Quiz 0

out of 63 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.

Circle your teaching fellow's name.

Alex Chang
Alex Hugon
Ana Roda
Ashin Shah
Ben Massenburg
Dan Armendariz
Dev Purkayastha
Doug Lloyd
Ellen Farber
Ferris Zhang
Fil Zembowicz
Gabrielle Ehrlich
Idriss Fofana
Jeff Solnet
Jeremy Cushman
John Tristan

Josh Bolduc
Katie Fifer
Ken Parreno
Lakshmi Parthasarathy
Lauren Carvalho
Lee Evangelakos
Luis Duarte
Marta Bralic
Matt Chartier

Melissa Niu
Michael Chen
Michael Oberst
Mike Teodorescu
Punit Shah
Rei Diaz
Rob Bowden
Rose Cao
Scott Crouch
Sophie Chang
Steve Tricanowicz
Tian Feng
Tommy MacWilliam
Wellie Chao
Willie Yao
Yuhki Yamashita

for staff use only

final score out of 63

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. (0 points.) Whose head can be found in the ceiling of the Lounge?

    a. Ceiling Cat's
    b. Happycat's
    c. Lolrus's
    d. Ken Parreno's

1. (1 point.) How many asterisks (`*`) will the code below print?

```
for (int i = 0; i > 0; i++)
    printf("*");
```

    a. 0
    b. $2^{31} - 1$
    c. $2^{32} - 1$
    d. infinity

2. (1 point.) How many asterisks (`*`) will the code below print if `x` is initially `0`?

```
while (x > 0)
{
    printf("*");
    x--;
}
```

    a. –1
    b. 0
    c. 1
    d. infinity

3. (1 point.) How many asterisks (`*`) will the code below print if `x` is initially `0`?

```
do
{
    printf("*");
    x--;
}
while (x > 0);
```

    a. –1
    b. 0
    c. 1
    d. infinity

for staff use only

–

4. (1 point.) How many times must you tear a 1,024-page phonebook in half in order to whittle it down to a single page?

   a. 8
   b. 10
   c. 32
   d. 512

5. (1 point.) The ASCII value of `A`, in binary, is

   a. 01000010.
   b. 10000010.
   c. 00100001.
   d. 01000001.

True or False.

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

6.	T	F	(0 points.) Gur nafjre vf G.
7.	T	F	(1 point.) In the worst case, the running time of Bubble Sort is in $O(n^2)$.
8.	T	F	(1 point.) `malloc` allocates memory on the stack.
9.	T	F	(1 point.) A function that calls itself is said to be recursive.
10.	T	F	(1 point.) I deserve a free point!

Itching for Week 0?

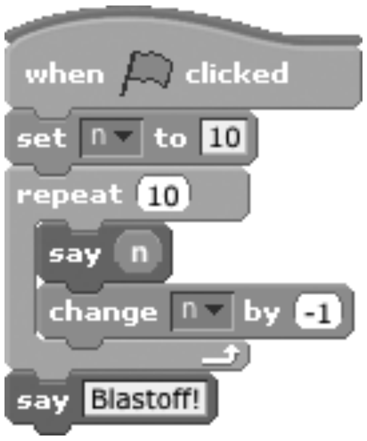
11. (1 point.) Perform the below calculation in binary. Show any work (*i.e.*, any 1s carried).

```
    0 0 0 1 1 0 0 1
+   0 0 0 1 1 0 0 1
-------------------
```

for staff use only

–

12. (2 points.) Consider the Scratch script below.



Complete the translation of this Scratch script to a C program below. Assume that **say** translates to `printf`; no need for newlines (`\n`). Rest assured that multiple translations are possible; you're welcome to introduce variables besides `n` and/or cross out our declaration of `n` altogether so long as your C program's output is equivalent to the Scratch script's.

```
#include <stdio.h>

int
main(void)
{
    int n = 10;
```

for staff use only

–

Role Reversal.

13. (2 points.) Consider Erfan's program, below, whose lines have been numbered for the sake of discussion.

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int
5  main(void)
6  {
7      int n = GetInt();
8      if (n == 0);
9          printf("zero");
10 }
```

According to Erfan, this program is supposed to print `zero` only if a user inputs `0` when prompted by `GetInt`. But there's a bug! In two sentences, state what it actually does and propose how to fix it.

14. (2 points.) Consider Kerry's program, below, whose lines have been numbered for the sake of discussion.[1]

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int
5  main(void)
6  {
7      int n = GetInt();
8      if (n = 1)
9          printf("one");
10 }
```

According to Kerry, this program is supposed to print `one` only if a user inputs `1` when prompted by `GetInt`. But there's a bug! In two sentences, state what it actually does and propose how to fix it.

for staff use only

–

[1] Assume that this program is compiled without `-Wall`.

Big OMG.

15. (4 points.) For each algorithm below, specify an upper (*O*) and lower (Ω) bound on its running time. As examples, we've filled in two boxes for you. Assume that the linked lists and arrays in question are all of length *n*. Do not assume that a data structure is sorted or unsorted unless told.

Algorithm	***O***	**Ω**
sorting an array with Merge Sort		
sorting an array with Selection Sort		
inserting into a **sorted** linked list	*n*	
searching a **sorted** array with Binary Search		
searching a **sorted** linked list with Linear Search		1

Bit Wise.

16. (3 points.) Complete the table below, specifying in the right-hand column the number of bits (not bytes) used to represent a variable of each type on a 32-bit architecture like `cloud.cs50.net`. As examples, we've filled in two boxes for you.

type	**bits**
`char`	
`char *`	
`double`	
`double *`	
`int`	32
`int *`	32
`unsigned int`	
`long long`	

for staff use only

–

Temp Variables.

17. (2 points.) Consider the program below, whose lines have been numbered for the sake of discussion. This program is meant to convert temperatures in Celsius (C) to Fahrenheit (F), the arithmetic formula for which is F = (9 / 5) × C + 32.

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int
5  main(void)
6  {
7      printf("Temperature in C: ");
8      float c = GetFloat();
9      float f = (9 / 5) * c + 32;
10     printf("%.1f C = %.1f F", c, f);
11 }
```

Even though this program converts the freezing temperature of water, 0° C, to 32° F just fine, its answers are several degrees off for most every other conversion. For instance, it thinks the boiling temperature of water, 100° C, is 132° F, even though it should be 212° F! In two sentences, explain why this program is erring and propose how to fix it.

This is CS50.

18. (2 points.) Recall that, to use any of the CS50 Library's functions, you not only need to put

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

atop your code, you also need to pass the `-lcs50` flag to `gcc` when compiling, which might feel redundant. But this `#include` and this `-lcs50` play different roles in the process of compilation. In two sentences, what role does each play?

for staff use only

–

Quickies.

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

19. (1 point.) What's pseudocode?

20. (2 points.) What's the difference between `\n` and `\r`?

21. (1 point.) Even though 2^{32} is 4,294,967,296, the largest value that a 32-bit `int` can represent is only 2,147,483,647. Why?

22. (2 points.) Propose two ways in which you can trigger a segfault.

for staff use only

–

User Error.

It turns out that the CS50 Library implements `GetFloat` much like it does `GetInt`. In particular, it calls `sscanf` in order to parse `line`, per the below excerpt from `cs50.c`, whose lines have been numbered for the sake of discussion.

```
1  float
2  GetFloat(void)
3  {
4      while (true)
5      {
6          string line = GetString();
7          if (line == NULL)
8              return FLT_MAX;
9          char c; float f;
10         if (sscanf(line, " %f %c", &f, &c) == 1)
11         {
12             free(line);
13             return f;
14         }
15         else
16         {
17             free(line);
18             printf("Retry: ");
19         }
20     }
21 }
```

23. (1 point.) Suppose that `sscanf` returns `0` in line 10. In a sentence, explain what the user must have done when prompted for a floating-point value.

24. (1 point.) Suppose that `sscanf` returns `2` in line 10. In a sentence, explain what the user must have done when prompted for a floating-point value.

for staff use only

–

Programmer Error.

25. (2 points.) Consider the below, a function intended to capitalize its argument, whose lines have been numbered for the sake of discussion.

```
1  void
2  capitalize(char c)
3  {
4      if (islower(c))
5          c = toupper(c);
6  }
```

Even though the function compiles just fine (assuming you include any necessary header files), the function nonetheless appears to be broken, because code like

```
char c = 'a';
printf("%c", c);
capitalize(c);
printf("%c", c);
```

prints a lowercase `a` both times. In just a few sentences, explain why `capitalize` does not actually capitalize its argument as intended and propose how to fix it.

for staff use only

–

Bottle(s).

Consider the program below, whose lines have been numbered for the sake of discussion.

```
1   #include <cs50.h>
2   #include <stdio.h>
3
4   int
5   main(void)
6   {
7       int n;
8       do
9       {
10          n = GetInt();
11      }
12      while (n < 0);
13      if (n == 1)
14      {
15          char *s = "bottle";
16      }
17      else
18      {
19          char *s = "bottles";
20      }
21      printf("%s\n", s);
22  }
```

Unfortunately, when you try to compile this program, `gcc` reports these errors:

```
program.c: In function 'main':
program.c:15: error: unused variable 's'
program.c:19: error: unused variable 's'
program.c:21: error: 's' undeclared (first use in this function)
```

26. (2 points.) In no more than three sentences, identify and explain the mistake(s) in the program that triggered these errors.

27. (1 point.) In a sentence or two, explain what this program is presumably intended to do, those errors aside.

for staff use only

–

Pointer Fun without Binky.

28. (2 points.) Consider the program below, whose lines have been numbered for the sake of discussion. Assume that `GetString` never returns `NULL`.

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int
5  main(void)
6  {
7      char *s1 = GetString();
8      char *s2 = GetString();
9      if (s1 == s2)
10         printf("same");
11     else
12         printf("different");
13 }
```

Even if I type exactly the same thing twice when prompted by `GetString`, this program always reports that my inputs are different. In no more than three sentences, explain why that is.

29. (3 points.) Consider the code below.

```
int i = 1;
int j = 2;
int k = 7;
int *p = &j;
*p += 1;
```

Complete the table below, specifying in the right-hand column exactly what would be printed by each call to `printf`, assuming `printf` is called after <u>all</u> of the lines above have been executed. As an example, we've filled in the first box for you.

statement	**what would be printed**
`printf("%d", i);`	1
`printf("%d", j);`	
`printf("%d", *p);`	
`printf("%d", k);`	

for staff use only

–

30. (3 points.) Recall that `strlen` is a function that, according to its man page, "calculates the length of the string `s`, not including the terminating `'\0'` character." Unfortunately, you seem to have misplaced `string.h`, where `strlen` is usually declared, and so you must rewrite this function yourself. Complete the implementation of `strlen` below.[2] If `s` happens to be `NULL`, your implementation must return `0`.

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

(Proto)typical Alex.

31. (3 points.) Below are the names of some popular C functions along with summaries thereof.[3]

function	**summary**
`isalnum`	true if a character is alphanumeric
`isalpha`	true if a character is alphabetic
`isdigit`	true if a character is a digit
`islower`	true if a character is a lowercase letter
`isupper`	true if a character is an uppercase character
`tolower`	converts a character to lowercase
`toupper`	converts a character to uppercase

Unfortunately, ~~Alex Hugon~~ a certain someone didn't realize that these functions already existed, and so he re-implemented some (but not all) of them himself. Embarrassed once he discovered he'd reinvented several wheels, he quickly redacted (*i.e.*, blacked-out) his functions' prototypes so that no one would know.

for staff use only

–

[2] For simplicity, we have declared `strlen`'s return type as `int` (instead of `size_t`), and we have not declared `strlen`'s parameter, `s`, as `const`.
[3] Adapted from `cppreference.com`.

Complete the table below by inferring and recording in the left-hand column a prototype for that row's function; no need for a semicolon. As an example, we've filled in the first box for you. Rest assured that multiple prototypes might be possible for each function. Recall that a prototype includes a function's return type, name, and any parameters. And realize that only four of the seven functions above are implemented below.

prototype	**function**

`bool`
`isalnum(char c)`
	[redacted]
	`{`
	`    if (isalpha(c) || isdigit(c))`
	`        return true;`
	`    else`
	`        return false;`
	`}`

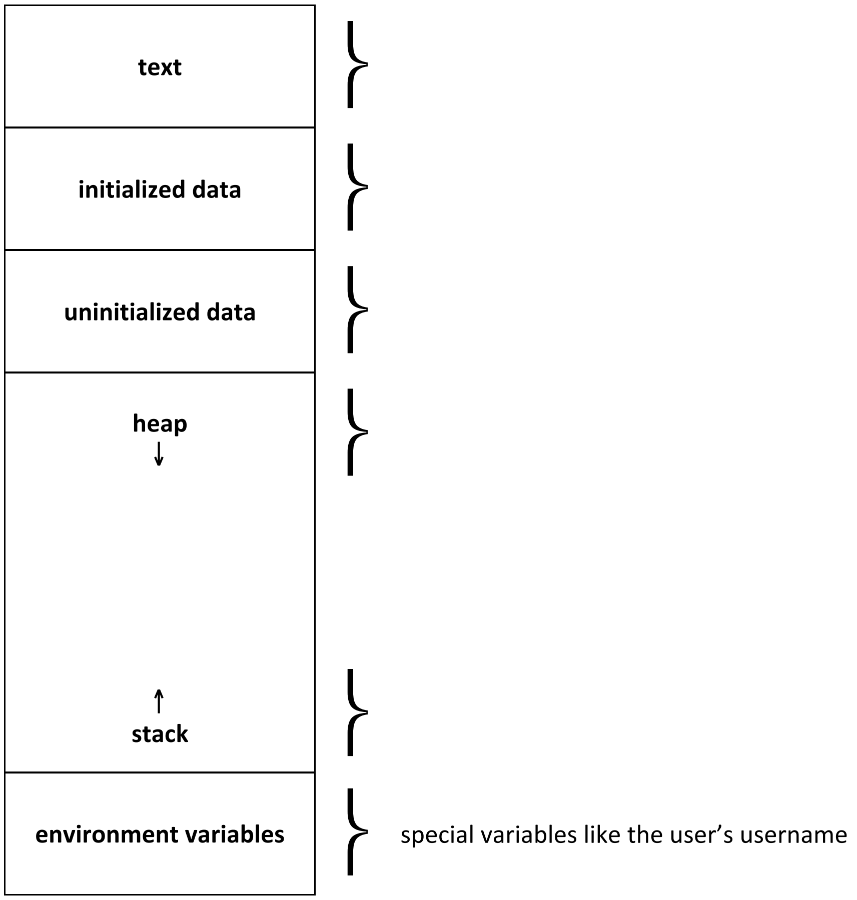
	[redacted]
	`{`
	`    if ('A' <= c && c <= 'Z')`
	`        return c - 'A' + 'a';`
	`    else`
	`        return c;`
	`}`

	[redacted]
	`{`
	`    if ('A' <= c && c <= 'Z')`
	`        return true;`
	`    else`
	`        return false;`
	`}`

	[redacted]
	`{`
	`    if ('0' <= c && c <= '9')`
	`        return true;`
	`    else`
	`        return false;`
	`}`

Ah, memories.

32. (5 points.) Recall that a program's memory space can be depicted as per the figure below. Next to each segment of memory, cite something that gets stored in that segment, just as we've done for you for the bottommost segment.



for staff use only

–

Argh.

33. (3 points.) Consider the program below, whose lines have been numbered for the sake of discussion.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int
5  main(int argc, char *argv[])
6  {
7      // TODO
8  }
```

Assume that this program is always executed as follows.

```
./a.out foo bar baz
```

Suppose that line 7 is replaced with each of the statements below, one at a time, and the program is recompiled each time. Complete the table below by recording in the right-hand column what would be printed when that row's statement is executed. As an example, we've filled in the first box for you.

statement	**what would be printed**
`printf("%s", argv[0]);`	`./a.out`
`printf("%c", argv[0][1]);`	
`printf("%c", argv[1][0]);`	
`printf("%s", &argv[1][1]);`	
`printf("%d", strlen(argv[0]));`	
`printf("%d", strlen(&argv[0][2]));`	
`printf("%d", argc);`	

for staff use only

–

From Floor to Ceiling.

34. (2 points.) Recall that the "floor" of some real number, *x*, usually written $\lfloor x \rfloor$, is the largest integer less than or equal to *x*. For instance, $\lfloor 50.0 \rfloor = 50$, $\lfloor 50.1 \rfloor = 50$, $\lfloor 50.5 \rfloor = 50$, $\lfloor 50.9 \rfloor = 50$, and $\lfloor 50.999 \rfloor = 50$. Complete the implementation of `floor` below. You may assume that `x` will be non-negative. You may not use any functions declared in `math.h`.

```
int
floor(float x)
{
```

35. (2 points.) Recall that the "ceiling" of some real number, *x*, usually written $\lceil x \rceil$, is the smallest integer greater than or equal to *x*. For instance, $\lceil 49.001 \rceil = 50$, $\lceil 49.1 \rceil = 50$, $\lceil 49.5 \rceil = 50$, $\lceil 49.9 \rceil = 50$, and $\lceil 50.0 \rceil = 50$. Complete the implementation of `ceil` below. You may assume that `x` will be non-negative. You may not use any functions declared in `math.h`, but you may call your own version of `floor`.

```
int
ceil(float x)
{
```

for staff use only

–

PC LOAD LETTER.

This one's just for fun. Go check your answers to real questions before tackling this one!

36. (0 points.) Consider the excerpt below from the script of *Office Space*.

```
PETER
Yeah.  I, I, I...Listen, that virus you're always talking about.  The one that,
that could rip off the company for a bunch of money...

MICHAEL
Yeah?  What about it?

PETER
Well, how does it work?

MICHAEL
It's pretty brilliant.  What it does is where there's a bank transaction, and the
interests are computed in the thousands a day in fractions of a cent, which it
usually rounds off.  What this does is it takes those remainders and puts it into
your account.

PETER
This sounds familiar.

MICHAEL
Yeah.  They did this in Superman III.

PETER
Yeah.  What a good movie.
```

Complete the implementation of Michael Bolton's idea below. Specifically, this function, `steal`, should round its input, `interest`, to the nearest hundredth, and then return the difference between that rounded value and the original. For instance, if `interest` is `1.23456789`, then `steal` should return `0.00456789`. You may disregard the inherent imprecision of floating-point values; movies aren't real.

```
double
steal(double interest)
{
```

Scrap Paper.

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