Multiple Choice.

For each of the following questions, circle the letter (a, b, c, or d) of the one response that best answers the question; you need not explain your answers.

0. (1 point.) How many desk lamps do you need to represent the decimal number 7 in binary?

   a. 2
   b. 3
   c. $\log_2 7$
   d. 7

1. (1 point.) How many times can you tear a phonebook with 128 pages (i.e., sheets of paper) in half, each time throwing away one of the halves, before only one page remains?

   a. 6
   b. 7
   c. 10
   d. 64

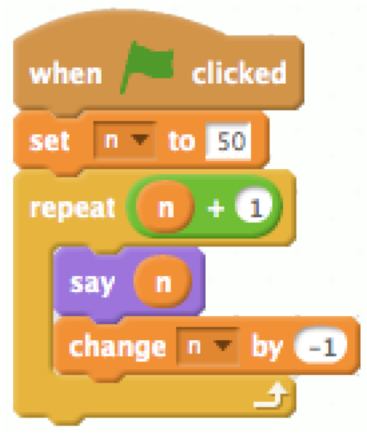
***O*(MG).**

2. (5 points.) Complete the table below by specifying lower (Ω) and upper (*O*) bounds on each algorithm's running time. Assume that the input to each algorithm is an array of size *n*. We've plucked off two cells for you. For the curious, Bogo Sort (otherwise known as Stupid Sort) randomly orders an array, checks if it's sorted, and repeatedly tries again if it's not. More formally, "it serves as a sort of canonical example of awfulness."

	Ω	***O***
Bogo Sort	*n*	∞
Bubble Sort		
Insertion Sort		
Linear Search		
Merge Sort		
Selection Sort		

Phew, Scratch.

3. (4 points.) Consider the Scratch script below.

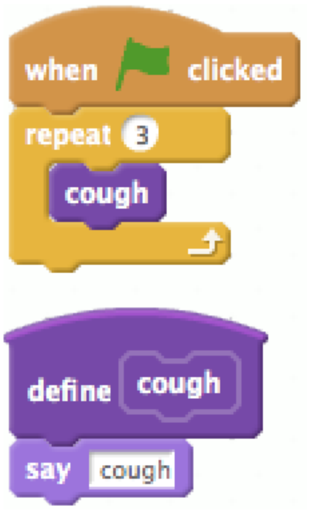


In the space below, translate the script into a C program that's functionally the same (albeit in a command-line environment); it needn't be structurally the same. Assume that Scratch's **say** block translates to `printf` in C, though any call to `printf` should include a trailing `\n`. And recall that **change n by -1** means to decrement **n** by 1.

```
#include <stdio.h>

int main(void)
{
```

4. (4 points.) Consider the Scratch scripts below.

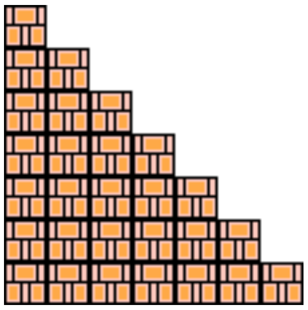


In the space below, translate the scripts into a C program with two functions, `main` and `cough`, that's functionally the same (albeit in a command-line environment); it needn't be structurally the same. Assume that Scratch's **say** block translates to `printf` in C, though any call to `printf` should include a trailing `\n`.

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

Itsa Mario again.

5. (4 points.) Toward the end of World 2-3 in Nintendo's Super Mario Brothers 3, Mario must descend a "half-pyramid" of blocks (unless he flies over it). Below is a screenshot.



Complete the implementation of the program below in such a way that it recreates this particular half-pyramid using hashes (`#`) for blocks. No need for user input; you may hard-code the half-pyramid's height (`7`) into your program.

```
#include <stdio.h>

int main(void)
{
```

CS50 Library 2.0.

6. (6 points.) Consider the program, `positive`, below.

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

int main(void)
{
    printf("Positive integer please: ");
    int n = GetPositiveInt();
    printf("Thanks for the %i!\n", n);
}
```

Consider how this program and, in turn, `GetPositiveInt` are meant to behave, as per the below, wherein underlined text represents some user's input.

```
jharvard@appliance (~/Dropbox/quiz0): ./positive
Positive integer please: -50
Retry: 0
Retry: 50
Thanks for the 50!
```

If only `GetPositiveInt` actually existed! Suppose that you'd like to implement it for us for the next version of the CS50 Library. Complete the implementation of `GetPositiveInt` below using `GetInt` (which does exist!) in such a way that the program above would indeed behave per the input and output above. Note that it's `main`, not `GetPositiveInt`, that's prompting the user with `Positive integer please:`. And be sure that `GetPositiveInt` only prompts the user with `Retry:` if the user fails to provide a positive integer.

```
int GetPositiveInt(void)
{
```

7. (4 points.) Consider the program, `random`, below.

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

int main(void)
{
    int n = RandomInt(0, 50);
    printf("Here's a %i!\n", n);
}
```

Consider how this program and, in turn, `RandomInt` are meant to behave, as per the below, wherein underlined text represents some user's input.

```
jharvard@appliance (~/Dropbox/quiz0): ./random
Here's a 42!
```

If only `RandomInt` actually existed! Suppose that you'd like to implement it for us for the next version of the CS50 Library. Complete the implementation of `RandomInt` below in such a way that, given `a` and `b`, the function returns a pseudorandom integer between `a` (inclusive) and `b` (exclusive) using `drand48`. Recall that `drand48` returns "nonnegative double-precision floating-point values uniformly distributed between" `0.0` (inclusive) and `1.0` (exclusive). You may assume that `b` will be greater than `a`. And you may assume both that `srand48` has already been called for you elsewhere and that `stdlib.h` (in which `drand48` and `srand48` are declared) has been `#include`'d for you elsewhere (and that `_XOPEN_SOURCE` is `#define`'d as needed).

```
int RandomInt(int a, int b)
{
```

#include?

10. (6 points.) Suppose that you've forgotten which header file declares `atoi`, and so you need to re-implement it yourself. Argh. Without calling any functions other than `strlen` (which you may call if you'd like), complete the implementation of `atoi` below in such a way that it converts `s` (e.g., `"123"`) to an `int` (e.g., `123`). If `s` happens to be `NULL`, or if `s` contains any character that isn't `'0'` through `'9'`, your implementation of `atoi` should return `0`. Otherwise, you may assume that `s` represents a non-negative integer that, when converted, will fit inside of an `int` without overflow. No need to `#include` any files (even if you call `strlen`).

```
int atoi(char* s)
{
```

Switching gears.

12. (4 points.) Consider the program below.

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

int main(void)
{
    int n = GetInt();
    switch (n)
    {
        case 1:
        case 2:
            printf("small\n");
            break;

        case 3:
            printf("medium\n");
            break;

        case 4:
        case 5:
            printf("large\n");
            break;
    }
}
```

Complete the re-implementation of this program below without using `switch` in such a way that it still behaves exactly the same.

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

int main(void)
{
    int n = GetInt();
```

This is not 50.

15. (2 points.) Convert the binary number below to decimal. Show any work (i.e., any arithmetic).

```
0 0 1 1 0 0 0 1
```

Making sense.

17. (2 points.) Consider the program below.

```
#include <stdio.h>

int main(void)
{
    int cents = 50;
    float dollars = cents / 100;
    printf("%.2f\n", dollars);
}
```

When executed, this program prints

```
0.00
```

which is not how much money we have! In no more than three sentences, explain why this program thinks that `50` cents divided by `100`, printed to `2` decimal places, is something other than `0.50`.

Short answers. (2 points each.)

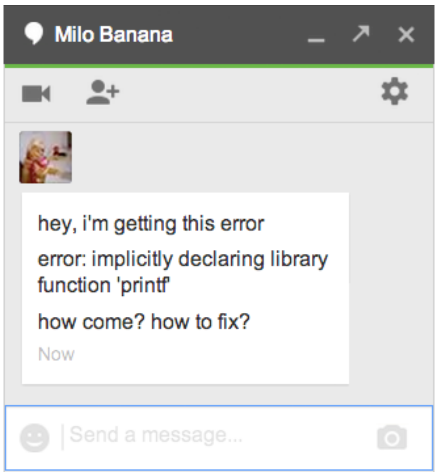
Answer each of the questions below in no more than three sentences.

19. Why is Selection Sort's running time in $\Omega(n^2)$ even when its input is already sorted?

23. In what sense is Vigenère's cipher more secure than Caesar's cipher?

Ponies.

24. (2 points.) Suppose that a classmate has just sent you the message at right. Without seeing your classmate's code, propose what your classmate has done wrong and how to fix it.



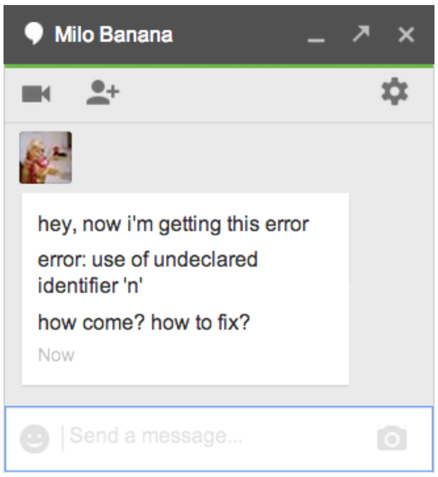
hey, i'm getting this error

error: implicitly declaring library function 'printf'

how come? how to fix?

Now

25. (2 points.) Suppose that the same classmate has just sent you the message at right. Without seeing your classmate's code, propose what your classmate has done wrong and how to fix it.



hey, now i'm getting this error

error: use of undeclared identifier 'n'

how come? how to fix?

Now