This is **CS50**.
Harvard University Fall 2014

Quiz 0

out of 75 points

Print your name on the line below.

Katarina Rossi

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. (NONE!)

Aidi Zhang	Emily Houlihan	Kevin Mu	Rob Bowden
Alex Pong	Eric Ouyang	Lily Tsai	Robbie Gibson
Allison Buchholtz-Au	Frederick Widjaja	Luciano Arango	Saheela Ibraheem
Ankit Gupta	Gabriel Guimaraes	Luis Perez	Sam Green
Armaghan Behlum	Gal Koplewitz	Lukas Missik	Stephen Turban
Arvind Narayanan	George Lok	Marcus Powers	Theo Levine
Belinda Zeng	Hannah Blumberg	Mehdi Aourir	Tiffany Wu
Camille Rekhson	Ian Nightingale	Michael Patterson	Tim McLaughlin
Chris Lim	Jackson Steinkamp	Michelle Danoff	Tomas Reimers
Cynthia Meng	Jason Hirschhorn	Nicholas Larus-Stone	Tony Ho
Dan Bradley	Jonathan Miller	Nick Joseph	Vipul Shekhawat
Daven Farnham	Jordan Canedy	Nick Mahlangu	Wellie Chao
David Kaufman	Joshua Meier	Rei Otake	Wesley Chen
Doug Lloyd	Keenan Monks	Rhed Shi	Willy Xiao
			Winnie Wu

for staff use only

final score out of 75

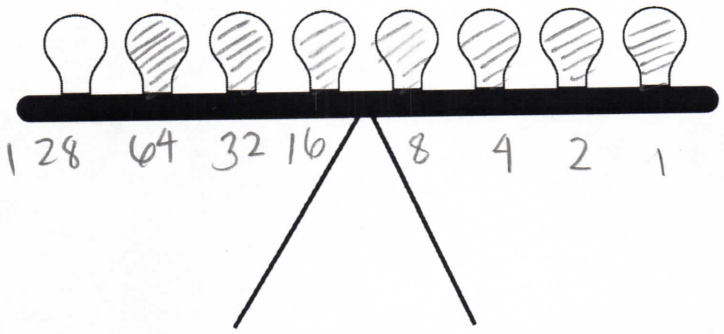
73

= 97% :)

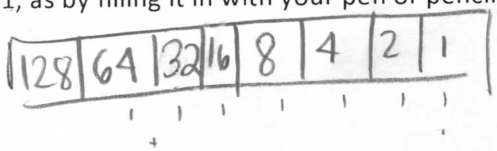
Binary Bulbs.

0. (2 points.) Consider the "binary bulbs" below that collectively represent a non-negative integer. Each of the bulbs represents a bit: a bulb that's off is a 0, and a bulb that's on is a 1. All of the bulbs are currently off. And the rightmost bulb represents the least significant bit (i.e., ones' place).

+2



Suppose that you want these bulbs to represent the decimal integer 127. Turn on the requisite bulbs by marking any bulb that should become a 1, as by filling it in with your pen or pencil.



Bit-Sized Questions.

1. (1 point.) With 3 bits, you can represent 8 distinct values. Why, then, is 7 the largest non-negative decimal integer that you can represent with 3 bits?

+1 Because you start with 0. 0-7 is 8 distinct values.

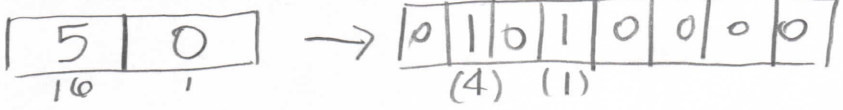
2. (1 point.) With *n* bits, how many distinct values can you represent?

+1 2^n.

This is CS80.

3. (1 point.) What's 0x50 in binary?

+1 01010000.



4. (1 point.) What's 0x50 in decimal?

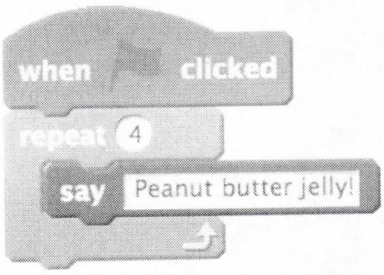
+1 64 + 16 = 80.

+6

for staff use only
- 0

Looping back to Scratch.

5. (4 points.) Consider the Scratch script below.



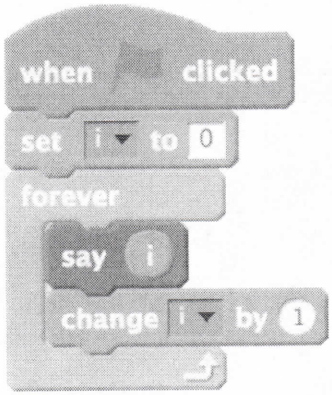
In the space below, complete the translation of this Scratch script to a C program in such a way that its output is equivalent. (Your program's structure needn't be equivalent.) Assume that **say** is `printf`. Output `\n` after you output each line of text.

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

int main(void)
{
    for (int i=0; i < 4; i++)
    {
        printf ("Peanut butter jelly! \n");
    }

}
```

for staff use only

\- 0

6. (4 points.) Consider the Scratch script below.



In the space below, complete the translation of this Scratch script to a C program in such a way that its output is equivalent. (Your program's structure needn't be equivalent.) Assume that **i** is an `int`, that **say** is `printf`, and that to **change** means to increment. Output `\n` after you output each `int`. Don't worry about integer overflow.

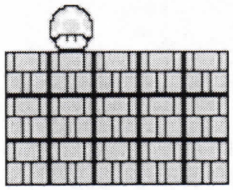
```
#include <cs50.h>
#include <stdio.h>

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

for staff use only

\- 0

Itsa Mario again.

7. (4 points.) Toward the beginning of World 3-7 in Nintendo's Super Mario Brothers 3, Mario encounters a grid of bricks, 5 bricks wide by 3 bricks high, inside of which is a free life! Below is a screenshot.



Were this same grid to be printed (sans mushroom) as "ASCII art" using hashes (#) for bricks, it might resemble the below:

```
#####
#####
#####
```

Complete the implementation of `PrintGrid` in the program below in such a way that it outputs (via `printf`) a grid of hashes per the specified `width` and `height` (without a mushroom).

```
#include <stdio.h>

void PrintGrid(int width, int height);

int main(void)
{
    PrintGrid(5, 3);
}

void PrintGrid(int width, int height)
{
    // print "rows"
    for (int i=0; i<height; i++)
    {
        // print "columns"
        for (int j=0; j<width, j++)
        {
            printf("# ");
        }
        printf("\n");
    }
}
```

8. (2 points.) A function like `PrintGrid` is said to have a side effect but not a return value. Explain the distinction.

* that persists beyond the function's call

A side effect means that the function has some effect* such as printing to stdout in this case, that is in addition to returning a value of the type that function is defined as returning. THIS function has a return type of "void", meaning nothing. NO return value.

5 < 15

it returns

for staff use only

\- |

hello, C.

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

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      string s = GetString();
7      printf("hello, %s\n", s);
8  }
```

9. (1 point.) Explain why line 1 is present.

Because the function GetString is defined in the cs50 library.

ü as is the typedef string for char*, dumb that I didn't notice that ü

10. (1 point.) Explain why line 2 is present.

Printf is defined in the standard input/output library whose Header file is stdio.h

11. (1 point.) Explain what `void` signifies in line 4.

The main function does not take any command line arguments.

12. (2 points.) Explain, with respect to memory, exactly what `GetString` in line 6 returns.

GetString returns an address of a dynamically allocated consecutive set of Bytes sizeof(s) somewhere on the Heap — a char* pointer.

for staff use only
- 1

Real Problems.

13. (2 points.) Consider the program below.

```
#include <stdio.h>

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

When compiled and executed, this program outputs

```
0.0
```

even though `1 / 10` is surely `0.1`! Why is this program outputting `0.0`?

Because 1 and 10 are both ints, the answer of 1/10 is truncated - everything after the decimal place is discarded. 1.0 / 10.0 or (float) 1 / (float) 10 would produce 0.1.

14. (2 points.) Consider the program below.

```
#include <stdio.h>

int main(void)
{
    printf("%.28f\n", 0.1);
}
```

When compiled and executed, this program outputs

```
0.1000000000000000055511151231
```

even though `0.1` is surely `0.1000000000000000000000000000`! Why is this program outputting `0.1000000000000000055511151231`?

Because computers can only store a finite amount of bits, they are inherently imprecise in storing floats (this is called floating point imprecision). The value it is printing is the closest value it can store to, 0.1000000000... the decimal value

for staff use only

- 0

One Plus One.

15. (1 point.) Consider the code below.

```
printf("%i\n", 1 + 1);
```

Assuming it's compiled and executed (as part of some program), exactly what does this line print?

2 (and a new Line)

16. (2 points.) Consider the code below.

```
printf("%i\n", '1' + '1');
```

Assuming it's compiled and executed (as part of some program), why does this line not print the same?

THIS prints 98, 49 + 49, Because '1' gives the value of the ASCII character 1, which is 49.

An Odd Question.

17. (3 points.) Complete the implementation of `odd` below in such a way that the function returns `true` if `n` is odd and `false` if `n` is even.

```
bool odd(unsigned int n)
{
    if (n % 2 != 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

for staff use only

- 0

Curses, recursive!

Consider the recursive function below.

```
int f(int n)
{
    if (n <= 1)
    {
        return 1;
    }
    else
    {
        return n * f(n - 1);
    }
}
```

18. (1 point.) What does this function compute? Express your answer succinctly in English or with a mathematical formula.

n! - n factorial

19. (4 points.) Complete the re-implementation of `f`, below, in such a way that the function is no longer recursive but iterative instead. Don't worry about integer overflow.

n = 5! = 5 x 4 x 3 x 2 x 1

```
int f(int n)
{
    int total = 1;
    while (n > 0)
    {
        total *= n;
        n --;
    }
    return total;
}
```

scrap

f(5)
total = 1
5 > 0
total = 1 * 5 = 5
n = 4
4 > 0
total = 5 * 4
n = 3
3 > 0
total = ((5 * 4) * 3) * 2) * 1
n = 2
n = 1
n = 0
return 5 * 4 * 3 * 2 * 1

for staff use only

\- 0

***O*(MG).**

20. (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 with stupid sort. Assume that the input to each algorithm is an array of size *n*. Take care not to specify any algorithm more than once.

	Ω	*O*
Linear Search	1	n
merge sort	$n \log n$	$n \log n$
BUBBLE & insertion sort	n	n^2
Selection sort	n^2	n^2
stupid sort	n	∞

Overflowing with Questions.

21. (2 points.) What's integer overflow?

Integer overflow occurs when you increment a value 1 more than the maximum value an int can store. For example, an unsigned int can store 2^{32} as its max value (11111 → so on). plus 1, you get 0. as you are carrying all the 1's until you run out of storage & the final 1 is lost.

22. (2 points.) What's a buffer overflow?

Buffer overflow is when you write some value into a buffer that is larger than the size of the buffer – so you start overwriting beyond the memory you should have access to. This can cause a seg. fault or can overwrite important information in the parent routine's stack such as the return address.

for staff use only

- 0

Forgetful.

23. (6 points.) Suppose that you've forgotten (as seems to happen annually) in which header file `strlen` is declared, and so you must re-implement it yourself. Complete the implementation of `strlen` below in such a way that it returns the length of `s`. Assume that `s` will not be `NULL`. And assume that `s` will be terminated with `\0`, which does not count as part of its length. For instance, the length of `hello` would be `5`. Do not worry about integer overflow.

```
int strlen(char* s)
{
    int len = 0;
    while ( s[len] != '\0')
    {
        len ++;
    }
    return len;
}
```

for staff use only

\- 0

s = hello \0
0 1 2 3 4 5

len = 0

h s[0] != \0 → len = 1
e s[1] != \0 → len = 2
l s[2] != \0 → len = 3
l s[3] != \0 → len = 4
o s[4] != \0 → len = 5
\0 s[5] return 5

Having said that...

Consider the remarks below, each of which sounds like an advantage but is not without an underlying disadvantage too. Complete each of the remarks, making clear the price paid (i.e., tradeoff) for the advantage.

24. (2 points.) Merge sort tends to be faster than bubble sort. Having said that...

Merge sort uses more memory.

25. (2 points.) A linked list can grow and shrink to fit as many elements as needed. Having said that...

You cannot jump to any specific element like you can indexing into an array — you give up random access.

26. (2 points.) Binary search tends to be faster than linear search. Having said that...

Binary search requires a sorted array to function (Linear search does not).

for staff use only
- 0

Swapping Stories.

27. (5 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. 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`. Assume that the address of `x` would be `0x10` and that the address of `y` would be `0x14`. Record in the blank boxes below the values of this program's variables at each moment in time, whether in scope or not. Boxes for variables not (still) on the stack have been blacked out.

	x	y	a (&y)	b (&x)	tmp
1 →	1				
2 →	1	2			
3 →	1	2	0x10	0x14	
4 →	1	2	0x10	0x14	1
5 →	2	2	0x10	0x14	1
6 →	2	1	0x10	0x14	1
7 →	2	1			

for staff use only

– 0

Role Reversal. (2 points each.)

Suppose that you encounter the error messages below during office hours next year as a CA or TF! Advise how to fix each of the errors.

28. `undefined reference to `GetString'`

It seems like you forgot to link the CS50 library when running clang (w/ -lcs50).

29. `implicitly declaring library function 'strlen'`

Don't forget to #include <string.h>.

30. `more '%' conversions than data arguments`

Functions like printf & sprintf require some value to fill the placeholders %i, %c, %s etc. Did you forget to put one in? For example printf("Go %i spaces!"); instead of printf("Go %i spaces", n);

31. `definitely lost: 40 bytes in 1 blocks`

Remember to free(); all memory you allocated with malloc() or realloc() etc.
(heap)

32. `Invalid write of size 4`

You wrote 4 bytes more of memory than you had access to— check to see if you indexed beyond the bounds of an array (for example did you store an int in array[5] of an array of size 5?) or tried to store something with a NULL or invalid pointer.

for staff use only

- 0