CSc 110 Assignment 4:
Count driven loops

Learning Outcomes:

When you have completed this assignment, you should understand:

- How to design functions that contain count driven loop.

How to hand it in:

Submit your `assignment4.py` file through the Assignment 4 link on the CSC110 conneX page.

Grading:

- Late submissions will be given a **zero grade**.
- You must use the file `assignment4.py` provided to write your solution. Changing the filename or any of the code given in the file will result in a **zero grade**.
- Your function names must match exactly as specified in this document or you will be given a **zero grade**.
- Function arguments must be exactly as specified in this document. Specifically, do not change the number of and/or order of the arguments or you will be given a zero grade for the function.
- We will do **spot-check grading** in this course. That is, all assignments are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors on the ECS 258 Lab machines or a **zero grade** will be given.
- It is recommended that you use a plain text editor such as Notepad++ as used in the labs or Atom for Mac computers. We also recommend you run your programs through terminal / command prompt, as shown in the labs.
- It is the responsibility of the student to submit any and all correct files. Only submitted files will be marked. Submitting an incorrect file is not grounds for a regrade.
- If the assignment requires the submission of multiple files, then all files must be submitted.

Marks will be given for…

- your code producing the correct output
- your **tests** for your functions **providing sufficient coverage** (at least 2 tests and enough to cover all possible paths through the code)
- your code following good coding conventions (see lab and lecture code for examples)
    - Proper indentation
    - Documentation of purpose, arguments and return value using docstring format
    - Names of variables should have meaning relevant to what they are storing
    - Use of whitespace to improve readability
    - Proper use of variables to store intermediate computation results

Get started…

- Download `assignment4.py` from conneX ***Files*** tab and save it to your working directory.
- Write your name and student V# in `assignment4.py`
- Part 1 and Part 2 below provide the specification for a set of 7 functions (1 in Part 1 and 6 in Part 2) you must define and test.
- For each of the 7 function specifications provided below you must:
    - Write **calls** to the function within the main function provided in `assignment2.py`
      NOTE: You are free to comment out these tests as you progress through the assignment so they do not run but you MUST leave them in place for us to grade
    - Write the function **definition** according to the specification provided
      NOTE: function names and the order and number of the arguments must be EXACTLY as specified or you will receive a zero grade on that function.

Part 1 – Loops that accumulate a result

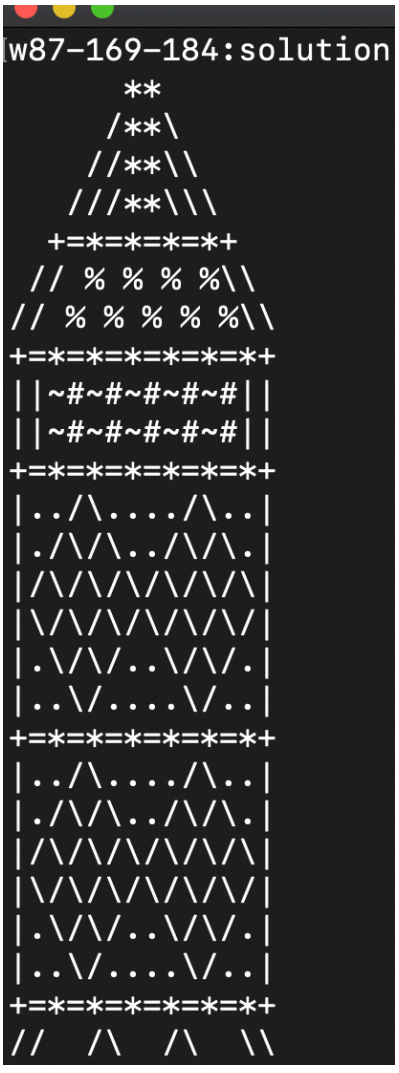
NOTE: function names and the order of the arguments must be EXACTLY as specified or you will receive a zero grade on that function.

1. Design a function called `sum_odd` that takes an integer n, and sums all of the odd numbers from 0 to n **inclusive**. The function must **return** the sum. Don't forget to write your tests.

Part 2 – Loops that print output

In this part of the assignment you are going to write a set of functions that will be used to output an ascii image of a Rocket.

A cropped screenshot of a sample output is shown here:



The `draw_rocket` function that you will design will take two arguments:

1) an integer for the **size** of the rocket
(in this example the size is 2)

2) an integer for the **number of boosters** the rocket will have
(in this example there are 2 boosters)

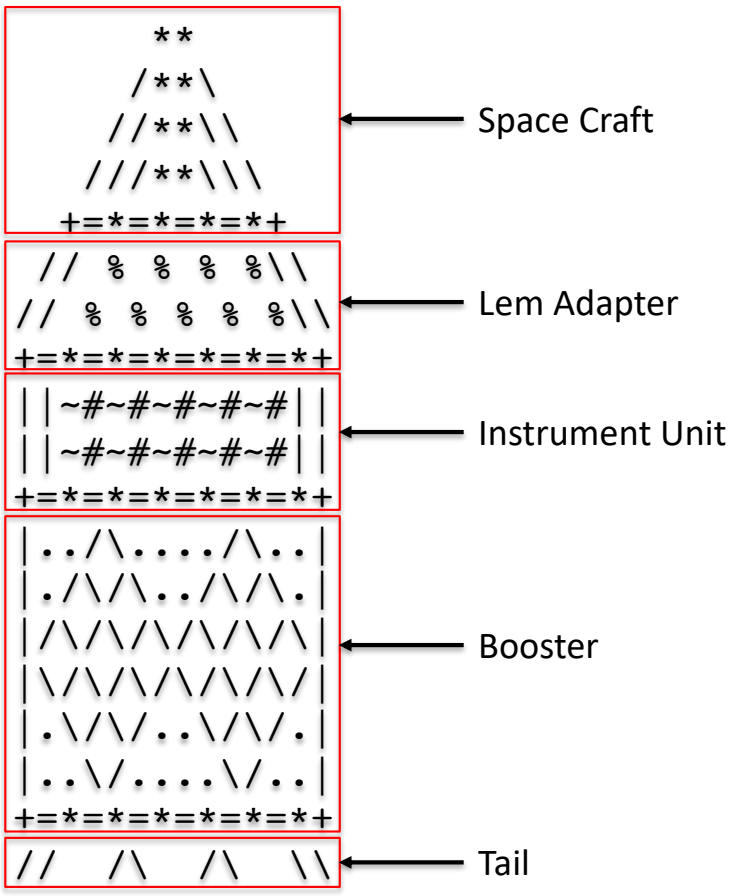
To encourage good software development practices we are requiring you to write and call a specified set of functions in your implementation.

You are free to include additional functions, but to receive full marks your implementation must include the required set described on the following page.

Decomposing the problem:

We are asking you to decompose your program according to the following diagram, where you will implement at least one function for each of the diagrammed parts in the image below.

This image was created with by calling `draw_rocket` with a size: 2 and number of boosters: 1



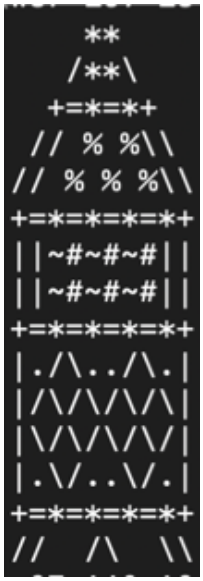
Your functions must adhere to the following specifications including **function names** and **order of arguments**. These functions must print the portion of the image in the figure above where the function name corresponds to the label name in the figure.
For example, the `draw_booster` function must print all parts of the shape labeled "Booster" and enclosed in the red square in the figure above.
A **zero** grade will be given for any function that does not follow the following specification:

1. `draw_tail` must take as an argument one integer representing the size of the rocket it is being drawn for
2. `draw_booster` must take as an argument one integer representing the size of the rocket it is being drawn for and print only ONE booster
3. `draw_instrument_unit` must take as an argument one integer representing the size of the rocket it is being drawn for
4. `draw_lem_adapter` must take as an argument one integer representing the size of the rocket it is being drawn for
5. `draw_space_craft` must take as an argument one integer representing the size of the rocket it is being drawn for
6. `draw_rocket` must take as arguments an integer representing the size of the rocket and an integer representing the number of boosters to be printed. Your implementation of this function must make use of the other functions you have written to draw the rocket.

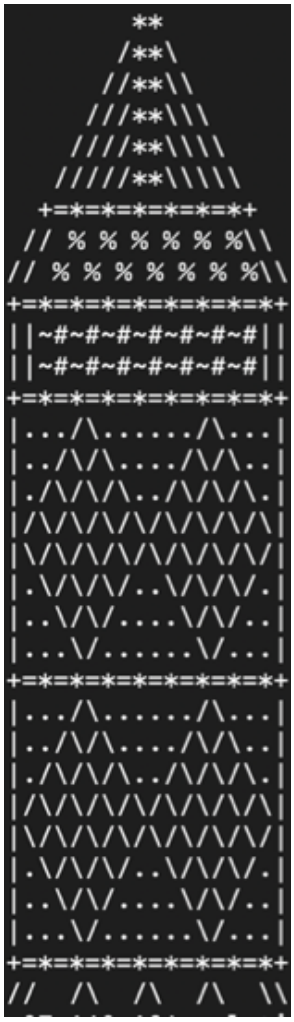
More cropped screenshots of sample runs have been provided below to give you idea of how the rocket should scale with a change in **size** and **number of boosters**.

Take your time with pen a paper to identify the relationship between the size and the output.

Created with:
`draw_rocket(1,1)`



Created with:
`draw_rocket(3,2)`



Created with:
`draw_rocket(4,3)`