Quiz 0

out of 85 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 $(8.5" \times 11")$ 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.

Unless otherwise specified, just a few sentences suffice when asked to explain.

Circle your teaching fellow's name.

Abby Lyons	Emily Houlihan	Larry Guo	Pulak Goyal
Alex Goldberg	Eric Ouyang	Larry Zhang	Rob Bowden
Alex Pong	Eric Twark	Leila Hofer	Robbie Gibson
Ali Monfre	Ezra Zigmond	Linda Song	Robert Krabek
Anita Xu	Filip Bujaroski	Marcus Powers	Ross Rheingans-Yoo
Annaleah Ernst	George Zhang	Maria Zlatkova	Sam Cheng
Arianna Benson	Hannah Blumberg	Mark Grozen-Smith	Samuel Green
Brian Arroyo	Henrique Vaz	Michael Ge	Thomas Lively
Camille Rekhson	Isaac Xia	Michael Patterson	Tim Tamm
Carlos Mendizabal	Jessica Zhu	MJ Richardson	Travis Yeh
Chris Lim	Jon Kim	Neel Mehta	Victor Domene
Collin Styring Jordan Canedy		Nicholas Larus-Stone	Ribeiro dos Santos
Daven Farnham	Jordan Hayashi	Nick Joseph	Walter Martin
	·	·	Winnie Wu
		Patrick Pan	Zack Chauvin
		Peter Chang	Edon Gridavill
Doug Lloyd	Kyle Kwong	Peter Jin	

for staff use onl

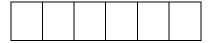
final score out of 85

Short sorts.

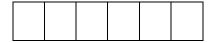
Suppose that the goal at hand is to sort, from left to right, smallest to largest, the array of integers below with some algorithm:

2	5	4	1	6	3
---	---	---	---	---	---

0. (2 points.) Suppose that <u>bubble sort</u> is used to sort the array. Exactly what does the array look like after just <u>one</u> iteration of bubble sort's outer loop?



1. (2 points.) Suppose that <u>selection sort</u> is used to sort the (original) array. Exactly what does the array look like after just <u>one</u> iteration of selection sort's outer loop?



2. (2 points.) Suppose that <u>insertion sort</u> is used to sort the (original) array. Exactly what does the array look like after just one iteration of insertion sort's outer loop?



Small problem.

3. (2 points.) Consider the program below.

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

int main(void)
{
    string s = "cs50";
    int size = sizeof(s);
    printf("%i", size);
}
```

Suppose that this program, when executed, outputs 8, even though "cs50" has only four characters. Why, then, is 8 outputted?

for staff use only

points off

initials

Role reversal.

4. (4 points.) Consider the email below from Rob.

```
From: Rob Bowden
Date: Wed, 14 Oct 2015 04:00:00 -0400
Subject: segfault

okay so i wrote this program that segfaults every time i try to run it with a command-line argument:

#include <stdio.h>

int main(int argc, char* argv[])
{
    // print out all command-line arguments except program's name for (int i = 1; i != 0; i++)
    {
        printf("%s\n", argv[i]);
     }
}

any ideas as to why? lol it's late. thx robz
```

Write back to Rob, explaining why his code is segfaulting and how he can fix it.

for staff use only

points off

5. (4 points.) Consider the email below from Jason.

From: Jason Hirschhorn
Date: Wed, 14 Oct 2015 12:59:59 -0400
Subject: Sorting

Hey, so I know you mentioned that the pseudocode I wrote for bubble sort is inefficient, but I don't see what the problem is. $^-_(\mathcal{U})_-/^-$ Mind letting me know what about my implementation makes it less efficient than it could be? I do want to stick with bubble sort, though. I just want to make it a bit better.

```
for i = 0 to i = n-2
    for j = 0 to j = n-2
        if array[j] > array[j+1]
            swap array[j] with array[j+1]
```

Thanks! <3

Write back to Jason, explaining why his implementation of bubble sort isn't as efficient as it could be and how he could improve it (without switching to some other algorithm altogether).

for staff use only

points off

Dollar store.

Consider the program below, which is intended to print \$0.00 through \$0.99, line by line.

```
#include <stdio.h>
int main(void)
{
    for (float f = 0.00; f != 1.00; f += 0.01)
        {
        printf("$%.2f\n", f);
     }
}
```

6. (2 points.) Unfortunately, when executed, this program loops infinitely, printing not only \$0.00 through \$0.99 but also \$1.00, \$1.01, and beyond, line by line. Explain why it does not, in fact, stop after \$0.99.

7. (2 points.) In the space below, complete the re-implementation of the function in such a way that it does, in fact, print only \$0.00 through \$0.99, line by line.

```
#include <stdio.h>
int main(void)
{
```

for staff use only

points off

CS50 Library 2.0.

8. (6 points.) Suppose that we'd like to add to the CS50 Library a function that copies a string. Complete the implementation of CopyString, below, in such a way that the function returns a character-for-character copy of s, with the copy's characters in their own block of dynamically allocated memory, terminated with '\0'. Return the copy (i.e., the address of the copy's first byte) unless some error occurs, in which case return NULL. Assume that s will not be NULL.

```
char* CopyString(char* s)
{
```

for staff use only

9. (4 points.) Suppose that we'd like to add to the CS50 Library a function that gets two integers from a user. Even though a function in C can return no more than one value, a function can take multiple arguments. If those arguments are pointers, a function can effectively return multiple values by changing the values to which those pointers point!

Complete the implementation of <code>GetInts</code> below in such a way that the function gets two integers from a user, storing the first at the address in a and the second at the address in b. Assume that neither a nor b will be <code>NULL</code>. And you are welcome to call <code>GetInt</code>.

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

void GetInts(int* a, int* b);

int main(void)
{
    int x, y;
    GetInts(&x, &y);
    printf("x is %i, y is %i\n", x, y);
}

void GetInts(int* a, int* b)
{
```

for staff use only

points off

WTf.

Consider the program below.

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

char f(char c)
{
    if (isalpha(c))
    {
       return c & 0xdf;
    }
    else
    {
       return c;
    }
}

int main(void)
{
    printf("%c\n", f(GetChar()));
}
```

10. (2 points.) Suppose that a user inputs a followed by Enter when prompted by GetChar. What would this program print? Recall that the ASCII value of a is 97 in decimal.

11. (2 points.) Suppose that a user inputs A followed by Enter when prompted by GetChar. What would this program print? Recall that the ASCII value of A is 65 in decimal.

12. (1 point.) What would be a more appropriate, descriptive name for f that makes clear its behavior? Note that 'a' - 'A' is 32, as is 'z' - 'Z'.

for staff use only

points off

initials

Cost-benefit analysis.

13. (9 points.) For each of the design decisions below, **X** instead of Y, cite one benefit and one cost of using **X** instead of Y.

	benefit	cost
linked list instead of array		
merge sort instead of insertion sort		
binary search instead of linear search		

for staff use only			
points off			
initials			

Searching and sorted.

14. (7 points.) Consider the incomplete implementation of search, below, to which line numbers have been added for the sake of discussion. Complete the implementation by filling the blanks in lines 2, 14 and 18.

```
1 /**
  * Searches sorted array of given length for value in O( )
  * time.
   * /
5 bool search(int value, int* array, int length)
6 {
7
      if (length == 0)
8
9
         return false;
10
      int middle = length / 2;
11
      if (value < array[middle])</pre>
12
13
         return search(_____, _____, _____);
14
15
     }
16
      else if (array[middle] < value)</pre>
17
         return search(_____, _____, _____);
18
19
      }
20
     else
21
     {
22
        return true;
23
      }
24 }
```

15. (4 points.) Complete the implementation of sorted, below, in such a way that the function returns true if an array of some length is already sorted from smallest to largest, else it returns false. Consider an empty array to be sorted. Assume that length will not be negative.

```
bool sorted(int array[], int length)
{
```

for staff use only

points off

Jack is back.

16. (2 points.) Recall that Jack wears the same clothes every day. When he removes his clothes, including his socks, he immediately washes them and then puts them away in a box. Come the next morning, up he does hop. He goes to the box and gets his clothes off the top. Suffice it to say Jack keeps his clothes in a stack! Explain why Jack's friend, Lou, instead recommends a queue.

17. (4 points.) Suppose that a <u>stack</u> for integers is defined per the below, wherein numbers is an array for the stack's integers, CAPACITY (a constant) is the stack's capacity (i.e., maximal size), and size is the stack's current size.

```
typedef struct
{
    int numbers[CAPACITY];
    int size;
}
stack;
```

Complete the implementation of <code>push</code> below in such a way that it pushes <code>n</code> onto a stack, <code>s</code>, if <code>s</code> isn't already full. Assume that <code>s</code> has been declared globally. Consider <code>s</code> full only if its <code>size</code> equals <code>CAPACITY</code>. No need to return a value, even if <code>s</code> is full. Your implementation should operate in constant time.

```
void push(int n)
{
```

for staff use only

points off

18. (4 points.) Suppose that a <u>queue</u> for integers is defined per the below, wherein numbers is an array for the queue's integers, CAPACITY (a constant) is the queue's capacity (i.e., maximal size), size is the queue's current size, and front is the index of the integer at the front of the queue.

```
typedef struct
{
    int front;
    int numbers[CAPACITY];
    int size;
}
queue;
```

Complete the implementation of <code>enqueue</code> below in such a way that it inserts <code>n</code> into a queue, <code>q</code>, if <code>q</code> isn't already full. Assume that <code>q</code> has been declared globally. Consider <code>q</code> full only if its <code>size</code> equals <code>CAPACITY</code>. No need to return a value, even if <code>q</code> is full. Your implementation should operate in constant time.

```
void enqueue(int n)
{
```

for staff use only

points off

(Re)curses.

19. (2 points.) Consider the program below. Recall that an unsigned int is just a non-negative integer.

```
#include <stdio.h>
void f(unsigned int i)
{
    if (i / 10 != 0)
        {
        f(i / 10);
        }
        printf("%i", i % 10);
}
int main(void)
{
    f(123);
}
```

Exactly what does this program print when executed?

Bold claims. (2 points each.)

For each of the claims below, explain why it is true or not true.

- 20. "An algorithm that runs in $O(\log n)$ time is always faster than an algorithm that runs in $O(n^2)$ time."
- 21. "Linear search is in $\Omega(1)$."
- 22. "Selection sort is in $\Omega(n^2)$."

for staff use only

points off

OIC.

23. (4 points.) You may have noticed that products in stores tend to have Universal Product Codes (UPCs), numbers (represented with barcodes) that uniquely identify them and allow cashiers to scan them at checkout. Consider, for instance, the below UPC from a 5-pound bag of Smarties (i.e., America's Favorite Candy Roll). Ignore the gaps between the UPC's numbers; the barcode represents 011206005003.



It turns out that if the UPC for some package of Smarties starts with 011206, then the candy was manufactured in a facility that's gluten-free. Complete the implementation of gf below in such a way that the function returns true if and only if code, inputted by a cashier as a string (e.g., "011206005003"), starts with 011206. Assume that code will be entirely numeric and of length 12.

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

bool gf(string code);

int main(void)
{
    if (gf(GetString()))
    {
        printf("gluten-free\n");
    }
    else
    {
        printf("not gluten-free\n");
    }
}

bool gf(string code)
{
```

for staff use only

points off

Similar but different. (2 points each.)

Similar but unrerent. (2 points each.)			
For	each of the pairs below, X versus Y, distinguish X from Y, making clear the meaning of each		
24.	local variable versus global variable		
25.	NULL versus '\0'		
26.	stack versus heap		
27.	#include <cs50.h> versus -lcs50</cs50.h>		

for staff use only

points off

initials