Comp 132 Problem Session 4 – March 4, 2019

Question 1

In the file huntington.txt attached find a list of patients together with their names, surnames, and chromosome 4 DNA sequences. Every line represents a different patient entry.

The gene that causes Huntington's disease is located on chromosome 4 and has a variable number of repeats of the CAG trinucleotide repeat. Write a program to determine the number of repeats and print will not develop HD If the number of repeats is less than 26, offspring at risk if the number is 37-35, at risk if the number is between 38 and 39; and will develop HD if the number is greater than or equal to 40. This is how Huntington's disease is identified in genetic testing.

- Read patient information from the huntington.txt file using Scanner.
- Write an enum class DiseaseStatusHD specifying the possible disease status: WILL NOT DEVELOP HD, OFFSPRING_AT_RISK, AT HD RISK, WILL DEVELOP HD.
- Write a method DiseaseStatusHD getDiseaseStatus (String sequence) using enum class you specified that returns one of the disease possibilities according to previous explanation.
- Write a method int numberOfRepeates (String sequence, String dnaRepeat) that returns the number of repeats of user specified DNA fragment in a sequence. You are supposed to use the Matcher class. Look at the documentation online.
- Print a report for all patients present in the huntington.txt file; with patient name, surname and disease status into a text file called HD_report.txt using Formatter.

Your HD_report.txt file should look like following:

```
#Name   Surname Number of Repeats       Disease Status
Johnny  Adams   177     WILL_DEVELOP_HD
Leila   Thomas  75      WILL_DEVELOP_HD
Jack    Berg    23      WILL_NOT_DEVELOP_HD
Daniel  Smith   38      AT_HD_RISK
Stefan  Hopkins 37      OFFSPRING_AT_RISK
```

Question 2

(Defining Your Own String Methods) Write your own versions of String search methods:

- **int indexOf (String input, char search).** This method searches for char “search” in the String “input”. It returns the **first** index at which char “search” is found taking the first index of the string as 0. If the character is not present in the string this method returns -1.
- **int lastIndexOf (String input, char search).** This method searches for char “search” in the String “input”. It returns the **last** index at which char “search” is found taking the first index of the string as 0. If the character is not present in the string this method returns -1.

Question 3

Given a text string txt[] of length N and a pattern string pat[] of length M, determine whether pat[] or any of its anagrams (any of its M! permutations) appears in the text.

Your algorithm should perform in a **linear time** (Hint: maintain a histogram of the letter frequencies for a given substring of length M in the text.).

Console example is following:

```
Enter the text string to be searched.
djsioufoeiujfdmcsdmoeffjrjgjpoedAdfjeCVfrfkeoejfuieasksldjiofsiue
Enter the pattern you wish to look for and press Enter.
iue
anagram found : eiu at index 8
anagram found : uie at index 48
anagram found : iue at index 62
In the entered text string there are 3 anagrams of the entered pattern.
```