Programming Assignment 9

1. Create a function, called **findString**, that takes a string and a file name as arguments and prints all lines in the file which contain the specified string (regardless of capitalization). Create a try and except clause to catch the error where the file name passed into the function is not found (this might happen either because the name of the file is misspelled or does not exist in the directory given) and prompt the user to enter the correct file name until successful. (Test using any text file) (20 points)

2. Write a function, called **wrapLines**, that takes a file name and a length integer k. The function wraps the lines longer than width k by splitting them into lines of length less than or equal to k, just like resizing a text editor window. Make sure not to break in the middle of a word. If the word is longer than k, you may split it using a hyphen or leaving the entire word on a single line. Create a try and except clause to catch the error where the file name passed into the function is not found (this might happen either because the name of the file is misspelled or does not exist in the directory given) and prompt the user to enter the correct file name until successful. (20 points)
   Hint: You will need to read the file line by line then write the result back to the same file. (You can test this using any text file)

3. Create a function, called **allAnagrams**, find all anagrams in a dictionary. An anagram is a word formed of the rearrangements of the letters of another word. (20 points)
   For example, listen and silent. (A dictionary file is provided in the assignment folder. Testing your code on the entire dictionary may take a while so you can create a subset of words from the dictionary file to test your code).

4. Breast Cancer Data (40 points)

This data set comes from:

https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

After a biopsy of a tumor tissue tests are run on the tumor cells to determine a diagnosis of “benign” or “malignant”. The tests result in 30 different cell attribute measurement values. Some or the measured aspects are “radius_mean”, “perimeter_mean”, “area_mean” which measure the mean value of cell radius (distance from center point), perimeter, and area. Looking at the web page and the data set you can see the other 27 different values. Based on these 30 values, a formula is applied to determine with there tumor is malignant or benign.

The two following files show the data:

- breastCancerDataReducedDimensions.cvs: Only the first 4 attributes (you can just read this file instead of the file containing the entire set)
- breastCancerData.csv: The full data set

Note, the first column is the sample id. The second column is the diagnosis for the sample, where “M” means malignant and “B” means benign.

At lunch one day, you and a medical technician come up with the idea that all this data and complicated formula are not needed. Instead, you decide you just need to look at the first four metrics {radius, texture, perimeter, area} means. The process is as follows:

a) strip the data to only consider those 4 values

b) Create four data files:
   q3_gte_13: third attribute - those data samples whose radius value is >= 13
   q4_gte_18: fourth attribute - those data samples whose texture value is >= 18
   q5_gte_85: fifth attribute - those data samples whose perimeter value is >= 85
   q6_gte_500: sixth attribute - those data samples whose area value is >= 500

c) Find the data ids that are in each of these four files. The idea is that if a data sample exceeds the threshold (13, 18, 85, and 500) for each of these 4 attributes then the tumor is **malignant**. If the data does not exceed any of these attributes, then the tumor is **benign**. If the tumor exceeds some, but not all, of these thresholds then the tumor could be **either** benign or malignant.

To do this, you need to take the intersection of 4 files where the files have the ids of these data sets.

The four files (q3_gte_13, q4_gte_18, q5_gte_85, q6_gte_500) also have the diagnosis of “B” or “M” from the original test included. You want to test the quality of your process.

To do this create 2 versions for each of the 4 dimensions:

q3_B, q3_M

Where B means the data has been diagnosed as Benign and M means it was diagnosed as Malignant.

Likewise for columns 4, 5, and 6 giving files:
q4_B, q4_M
q5_B, q5_M
q6_B, q6_M

With these files you can now test how well your new easier process works.

Let file **NewResult** contain the intersection of ids from the four files (q3_gte_13, q4_gte_18, q5_gte_85, q6_gte_500), i.e. regardless of whether the original methods said M or B.

There are two ways to test this new process.

Method 1:

Compare **NewResult** to the data found in the 4 files you created of ids of M data: q3_M, q4_M, q5_M, and q6_M. Let **SubsetMResult** contain the data that is the union of the four files (q3_M, q4_M, q5_M, and q6_M). Then, calculate:

Difference_1 = SubsetMResult - NewResult

If your new method is capture all the same data, then Difference should be the empty set.

Method 2:

From the original data set, breastCancerData.csv, find all the ids that are marked M. Call this set **OriginalResult**.

Difference_2 = OriginalResult - NewResult

Again, if your new method is capture all the same data, then Difference should be the empty set.

To turn in:

contents of difference 1 (in sorted order)

contents of difference 2 (in sorted order)

A short (4 sentence) answer to Q1: What does the value of set difference_1 obtained from method 1 tell you about how well your new method works?

A short (4 sentence) answer to Q2: What does the value of set difference_2 obtained from method 1 tell you about how well your new method works?

A short (4 sentence) answer to Q3: Why are difference_1 and difference_2 not the same?