Miles Grossenbacher

Problem 1.

(a) 11
(b) Because of the structure of a “for” loop, i gets incremented at the end of each iteration and then checked for the boolean condition to start the next iteration. At the end of the final loop, our for loop increments i from 10 to 11, which makes the condition for the loop false, but then i gets printed after the loop and is still at the value 11.

(a) 458358838
(b) Yes, this number is some random junk number because the int i in the scope of the main() is never initialized with a value. Initializing the i in the for loop makes i a completely different variable inside the scope of the loop, but not in the scope of the main().

Problem 2.

(a) The destination has the entire string, but the source string got messed up
(b) This happens because the string copy causes a buffer overflow since the destination can’t hold the entire string. This causes characters to be written past the memory allocated for the destination, and since we first initialized the source and then the destination, the destination is directly above the source in the stack. Because of this, the overflow of characters goes into the memory allocated for the source including the terminating null character. A null character signifies the end of a string to the compiler, so since the compiler sees a terminating character in the source allocated memory, it terminates after reading the part of the string from the destination that overflowed.
(c) As long as the source and destination are both large enough to fit the entire length of the string in their allocated memory, the strcpy method will be able to copy the source into the destination without overflowing. So I just made the destination >= the number of characters in the string and it worked.

(a) No, because it cut-off the string after 16 characters which means it didn’t hit the first terminating character, so it continued on until it hit the terminating character at the end of the source string.

Problem 3.

(a) src is changed because dest overflows into src after 16 characters, and src sees a null terminating character after four characters.
(b) Since scanf can use formatters, we explicitly told it that we only want the first 15 characters of the input stored in destination. This avoids the problem of an overflow, so src doesn't get affected and prints whatever we stored in it.

Problem 4.

```
pascal[i][j] = pascal[i-1][j-1] + pascal[i-1][j];
```

p[0][0] = 1
p[1][0] = 1 p[1][1] = 1
p[2][0] = 1 p[2][1] = 2

(a) if the row number is a power of 2, then all of the numbers in that row will be odd. Visually, there is also a repetitive pattern of triangles that make up the large triangle.