New York University Abu Dhabi
CS-UH 1050 – Fall 2019
Programming Assignment 1
Out: Thursday, September 21, 2019
Due: Saturday, October 12, 2019 11:59 pm

Introduction:
In this assignment, you will develop a speller application termed `mydictionary`, that can help a user lookup English words in a provided dictionary. The dictionary files contain one English word per line in lexicographic order. You will create your program using the elementary data structures discussed in class (e.g. Arrays, Vectors). You will have to use the `Linux/Unix/macOS` environment for development.

Overall Description:
Your program should take as command line, the name of the dictionary file to be used.Once the `mydictionary` is in operation, at its prompt the user could search specific word(s). Your program should provide following three key functionalities:

- Full-Word-Search: Program should tell whether a user-provided word is found in the dictionary used.
- Prefix-Search: Program should list all the words of the dictionary that start with a user-provided *prefix*
- Wildcard-Search: Program should list all the words of the dictionary that match the search-string up to one character.

How to Invoke your Application:
Your application should be invoked as following:

```
mymachine-promt >>  ./mydictionary -d <dictionaryFile> -l <MaxNumOfWordsInOutput>
```

where the flag `-d` indicates that the lexeme that follows is the file name of the dictionary to be used and the `-l` flag indicates the number that follows is the maximum number of results to be printed on the output.

Once the program starts, it presents the user with a prompt. Every time, the user types in a query, your application generates and prints out the result(s) and finally, it should give the prompt again to indicate that it expects the next query. Your program should terminate once the user types `exit` at the prompt.

(i) When searching for full words, your program should report:

- whether a word was found or not.
- the number of word comparisons your program performed.

For instance, if you are looking for the word `arcade` and the if the word is part of the dictionary in use, the outcome could be:

```
>>arcade
word found
631 word comparisons carried out
>>
```

(ii) Prefix search is performed through the use asterisk `*`. Which means "match any number of characters until the end of the string". E.g., search for `explo*`. In this case your program should display:

- The number of words in the dictionary that match the non-empty prefix.
- A list of these words shown on the `terminal(tty)` up to `<MaxNumOfWordsInOutput>` entered as the command line argument of `mydictionary`.
- The number of comparisons it took to reach the first match found in the dictionary.

For example if you enters `explo*` in the `terminal`, the outcome could be:

```
>>explo*
4 matches found
exploration
exploratory
explore
explorer
204 word comparisons up-to the first match
>>
```

(iii) The third type of input is a word with exactly one character, at position 2 or later, replaced by the wildcard (any single character). E.g. If the user enters search-string `explo?e`, the program (`mydictionary`) should display:

- the number of words in the dictionary that match the input.
- a list of these words shown on the `terminal`, up to `<MaxNumOfWordsInOutput>` entered as the command line argument of `mydictionary`.
- the number of comparisons it took to return the entire answer.

For exmaple if the user enters `explo?e`, as a query the outcome could be:

```
>>explo?e
2 matches found
explode
explore
250 word comparisons executed
>>
```

You should try to avoid searching the entire dictionary for each query, instead you can perform faster search e.g. binary-search.

Procedural Matters:
◇ Your program should be written in `C++` and must run on the `Linux` , `Unix` or `macOS` operating system.
◇ You will have to first submit your project electronically. If required you will be asked to demonstrate your work.

Group submission:
You should submit your work in groups of two. Both members of a group should be able to answer questions about *any* aspect of the project. Please write down the names of both group members at the top of your code in comments **Only one** member of the group should submit the files before the deadline.

What you Need to Submit:

1. A directory that contains all your work including source, header, `Makefile`, etc.
2. A short write-up about the algorithmic/design choices you made; 2-3 pages in PDF format are expected.
3. All the above should be submitted in the form of a single `zip` or `tar` file bearing your name(s) (for instance `WalterBenjamin-Proj1.tar`).
4. Submit the above zip/tar-ball using *NYUclasses*.

Grading Scheme:

Aspect of Programming Assignment	Percentage of Grade (0–100)
Quality of Code Organization & Modularity	20%
Correct and Efficient Execution of Queries	60 %
Use of Makefile & Separate Compilation	10 %
Well-Commented Code	10 %

Other Noteworthy Points:

1. You have to use *separate compilation* in the development of your program. In other words your code should be distributed into at least two `cpp` files (perhaps more if necessary).
2. We will provide on *NYUclasses* dictionaries files of different sizes.
3. Although it is understood that you may exchange ideas on how to make things work and seek advice from fellow students outside your group, sharing of code is *strictly not allowed*.
4. If you use code that is not your own, you will have to provide *appropriate citation* (i.e., explicitly state where you found the code) Otherwise, plagiarism issues may arise. There are tools for detecting network-wide code similarities and we do use them.
5. Don't share you code on public code sharing websites e.g. GitHub.