COEN 352 FALL 2019 PROJECT

Project Goal: Given a dictionary of distinct words and an input text file, find the frequency of words in the text file that exist in the dictionary.

Dictionary: A text file consists of a distinct set of words separated by spaces.

dictionary.txt

able anywhere came Carpathians dinner dish done get good indeed know left Mina national paprika pepper pretty rather recipe red said smattering stopped time way will with wolves woman word work worse your

Input Files: Input files are chapters from a book. They include punctuation, new lines and other characters. There are 3 types of input files:

- Clean: One-page text files with around 300 words.

clean0.txt

We left in pretty good time, and came after nightfall to Klausenburgh. Here I stopped for the night at the Hotel Royale. I had for dinner, or rather supper, a chicken done up some way with red pepper, which was very good but thirsty. (_Mem._, get recipe for Mina.) I asked the waiter, and he said it was called "paprika hendl," and that, as it was a national dish, I should be able to get it anywhere along the Carpathians. I found my smattering of German very useful here; indeed, I don't know how I should be able to get on without it.

- Space Removed: One-page text files with random spaces removed.

removed_spaces0.txt

We left in pretty good time, and came after nightfallto Klausenburgh. Here I stopped for the night at the Hotel Royale. I had for dinner, or rather supper, a chicken done up some way with red pepper, which was very goodbut thirsty. (_Mem._, get recipe for Mina.) I asked the waiter, and he said it was called "paprika hendl," and that, as it was a nationaldish, I should be able to get it anywhere along the Carpathians. I found my smattering of German very useful here; indeed, I don't know how I should be able to get on without it.

- Character Mutated [Bonus]: One-page text files with a character in a word substituted with another random character (any character including spaces, newline, tabs, etc.)

mutated0.txt

> **We left in pretty g=od time, and came after nightfall to Klausenburgh. Here I]topped for the night at the Hotel Royale. I had for dinner, or rather suppe4, a chicken done up some way with red pepper, wich was very good but thirsty. (_Mem._, get recipe for MiRa.) I asked the waiter, and he said it was called "paprika hendl," and that, as it was a national dish, I sould be able to get it anywhere along the Carpathians. I found my smattering of German very useful here; indeed, I don't know ho+ I should be able to get on without it.**

Project Description:

The dictionary of words must be stored in a BST implemented purely by you. At the beginning of the program the dictionary text file is read, and a BST is created with those words. Any search from the dictionary words must be done using the BST.

Part 1 – clean files

Input: a clean text file for example clean0.txt

Output:

1. Frequency of words (case insensitive) in the input file that exists in the dictionary, **sorted by their frequencies and then alphabetically in ascending order for both.**
2. Text file with words repeated by their frequencies sorted alphabetically.
3. word cloud image generated using https://www.wordclouds.com by imputing the input of repeated.txt file
4. A bar plot of the frequency of words generated using excel by imputing the input of frequencies.txt and plotting the bar plot of the data.

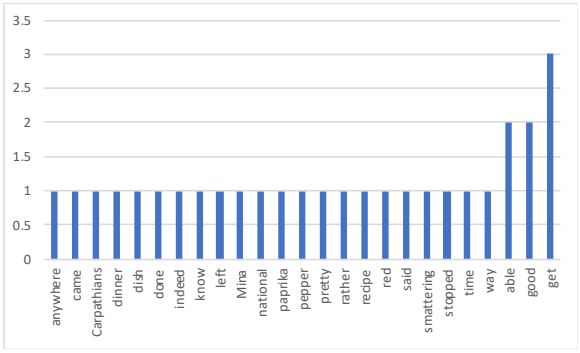
Example of output for the example given above in dictionary.txt and clean0.txt

frequencies.txt

anywhere 1
came 1
Carpathians 1
dinner 1
dish 1
done 1
indeed 1
know 1
left 1
Mina 1
national 1
paprika 1
pepper 1
pretty 1
rather 1
recipe 1
red 1
said 1
smattering 1
stopped 1
time 1
way 1
able 2
good 2
get 3

repeated.txt

able able anywhere came Carpathians dinner dish done get get get good good indeed know left Mina national paprika pepper pretty rather recipe red said smattering stopped time way



Part 2 – Space Removed files

Input: a space removed text file for example removed_spaces0.txt

Output:

- Same output as part 1: frequencies.txt, repeated.txt, word cloud diagram and bar plot.
- A file containing each word with space removed detected as a word in the dictionary presented in its original form and after adding the space.

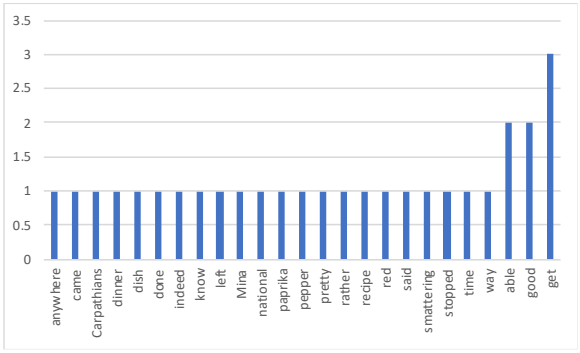
Example of output for the example given above in dictionary.txt and removed_spaces0.txt

frequencies.txt

```
anywhere 1
came 1
Carpathians 1
dinner 1
dish 1
done 1
indeed 1
know 1
left 1
Mina 1
national 1
paprika 1
pepper 1
pretty 1
rather 1
recipe 1
red 1
said 1
smattering 1
stopped 1
time 1
way 1
able 2
good 2
get 3
```

repeated.txt

able able anywhere came Carpathians dinner dish done get get get good good indeed know left Mina national paprika pepper pretty rather recipe red said smattering stopped time way



corrected_words_detected.txt

```
goodbut, good
nationaldish, dish
```

Note: if the words but and national were part of the dictionary then the output would be

goodbut, good but

nationaldish, national dish

Part 3 – Character Mutated files [Bonus worth 5% of your final course grade]

Similar to Part 2.

Input: a randomly mutated text file for example mutated0.txt

Output:

1. Same output as part 1: frequencies.txt, repeated.txt, word cloud diagram and bar plot.
2. A file containing each word with character removed detected as a word in the dictionary presented in its original form and corrected form.

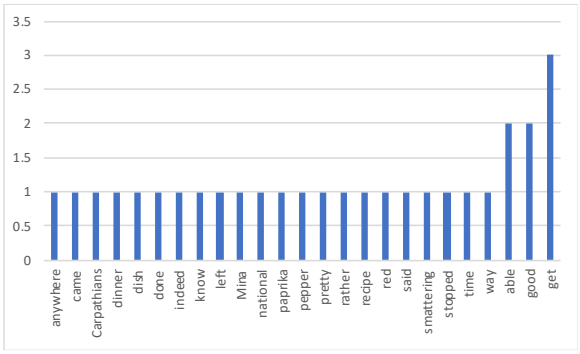
Example of output for the example given above in dictionary.txt and mutated0.txt

frequencies.txt

```
anywhere 1
came 1
Carpathians 1
dinner 1
dish 1
done 1
indeed 1
know 1
left 1
Mina 1
national 1
paprika 1
pepper 1
pretty 1
rather 1
recipe 1
red 1
said 1
smattering 1
stopped 1
time 1
way 1
able 2
good 2
get 3
```

repeated.txt

able able anywhere came Carpathians dinner dish done get get get good good indeed know left Mina national paprika pepper pretty rather recipe red said smattering stopped time way



corrected_words_detected.txt

g=od, good
]topped, stopped
MiRa, Mina

You will be provided a set of input files for every part. The files for part 2 and part 3 are the same as the files in part 1 but with the space being removed and characters being mutated.

For example, removed_spaces0.txt and mutated0.txt were generated from clean0.txt.

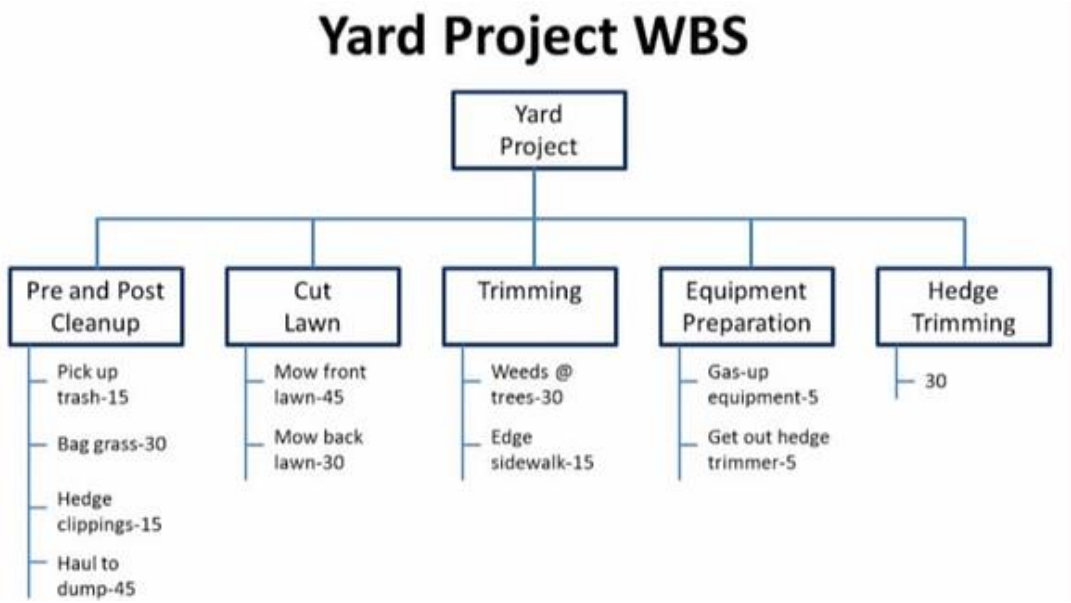
In a perfect world, all 3 parts should result in the same exact frequencies.txt, repeated.txt, word cloud diagram and bar plot.

Project Report

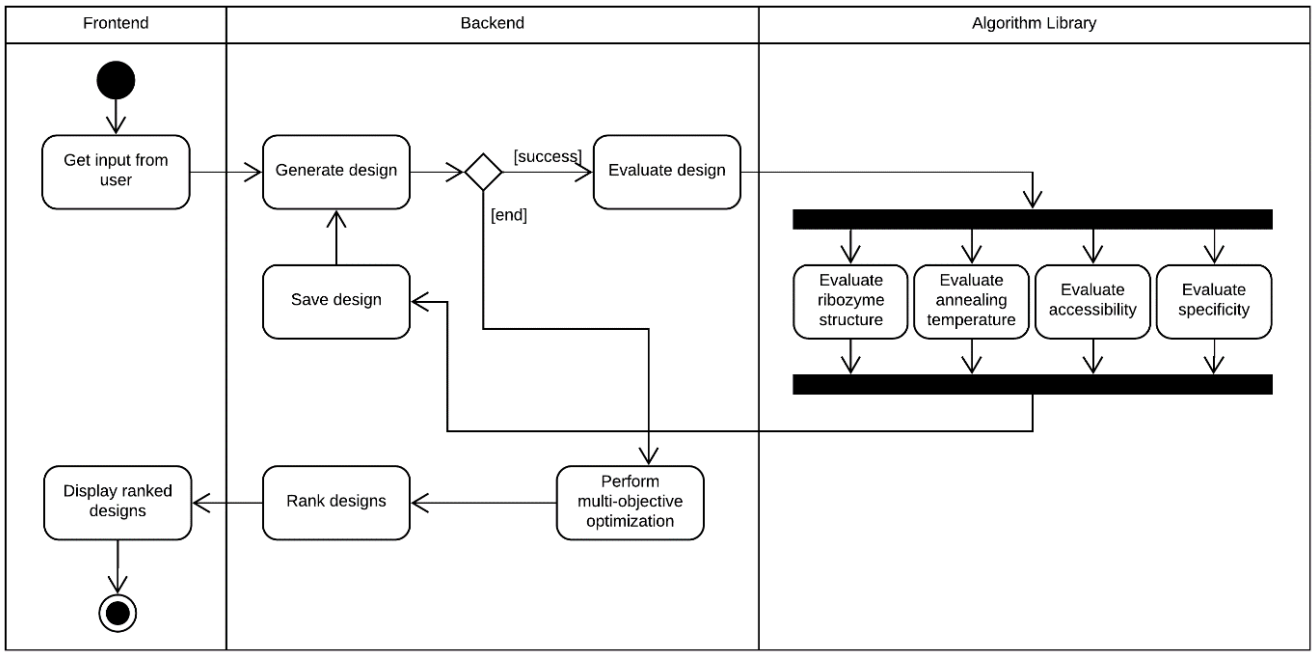
Must include, at least:-

- *Brief* (1-2 paragraph) natural language **Problem Description**, in your own language. The purpose of this exercise is to see if you can describe a complex, multi-part problem using intuitive natural language, to an average non-technical person, so he/she understands *what you're required to do*, rather than how you intend to it.

- **Problem Breakdown:** use a *work breakdown structure* (or WBS) diagram to, *hierarchically*, breakdown the problem into a tree of sub-problems, each presenting a well-defined algorithmic challenge. Below is an example of a normal real-life problem (task) broken down into five main sub-tasks, with each one further divided into a sequence of steps. Your WBS will not look like this one, but the idea is to break-down your problem into constituent sub-problems, each amenable to a single algorithmic treatment, such as searching or sorting.



- **Solution Design**: use an *activity diagram* (such as the one below) to provide a high-level description of the information-processing activities, of your program. Hence, use high-level *pseudo-code* to describe the *most important* activities (e.g., the four evaluation activities in the algorithm library). A pseudo-code example is presented below, also.



Activity Diagram Example for Ribosoft

Algorithm 1 quicksort

```
1: procedure QUICKSORT(a, p, r)
2:     if p < r then
3:         q = PARTITION(a, p, r)
4:         QUICKSORT(a, p, q − 1)
5:         QUICKSORT(a, q + 1, r)
6:     end if
7: end procedure
8: procedure PARTITION(a, p, r)
9:     x = a[r]
10:    i = p − 1
11:    for j = p to r − 1 do
12:        if a[j] < x then
13:            i = i + 1
14:            exchange a[i] with a[j]
15:        end if
16:        exchange a[i] with a[r]
17:    end for
18: end procedure
```

Pseudo-code example for Quicksort

- **Results & Analysis**: this is the results required in parts 1-3 above plus any comments or conclusions you wish to make about them (usually pertaining to interesting or unexpected results). In addition, include average and standard deviation of runtime to process n different files and a plot showing the empirical relationship between runtime vs the size of input file (number of words processed).
- **Mini Manual**: step-by-step instructions on how to use the program correctly.

Grading:
You will be evaluated based on the **correctness of your output, design and implementation of your algorithms and scalability of your solutions.** Scalability will be evaluated relative to the solutions of everyone in class.
The project will be demoed on the 2nd and 3rd of December 2019 to the TAs to evaluate and test your solutions. You will be provided with different dictionary and input files before the demo to be used for grading.

Part 1 (40%):	
Correctness of Output	**20%**
Design and implementation of Algorithms	**10%**
Scalability of solutions	**10%**
Part 2 (45%):	
Correctness of Output	**20%**
Design and Implementation of Algorithms	**15%**
Scalability of Solutions	**10%**
Report	**15%**
Bonus Part 3	**5% towards your final course grade**