Robbie Gibson-Practice Questions **Name:** ____________________

Quiz 0

1. Angela loves Catalan numbers and their cool properties in combinatorics. Write a function to find the nth Catalan number. The Catalan numbers are expressed as:

$$\frac{(2n)!}{(n+1)!n!}$$

2. Below is a C program that sets a lot of variables. In the table, write the value of each variables after the given line has been executed. You should use &x and the like when writing the value of the pointers.

```
 1  int main(int argc, char** argv)
 2  {
 3      int x = 2, y = 3, z = 4;
 4      int* a = &x;
 5      int* b = &y;
 6      int* c = &z;
 7
 8      y += *a;
 9      z = y + *b;
10      *a = x + *b;
11      c = b;
12      *b = *c;
13      y = z + *c;
14
15      return 0;
16  }
```

Line	x	y	z	a	b	c
6						
8						
9						
10						
11						
12						
13						

3. True or False:

    1. Malloc allocates memory on the stack.
    2. A string in C is an array.
    3. It's possible to sort an array in $O(n \log n)$ time.
    4. Angela creates and declares an int array, but doesn't initialize it. It's filled with 0's.
    5. Typing `next` in gdb will step into a function called on that line.
    6. The size of an int is 4 bytes.
    7. 4 bytes is 36 bits.

4. Angela writes some code to determine if a number is prime. Unfortunately, it's buggy and that makes her sad.

   How can you fix this implementation?

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

int main(int argc, char** argv) {
  int n, flag;

  printf("Enter value of N > ");
  n = GetInt();

  flag = 0;
  for (int i = 2; (i < (n / 2)) && flag; ) {
    if (n % i == 0) /* If true n is divisible by i */
      flag = 0;
  }

  if (flag)
    printf("%d is prime\n", n);
  else
    printf("%d has %d as a factor\n", n, i);
  return 0;

}
```

5. How many different, unique values can you represent with 4 bits?

6. Angela's arguing with Robbie about how to best sort an array. Angela thinks MergeSort is faster, but Robbie keeps saying Bubblesort is faster. Who's correct, what are the best-case and worst-case runtimes, and why are the two runtimes different?

7. Angela says a pointer is a variable whose value is the direct address of a memory location.

    1. Is there anything wrong with this definition? What's wrong with it?
    2. Declare a pointer.
    3. Why are pointers useful?

8. Write your own string compare function. The real `strcmp()` has complicated return values, but yours should `return 1` if the strings are equal and `return 0` otherwise. Use pointer arithmetic to check the string equality and not array indexing.

```
int strcmp(char* str1, char* str2)
{
```