1. Create a function two-three-or-five that takes a positive integer n and produces the list of positive integers from 1 up to n that are divisible by 2, 3, or 5.

For example, two-three-or-five (7) should produce the list [2, 3, 4, 5, 6].

KU: 1 – data structure (creating the list)

Appl: 2 - update the list using a function

T: 1- algorithm design (I/O/Process)

T:3 – test plan

Comm:1 – comments

1. Consider the following data definition: a card is an object with two attriburtes”

* Rank (a digit from 1 to 10, or one of the following characters: A, J, Q, K)
* Suit: hearts, diamonds, clubs and spades

Create a function all-reds that take a list of cards and produces a list of the red-suited cards (hearts and diamonds).

KU: 2 – data structures (the original list and the final one)

KU: 2 – class definitions and methods \_\_init\_\_and\_\_str\_\_

Appl: 3 – creating the final list using a function

T: 1 – algorithm design (I/O/Process)

T:3 – test plan

Comm: 1 – comments

1. Create a function count-suits that takes a list of cards (as defined in the previous question) and produces a suit-count list indicating the number of cards of each suit.

KU: 2 – data structures (the original list and the final one)

KU:2 – class definitions and methods\_\_init\_\_and\_\_str\_\_

Appl: 2 – creating the final list using a function

T: 1 – algorithm design (I/O/Process)

T:3 – test plan

1. Write a function build-range which takes three numbers start, finish and pts, and produces a list containing pts numbers, evenly spaced from start to finish. You may assume that pts is an integer with value at least 2; if finish is less than start, then the entries in the produced list will be in decreasing order.

For example, build-range (2 3 5) should produce the list [2, 2.25, 2.5, 2.75, 3]

KU: 1 – data structure (creating the list)

Appl: 2 - update the list using a function

T: 1- algorithm design (I/O/Process)

T:3 – test plan

Comm:1 – comments

1. Write a function make-oh that takes a positive integer k and prints a k-size letter O. A k-size O consists of a top, a middle, and a bottom, each consisting of k rows of x’s. The rows of a middle each consist of k x’s placed over the blanks in the middle; each subsequent row has two or more x’s added at the outside edges of the previous line of x’s. A bottom is like a top but upside-down. You may refer to the examples below (for k=2 and k=3)

xx

xxxx

xx xx

xx xx

xxxx

xx

xxx

xxxxx

xxxxxxx

xxx xxx

xxx xxx

xxx xxx

xxxxxxx

xxxxx

xxx

Appl : 3 – the function make-oh

T: 2 – algorithm design (