Faculty of Arts and Sciences
Department of Computer Science
CMPS 200 – Introduction to Programming
Assignment 5 – Due Oct 15, 2018

Reading Material:

- Section 4.3 from textbook

1. Password checker.
Write a program `pwd_check.py` that reads a string from the command line and checks whether it is a "good" password. Assume "good" in this context means that it (i) is at least 8 characters long, (ii) contains at least one digit 0-9, (iii) contains at least one upper case letter, (iv) contains at least one lower case letter, and (v) contains at least one non-alphanumeric character.

2. Caesar.
One of the oldest encryption schemes is attributed to Julius Caesar. Caesar used to send secret messages to his friends using a scheme that is now known as a Caesar cipher. Each letter is replaced by the letter k positions ahead of it in the alphabet (and you wrap around if needed). The table below gives the Caesar cipher when $k = 3$.

```
Original:  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Caesar:    D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

For example the message VENI, VIDI, VICI is converted to YHQL, YLGL, YLFL.

Write a program `caesar.py` that takes a command line parameter k and a string to be encrypted, and applies a Caesar cipher with shift = k to the sequence of letters in the string. You program should handle lower and upper case letters only, and leave all other characters (punctuation marks, digits, whitespace, etc.) unchanged. For example,

```
> python caesar.py 3 'VENI, VIDI, VICI'
YHQL, YLGL, YLFL
```

You will need to write:

- a function that takes in an integer argument, and one of the 26 uppercase (A-Z) or 26 lowercase (a-z) letters of the alphabet, and returns the shifted letter.
  Hint. The functions `ord()` and `chr()` return the integer ASCII value of a character and vice versa.
- a function that takes in a string and an integer defining the cipher, and returns an encrypted version of the string.

and use them in your program.

3. Draw cards.

- write a function `deck()` that constructs and returns a list of 52 strings representing a deck of cards. Your strings should be of the following form: `'10 of Spades'`, `'Queen of Hearts'`, etc. The four suits in a deck of cards are called Hearts, Diamonds, Spades, and Clubs, and the thirteen ranks are 2-10, Jack, Queen, King, and Ace.
- write a function `choose_cards(n)` that returns a list of n cards from a full 52-card deck chosen *with replacement*, i.e. the card is returned to the deck after every draw and may be randomly chosen again.
- write a function `draw_cards(n)` that returns a list of n cards from a full deck *without replacement.* An algorithm for doing so will first shuffle the deck and return the first n cards from it.
- write a function `suit_count(hand)` that takes in a list of cards and returns a list of four numbers containing how many cards there are in every suit starting with spades, followed by hearts, diamonds, and finally clubs.

Write appropriate calls to test the functions you write.

4. Pythagorean triplets.
Pythagorean triplets are sets of 3 positive integers a, b, c satisfying the relationship $a^2 + b^2 = c^2$. The smallest and best-known Pythagorean triple is $(a, b, c) = (3, 4, 5)$. Write a program that reads a command line argument n and prints to the screen all Pythagorean triplets whose sum is less than n (i.e., $a+b+c < n$) and that are not multiple of the $(3, 4, 5)$ triplet.

Your program will represent triplets as 3-tuples, and should consist of three functions:

- a function that takes in a tuple and returns a boolean indicating whether the Pythagorean relationship holds or not.
- a function that takes in a tuple and returns a boolean indicating whether a triplet is a multiple of the smallest triplet or not.
- a function that takes in an integer n and generates the Pythagorean triplets as specified above. The function should return a list of tuples.

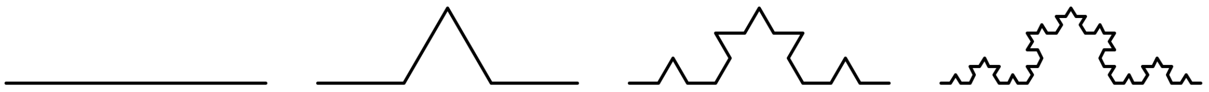
The main portion of your program `pythagore.py` will read in the command line input, call the last function described above, and print the results one triplet per line.

5. Longest palindrome.
Write a function `longest_palin(st)` that returns the longest substring in `st` that is a palindrome. A brute force method of finding the longest palindromic substring is to loop over all possible substrings and keep track of the longest one encountered. To loop over all substrings, you may use a nested loop with an outer counter i $(0 \leq i < n)$ and an inner counter j $(i \leq j < n)$ to generate the substrings `st[i:j+1]`. Use the `is_palindrome()` function you wrote earlier. For example, the call `longest_palin('43xyyx1')` returns `'xyyx'`.

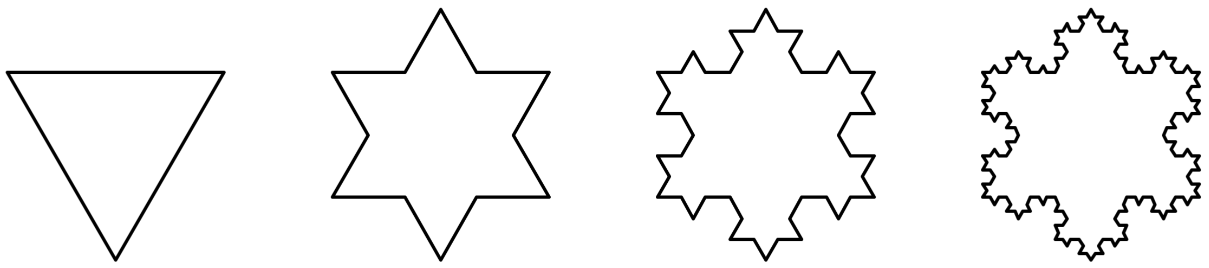
6. Factorial, recursive version
Write a recursive version of the factorial function `rfactorial(n)` which returns the value of $n!$ where n is assumed to be a positive integer.

7. Palindrome, recursive version
Write an `is_palindrome2(st)` function, a recursive version of the function that returns a boolean determining whether or not its argument is a palindrome.

8. Snowflake.
Write a recursive function `fractal(order, length)` that draws a fractal curve similar to the ones below. A fractal line of order 0 is a straight line segment of a given length. A fractal line of order 1 is obtained as follows: instead of drawing just one line, draw instead four smaller segments of length/3 each and with 60^o angles. An order 2 fractal line is obtained by repeating the same pattern on each of the order 1 segments. Repeating the pattern again gets us an order 3 fractal line such as the one shown on the right below.



Write a function `snowflake(order, length)` that draws "fractal snowflakes". The snowflake is an equilateral triangle, where each side is a fractal line of a given order and length.



Note. The fractal snowflake of infinite order is a shape that has a number of curious properties: its perimeter is infinite while its area is finite (equal to 8/5 of the triangle); it is a continuous curve but does not have a tangent anywhere. For more details check out the wikipedia page on "Koch snowflake".

9. Flatten.
Write a function `flatten(lst)` that returns a simple list containing all the values in a nested list. For example, `flatten([2,9,[2,1,13,2],8,[2,6]])` returns `[2,9,2,1,13,2,8,2,6]`

Zip your files in a single archive file `asst5_`*`netid`* where *`netid`* is your AUBnet user name. Your submission to Moodle must be received by noon of the due date. Late submissions will get no grade.