Quiz 0

out of 83 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, assume that any problems herein refer to C.
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.

Circle your teaching fellow's name.

Alex Chang
Ali Nahm
Alisa Nguyen
Angela Li
Balaji Pandian
Bannus Van der Kloot
Ben Shryock
Blake Walsh
Bo Han
Casey Fleeter
Casey Grun
Chris Gerber
Chris Mueller
Christopher Bartholomew
Conner Dalton
Cynthia Meng
Dan Bradley
Daven Farnham
David DiCiurcio
Doug Lloyd
Elena Agapie
Emmet Jao
Iva Milo
Jackson Steinkamp
Jacob Pritt
Jelle Zijlstra
Jimmy Sun
Joe McCormick
John Mussman
Jonathan Miller
Jordan Jozwiak
Joseph Ong
Joy Ming
Julia Mitelman
Jun S. Lee
Karen Xiao
Katryna Cadle
Kevin Mu
Kevin Schmid
Komal Syed
Larry Ehrhardt
Lauren Carvalho
Levi Roth
Lexi Ross
Lucas Freitas
Mark Grozen-Smith
Meg Quintero
Melissa Niu
Michelle Luo
Mike Tucker
Mimi Xu
Nancy Chen
Naomi Bolotin
Nate Hardison
Ore Babarinsa
Paul Bowden
Peter Hung
R.J. Aquino
Rob Bowden
Ryan Lee
Sebastian Pierce-Durance
Tim McLaughlin
Tommy MacWilliam
Travis Downs
Tyler Morrison
Vipul Shekhawat
Wesley Chen
Yaniv Yacoby
Yixiao Wang
Yuechen Zhao
Zak Burke
Zamyla Chan

for staff use only

final score out of 83

Multiple Choice.

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

0. (1 point.) How many times can you tear a 1,024-page phonebook into half, each time throwing away one of the halves, before only one page remains?

   a. 10
   b. 32
   c. 512
   d. 1,024

1. (1 point.) How many bits do you need, minimally, in order to represent the (decimal) number 7 in binary?

   a. 3
   b. 4
   c. 7
   d. 8

2. (1 point.) How many *n*-letter keywords are possible when using Vigenère's cipher, assuming a 26-letter alphabet?

   a. n
   b. 26
   c. 26^n
   d. n^{26}

3. (0 points.) What entices you about the iPhone?

   a. It is an iPhone.
   b. It's the best phone.
   c. I can download apps to it.
   d. It's 3G and has the Wi-Fis.

True or False.

For each of the statements below, circle T if the statement is true or F if the statement is false.

4. T F (1 point.) `gdb` is a debugger.
5. T F (1 point.) `GetString` uses `malloc` in order to allocate memory for users' input.
6. T F (1 point.) `make` is a compiler.
7. T F (1 point.) `main` must be the first function defined in a `.c` file.

for staff use only

–

Dang, `clang`. (2 points each.)

As if C code weren't already cryptic enough for the uninitiated, sometimes compilers' error messages are even more cryptic than the code they're compiling! Even so, embedded in each of the error messages from `clang` below is an explanation of something gone wrong. For each message, explain, in no more than three sentences, what the programmer has done wrong and propose how to fix.

8. `implicit declaration of function 'GetString' is invalid in C99`

9. `use of undeclared identifier 'n'`

Rapid Fire. (2 points each.)

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

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

11. Describe two features of `gdb`.

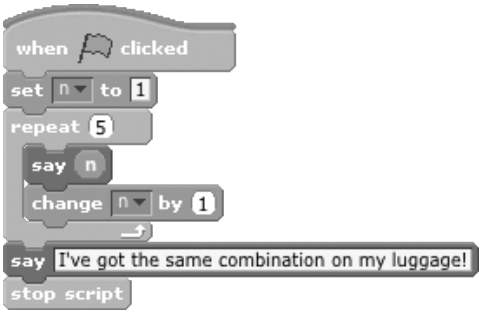
12. When should you call `free`?

for staff use only

–

O hai, Scratch.

13. (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 each call to `printf` should include a trailing `\n`.) And recall that **change n by 1** means to increment **n**.

```
#include <stdio.h>

int main(void)
{
```

This is 50.

14. (2 points.) Convert the binary number below to decimal. Show any work (i.e., any arithmetic).

1 1 0 0 1 0

for staff use only
–

I C what you did there.

Consider the program below, to which line numbers have been added for the sake of discussion.

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("This is Quiz 0.\n");
6      return 0;
7  }
```

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

15. (2 points.) What is inside of `stdio.h` that's of interest to this program?

16. (2 points.) What does `void` signify in line 3?

17. (2 points.) What does returning `0` from `main`, as in line 6, generally signify?

for staff use only

–

18. (2 points.) Consider the program below.

```
#include <stdio.h>

int main(void)
{
    float answer = 1 / 10;
    printf("%.1f\n", answer);
    return 0;
}
```

When executed, this program prints

```
0.0
```

which is not, of course, one tenth! In no more than three sentences, explain why this program thinks that `1` divided by `10`, printed to `1` decimal place, is something other than `0.1`.

19. (2 points.) Suppose that we try changing `1` to `1.0` and `10` to `10.0` and that we also try printing `answer` to `50` decimal places instead of `1`, as in the below.

```
#include <stdio.h>

int main(void)
{
    float answer = 1.0 / 10.0;
    printf("%.50f\n", answer);
    return 0;
}
```

When executed, this program prints

```
0.10000000149011611938476562500000000000000000000000
```

which is also not the same as one tenth! In no more than three sentences, explain why this program thinks that `1.0` divided by `10.0`, printed to `50` decimal places, is something other than `0.10000000000000000000000000000000000000000000000000`.

for staff use only

–

20. (2 points.) Consider the program below.

```
#include <stdio.h>

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

Even though, at first glance, this program appears to induce an infinite loop, it actually stops printing numbers eventually. In no more than three sentences, explain why.

21. (3 points.) Consider the program below.

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

int main(void)
{
    // ask user for an integer
    printf("Give me an integer between 1 and 10: ");
    int n = GetInt();

    // judge user's input
    if (n >= 1 && n <= 3)
        printf("You picked a small number.\n");
    if (n >= 4 && n <= 6)
        printf("You picked a medium number.\n");
    if (n >= 7 && n <= 10)
        printf("You picked a big number.\n");
    if (n < 1 || n > 10)
        printf("You picked an invalid number.\n");

    return 0;
}
```

This program is correct but not well designed because of an inefficiency. In no more than 3 sentences, identify the inefficiency and propose how to eliminate it.

for staff use only

–

22. (2 points.) Consider the program below.

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

int main(void)
{
    do
    {
        printf("Give me an int: ");
        int i = GetInt();
    }
    while (i < 0 || i > 10);
    printf("Thanks, i is %d\n", i);
    return 0;
}
```

It turns out this program has a bug. Indeed, it doesn't even compile with `clang`, even with `-lcs50`. In no more than three sentences, identify the bug and propose how to eliminate it. You're welcome to write on or alongside the code itself.

23. (2 points.) Consider the program below.

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

int main(void)
{
    string s = GetString();
    string t = GetString();
    if (s == t)
        printf("You typed the same thing!\n");
    else
        printf("You typed different things!\n");
    return 0;
}
```

Even when a user types exactly the same English word twice, this program always responds that the inputs are different. In no more than three sentences, explain why.

for staff use only

–

***O*(MG).**

24. (4 points.) Complete the table below by specifying any algorithm (that was covered in a lecture or short) whose running time falls within the specified lower (Ω) and upper (*O*) bounds, just as we've done for you for Linear Search.

	Ω	***O***
	n^2	n^2
	$n \log n$	$n \log n$
	n	n^2
	1	$\log n$
Linear Search	1	n

Stack Cats.

25. (2 points.) Consider the program below.

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

int main(int argc, string argv[])
{
    for (int i = argc - 1; i > 0; i--)
        for (int j = strlen(argv[i]) - 1; j >= 0; j--)
            printf("%c", argv[i][j]);
    return 0;
}
```

If this program is compiled as `a.out` and executed with

```
./a.out stack cats
```

exactly what does it print?

for staff use only

–

Binky Pointer Fun.

Recall the program below, whose lines have been numbered for the sake of discussion, which crashed when Binky tried to execute it.

```
1  #include <stdlib.h>
2
3  int main(void)
4  {
5       //
6       int* x;
7
8       //
9       int* y;
10
11      //
12      x = malloc(sizeof(int));
13
14      //
15      *x = 42;
16
17      //
18      *y = 13;
19
20      //
21      y = x;
22
23      //
24      *y = 13;
25
26      return 0;
27 }
```

26. (1 point.) Which line of code caused this program to crash for Binky?

27. (2 points.) In no more than two sentences, explain why that line caused this program to crash.

28. (6 points.) Atop some of this program's lines are placeholders for comments. Next to each such `//`, explain in precise (but succinct) technical terms what the line immediately below it is doing. Write on the program itself, not below. No need to comment on the line that you already identified as the source of the crash.

for staff use only

–

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

29. (4 points.) Suppose that you can't recall the header file in which `toupper` is declared, and so you have to implement a version of the function yourself. Complete the implementation of `toupper` below in such a way that the function returns `c` in uppercase if `c` is a lowercase (ASCII) letter, else it returns `c` unchanged. Recall that the ASCII value of `'a'` is greater than that of `'A'`. You may not call any functions in your function.

```
char toupper(char c)
{
```

30. (6 points.) Suppose that you can't recall the header file in which `strlen` is declared, and so you have to implement a version of the function yourself. Complete the implementation of `strlen` below in such a way that the function returns the length of `s`. Recall that `string` is just a synonym (defined in the CS50 Library) for `char*`, so you're welcome (but not required) to use pointer arithmetic instead of square brackets. Take care to return `0` if `s` is `NULL` (or if `s` is of length `0`). You may not call any functions in your function.

```
int strlen(string s)
{
```

for staff use only

–

Swapfest.

31. (7 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→
      return 0;
}
```

Suppose that each of the numbered arrows represents a moment in time during this program's execution. Record in the blank boxes below the values of variables 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 variables not in scope have been blacked out.

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

for staff use only

–

32. (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.

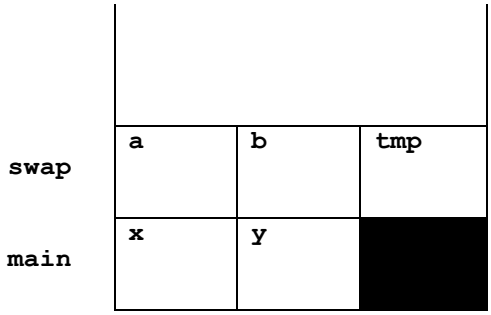
```
void swap(int* a, int* b)
{
     int tmp = *a;
     *a = *b;
     *b = tmp;
  6→
}

int main(void)
{
     int x = 1;
     int y = 2;
     swap(&x, &y);
     return 0;
}
```

Suppose that, as before, the numbered arrow represents a particular moment in time during this program's execution. Moreover, suppose that

- the address of `x` in memory is `10`,
- the address of `y` in memory is `14`,
- the address of `a` in memory is `22`,
- the address of `b` in memory is `26`, and
- the address of `tmp` in memory is `30`.

In other words, if you look at byte `10` in RAM, you'll find `x`, and so forth. 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

–

File I/O.

33. (4 points.) Recall that a CSV file is just a text file with comma-separated values, whereby those commas effectively demarcate columns like those in a spreadsheet. For instance, consider the excerpt below from a CSV file, each of whose lines represents a member of CS50's staff.

```
Malan,David,malan@harvard.edu
Hardison,Nate,nate@cs.harvard.edu
Bowden,Rob,rob@cs.harvard.edu
MacWilliam,Tommy,tmacwilliam@cs.harvard.edu
Chan,Zamyla,zamyla@cs50.net
```

In other words, this CSV file resembles a spreadsheet with 3 columns: one for last names, one for first names, and one for email addresses.

Suppose that a member of CS50's staff can be represented with a `struct` called `staff`, per the below.

```
typedef struct
{
    string last;
    string first;
    string email;
}
staff;
```

Suppose further that each time someone is hired to join CS50's staff, he or she must be added to the CSV file by calling a function called `hire`. Complete the implementation of `hire` below in such a way that it appends a new hire's last name, first name, and email address to `staff.csv` with `fprintf` before calling `fclose`. You may assume that `file` will not be `NULL` and that each member of `s` (i.e., `last`, `first`, and `email`) will be an actual `string` (and not `NULL` or garbage). Recall that `"a"` means to append to `staff.csv` (whereas `"w"` would overwrite it). And realize that you don't need to write many lines of code to complete this function's implementation!

```
void hire(staff s)
{
    FILE* file = fopen("staff.csv", "a");
```

for staff use only

–

Scrap Paper.

Nothing on this page will be examined by the staff unless otherwise directed in the space provided for some question.