Quiz 0

Answer Key

Answers other than the below may be possible.

Multiple Choice.

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

True or False.

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

Dang, `clang`.

8. The programmer has called `GetString` but has omitted

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

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

9. The programmer 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.

Rapid Fire.

10. 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.

11. `gdb` lets you pause execution of a program on any line by setting "breakpoints" so that you can step through code line by line. `gdb` also lets you view (and change) the values of variables while a program is running without having to resort to `printf`.

12. Anytime you allocate memory with `malloc` (or functions like `GetString` that, in turn, call `malloc` and return pointers to the allocated memory), you should call `free` when done using that memory.

O hai, Scratch.

13.
```
#include <stdio.h>

int main(void)
{
    for (int n = 1; n <= 5; n++)
        printf("%d\n", n);
    printf("I've got the same combination on my luggage!\n");
    return 0;
}
```

This is 50.

14. $1 \cdot 32 + 1 \cdot 16 + 1 \cdot 2 = 50$

I C what you did there.

15. The declaration of `printf` (including its return type and parameters).

16. That main does not expect any command-line arguments.

17. That the program executed successfully without any errors. By contrast, non-`0` values generally indicate failures of some sort.

18. Because 1 and 10 are both of type `int`, the expression `1 / 10` evaluates to an `int` as well, in which case everything after the decimal is discarded, and so `0.1` becomes `0`. It is that `0` that's then implicitly casted to a `float`, stored in `answer`, and printed to `1` decimal place, and so the `0` is printed as `0.0`.

19. A float is inherently imprecise, because with only finitely many bits (32 on a 32-bit architecture like the CS50 Appliance), a `float` can only represent a finite number of real numbers. `0.100000000000000000000000000000000000000000000000` is not among those numbers, and so the nearest approximation is what's stored in answer, which happens to be `0.100000001490116119384765625000000000000000000000000`.

20. Because this program keeps incrementing `i`, it eventually "overflows" the `int`, flipping one too many bits, at which point its bits happen to represent a negative value.

21. Even though this program's conditions are logically mutually exclusive, the program nonetheless evaluates every each of the parenthetical conditions. Re-writing those conditions as follows would eliminate the inefficiency.

```
if (n >= 1 && n <= 3)
    printf("You picked a small number.\n");
else if (n >= 4 && n <= 6)
    printf("You picked a medium number.\n");
else if (n >= 7 && n <= 10)
    printf("You picked a big number.\n");
else
    printf("You picked an invalid number.\n");
```

22. This program declares `i` on line 9, inside of the `do` block, but then tries to use it on line 11, at which point it's out of scope. Declaring `i` before line 6 would eliminate the bug.

23. `GetString` returns the address of the first `char` in a `string`. Even though a user might type the same English word twice, each `string` will be stored in a different location in memory, and so the addresses in `s` and `t` will be different.

***O*(MG).**

24.

	Ω	***O***
Selection Sort	n^2	n^2
Merge Sort	$n \log n$	$n \log n$
Insertion Sort[1]	n	n^2
Binary Search	1	$\log n$
Linear Search	1	n

[1] Or Bubble Sort.

Stack Cats.

25. `stackcats`

Binky Pointer Fun.

26. 18

27. Even though `y` was declared on line 9, it hadn't yet been assigned a value, and so it contained garbage (i.e., an invalid address). Trying to store a value at that invalid address induced the crash.

28.

```
1  #include <stdlib.h>
2
3  int main(void)
4  {
5       // declare a pointer to an int called x
6       int* x;
7
8       // declare a pointer to an int called y
9       int* y;
10
11      // allocate space for an int and store its address in x
12      x = malloc(sizeof(int));
13
14      // store 42 in the allocated location
15      *x = 42;
16
17      // CRASH
18      *y = 13;
19
20      // store in y the address that's in x
21      y = x;
22
23      // store 13 in the allocated location
24      *y = 13;
25
26      return 0;
27 }
```

```
#include <uhhh.h>
```

29.

```
char toupper(char c)
{
    if (c >= 'a' && c <= 'z')
        return c - ('a' - 'A');
    else
        return c;
}
```

30.

```
int strlen(string s)
{
    if (s == NULL)
        return 0;
    int n = 0;
    while (s[n] != '\0')
        n++;
    return n;
}
```

Swapfest.

31.

	x	y	a	b	tmp
1 →	1				
2 →	1	2			
3 →			1	2	
4 →			1	2	1
5 →			2	2	1
6 →			2	1	1
7 →	1	2			

32.

swap	**a** 10	**b** 14	**tmp** 1
main	**x** 2	**y** 1	

File I/O.

33.

```
void hire(staff s)
{
    FILE* file = fopen("staff.csv", "a");
    fprintf(file, "%s,%s,%s\n", s.last, s.first, s.email);
    fclose(file);
}
```