• This lab has two checkpoints that are to be completed by the end of your assigned lab section. Show your code and results to a TA or mentor to receive credit.

• This lab is to be completed individually. Do not share your work or code with anyone else.

• Labs are generally available by 6:00PM on Mondays before your lab sessions. Plan to start each lab early and ask questions during office hours, in the Submitty Discussion Forum, and during your lab session.

1. Write a program that repeatedly asks the user for a positive odd integer \( n \), building a triangle for each valid input, as shown in the example below. If invalid input is given, display a suitable error message and allow the user to start again. Implement this using nested for loops.

   What is \( n \)? 7
   *******
   *****
   ***
   *

   What is \( n \)? 11
   ***********
   ********
   ****
   ***
   *

If you did not do so in the first place, write a separate function to build this triangle. And to stop program execution, use \(-1\) as the sentinel value to end the program (and let the user know this in the prompt). (v1.1) Look for “field width” in the man page for printf() and make use of ‘*’ with an argument.

Next, write another function to accomplish the same results, but use printf() to achieve the formatting and at most one loop. Check out the man page for (v1.1) memset() for how you might do this.

Finally, have your program calculate and display the area of each triangle. Assume the triangle is an isosceles triangle with the given base of length \( n \) and a height equal to the number of rows used to build the triangle. Use printf() to display exactly two digits of precision.

2. The Fibonacci sequence is calculated recursively by summing the previous two values of the sequence, i.e., \( \text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \). Assume that this sequence starts with \( \text{fib}(0) = 0 \) and \( \text{fib}(1) = 1 \) as its first two elements. Write a program (using long and not int for the return value of fib) that asks the user for a non-negative integer, then computes its Fibonacci number using a recursive function. Run your program on larger and larger numbers. Does it take a long time to compute?

Write a separate function to calculate \( \text{fib}(n) \) without using recursion. Compare how long it takes to compute without recursion. How high can you go until your program crashes?