Quiz 0

out of 84 points

Print your name on the line below.

_____

Do not turn this page over until told by the staff to do so.

This quiz is "closed-book." However, you may utilize during the quiz one two-sided page (8.5" × 11") of notes, typed or written, and a pen or pencil, nothing else.

Scrap paper is included at this document's end.
Unless otherwise noted, you may call any functions we've encountered this term in code that you write.
You needn't comment code that you write, but comments may help in cases of partial credit.
If running short on time, you may resort to pseudocode for potential partial credit.

Circle your teaching fellow's name.

Alisa Nguyen
Allison Buchholtz-Au
Angela Li
Armaghan Behlum
Ben Shryock
Bo Ning Han
Casey Grun
Chi Zeng
Christopher Bartholomew
Chris Mueller
Cynthia Meng
Dan Bradley
Daven Farnham
David Kaufman
Dianna Hu
Doug Lloyd
Ed Hebert
Gabriel Guimaraes
George Wu
Hannah Blumberg
Ian Nightingale
Jackson Steinkamp
Jared Pochtar
Jason Hirschhorn
Joe McCormick
Jonathan Marks
Jonathan Miller
Jordan Jozwiak
Joseph Ong
Julian Salazar
Karen Xiao
Katryna Cadle
Keenan Monks
Kendall Sherman
Kevin Mu
Kevin Schmid
Levi Roth
Lucas Freitas
Luciano Arango
Lucy Cheng
Lydia Chen
Marshall Zhang
Mike Rizzo
Patrick Schmid
R.J. Aquino
Rhed Shi
Rob Bowden
Roger Zurawicki
Ryan Lee
Saheela Ibraheem
Sharon Zhou
Tim McLaughlin
Tomas Reimers
Travis Downs
Tyler Morrison
Varun Sriram
Wesley Chen
William Hakim
William Xiao
Zamyla Chan

for staff use only

final score out of 84

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		

for staff use only
–

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.

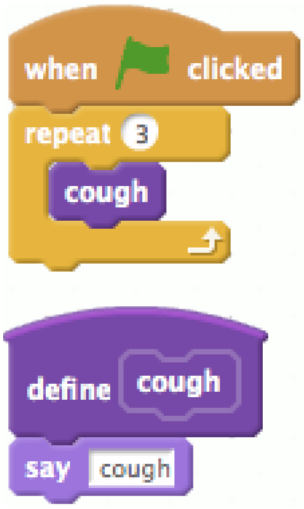
```
#include <stdio.h>

int main(void)
{
```

for staff use only

–

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`.

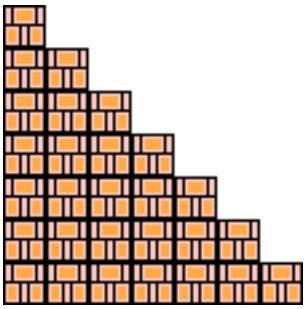
```
#include <stdio.h>
```

for staff use only

–

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)
{
```

for staff use only

–

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)
{
```

for staff use only

–

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)
{
```

for staff use only

–

Swapfest.

8. (6 points.) Consider the program below, between whose lines some numbered arrows have been drawn for the sake of discussion.

```
void swap(int* a, int* b)
{
   3→
    int tmp = *a;
   4→
    *a = *b;
   5→
    *b = tmp;
   6→
}

int main(void)
{
    int x = 1;
   1→
    int y = 2;
   2→
    swap(&x, &y);
   7→
}
```

Suppose that each of the numbered arrows represents a moment in time during this program's execution. And suppose that `&x` (i.e., the address of `x`) is `0x123` and that `&y` (i.e., the address of `y`) is `0x127`.

Record in the blank boxes below the values in scope at each moment in time. For instance, if the program's execution is paused at numbered arrow **2**, the value of `x` would be `1`, and the value of `y` would be `2`. Boxes for values not in scope have been blacked out. Keep in mind that `a` represents the value in `a`, while `*a` represents the value at `a`. The same holds for `b` and `*b`.

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

for staff use only

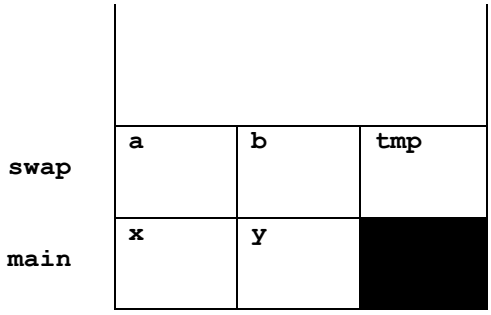
–

9. (5 points.) Consider the similar, but different, program below, between two of whose lines just one numbered arrow has been drawn for the sake of discussion, albeit numbered **6** for consistency.

```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
  6→
}

int main(void)
{
    int x = 1;
    int y = 2;
    swap(x, y);
}
```

Suppose that this program's execution is paused at numbered arrow **6** and that the diagram below represents the program's stack frames at that moment. Record in the blank boxes below the values of `x`, `y`, `a`, `b`, and `tmp` at that moment. (Recall that, even though `x` and `y` are out of scope, they still exist in memory.)



for staff use only

–

#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)
{
```

for staff use only

–

11. (6 points.) Suppose that you've also forgotten which header file declares `strlen`, and so you need to re-implement it yourself (even if you didn't just use it). Bah. Even worse, neither `[` nor `]` currently works on your keyboard (or pencil or pen). Without calling any functions at all and without using any square brackets, complete the implementation of `strlen` below using pointer arithmetic in such a way that the function returns the length of `s`. If `s` happens to be `NULL`, your implementation of `strlen` should return `0`.

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

for staff use only

–

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();
```

for staff use only

–

Did you mean: *recursion*.

Consider the program below.

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

int sigma(int m)
{
    printf("%i", m);
    if (m == 0)
    {
        printf("=");
        return 0;
    }
    else
    {
        printf("+");
        return (m + sigma(m - 1));
    }
}

int main(void)
{
    int n = GetInt();
    int answer = sigma(n);
    printf("%i\n", answer);
}
```

13. (2 points.) Suppose that a user runs this program, inputting `5` (followed by Enter) when prompted by `GetInt`. Exactly what does the program print before exiting?

14. (2 points.) Suppose that a user runs this program, inputting a negative value like `-5` (followed by Enter) when prompted by `GetInt`. Why might the program segfault?

for staff use only

–

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 progress.

16. (4 points.) Consider the source code below.

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

int main(void)
{
    string s = GetString();
    printf("hello, %s\n", s);
}
```

Recall that the process of "compiling" this source code into an executable program via `make` actually involves four distinct phases, though not necessarily in this order: assembling, compiling, linking, and preprocessing. With respect to this program, describe what happens during each of those phases, in at least one sentence per phase, in the order in which those phases occur.

for staff use only

–

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`.

18. (2 points.) Consider the program, `think`, below.

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

int main(int argc, char* argv[])
{
    for (int i = 0, n = strlen(argv[0]); i < n; i++)
    {
        printf("%c", argv[0][i]);
    }
}
```

Suppose that this program is executed as follows.

```
./think twice
```

Exactly what would be printed?

for staff use only

–

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?

20. In what sense have we been leaking memory when using `GetString` (at least in most lectures)?

21. When some algorithm could be implemented in code either iteratively or recursively, why might you want to implement it recursively?

22. When some algorithm could be implemented in code either iteratively or recursively, why might you want to implement it iteratively?

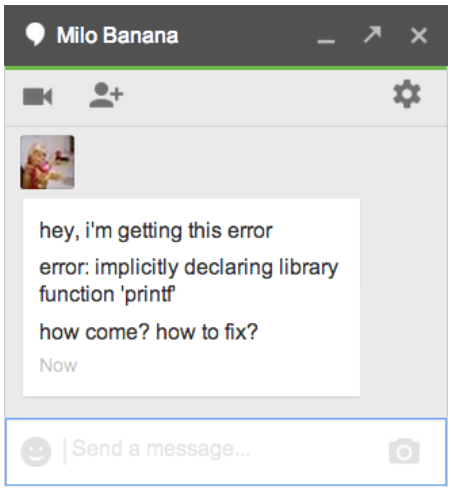
23. In what sense is Vigenère's cipher more secure than Caesar's cipher?

for staff use only

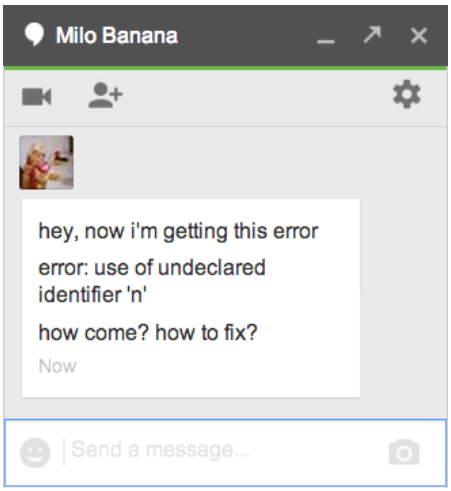
–

Ponies.

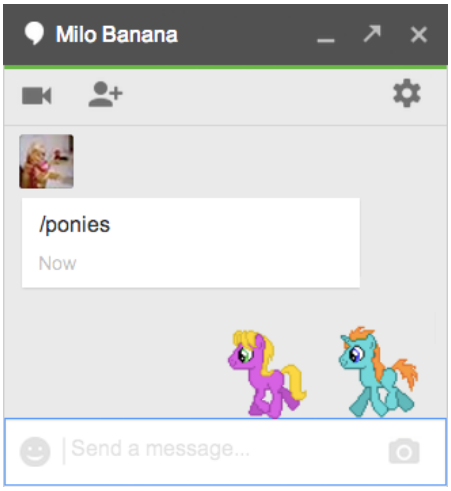
24. (2 points.) Suppose that a classmate has just sent you the message at right. Without seeing your classmate's code, propose what your classmate has done wrong and how to fix it.



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.



26. (0 points.) Now suppose that your classmate has just sent you the message at right. D'aww.



for staff use only

–

Scrap Paper.

Nothing on this page will be examined by the staff unless otherwise directed.