiler). They are also vulnerable to stack overflow if more recursive calls are made than can even fit on the stack. In such cases, an iterative implementation might prove more memory-efficient and more capable of handling large inputs.

23. Whereas Caesar's cipher only uses a one-byte key (for which there are 256 possible values), Vigenère's cipher uses an n-byte keyword (for which there are 2^{8n} possible values, given that there are 8 bits in a byte). Cracking Vigenère's cipher, as by trying all possible keys, therefore requires more effort than does cracking Caesar's cipher (assuming n is greater than 1), and so Vigenère's cipher is considered more secure.

Ponies.

24. The classmate has called `printf` but has omitted

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

    atop the program's file. Adding that line should fix.

25. The classmate has tried to use a variable, `n`, without first declaring it. (Or, even if declared somewhere, it's at least not in scope.) Declaring `n` within the same scope in which it's being used (or globally) should fix.

26. Try it (in Hangouts)!