

Homework 1 (20 points)

Due Friday January 28 by 9:30AM. Remember you can turn it in late with a 10% per day late penalty.

In this assignment you will write a few different C functions and call them from your **main** function. Create a directory named **hw1** in your personal course repo. Note the spelling. Make sure the directory is named **hw1** and not **HW1** or **Hw1**, etc. Also, a general rule of thumb as a programmer; never, ever, put spaces in a file name or directory, on any operating system. Make sure all of the files outlined below are in your **hw1** directory and in your course repo by the due date and time.

1. [5 points] Write a C function **sum3or5** that takes an integer parameter **n** and returns the sum of the integers less than **n** that are multiples of 3 or 5. For example if **n** was 10 then the function would return 23 because $3 + 5 + 6 + 9 = 23$. Put your code in a file named **sum3or5.c** and make sure it compiles separately using the command **gcc -c sum3or5.c**. Make sure to test your code on several values of **n**. When **n** is 16 the answer should be 60. When **n** is 1000 the answer should be 233168.

2. This problem should go in a file named **sequence.c** and should be separately compilable using the command **gcc -c sequence.c**

a) [5 points] Consider the integer sequence defined as follows. Starting with any positive integer:

If **n** is even, the next number in the sequence is $n / 2$. If **n** is odd, the next number in the sequence is $3n + 1$. It is conjectured (thought it is not known) that every such sequence eventually reaches 1. For example, if we started with 10 then the sequence is **10 5 16 8 4 2 1**.

Write a function named **sequence** that takes an integer **n** and returns the length of the sequence generated. In the above example this would be 7.

- b) [5 points] Write a function named **longest** that returns the $n \leq 1000000$ that generates the longest sequence. This function takes no parameters.
3. [5 points] Write a function named **diamond** that prompts the user for and reads a non-negative odd integer **n** from the user and prints a diamond pattern of **n** lines made up of asterisks. The function should be in a file named **diamond.c**. The middle row of asterisks should have **n** asterisks. The function should continue to prompt the user until they enter a -1. Here is a sample run of the function.

Enter non-negative odd integer: 5

```
  *
 ***
*****
 ***
  *
```

Enter a non-negative odd integer: 7

```
    *
   ***
  *****
 *****
  *****
   ***
    *
```

Enter a non-negative odd integer: -1
Bye!

4. Include a function named **main** in a file named **main.c** that calls each of your functions above.