CSE102 – Computer Programming

Homework #7

Functions and Selection

Due Date: 10/05/2023 23:59

Hand in: A student with number 20180000001 should hand in a zip file named 20180000001.zip for this homework.

In this assignment, you will learn how to manipulate strings.

Assume that you have the following files for processing:

- input.txt: A file containing a long text in English (for example a novel or an article in a newspaper).
- ignore.txt: A text file containing a set of words that will be ignored if seen in the file "input.txt".
- dictionary.txt: A file containing numeric word representations for each word in a dictionary. For example, int the following

```
num_words=5, vector_size=3
student     -0.2  0.5   1.0
course      -1.0  0.2   0.7
car         -0.1  0.9   1.2
bus         -0.2  0.8   1.1
instructor  -0.3  0.6   0.9
```

  the first line defines the dimension of the dictionary (5 words in the given file) as well as the dimension of the vector representing each word (3 floating point numbers in this case).

  In a typical example, there will be thousands of words (10,000<=num_words<=300,000). Number of floating points to represent a word is usually in the low 100s (100<=vector_size<=400).

  Given two words w_i and w_j, and their vector representations v_i and v_j (for example, w_i="car", $v_i = [-0.1, 0.9, 1.2]$, w_j="bus", and $v_j = [-0.2, 0.8, 1.1]$), the dissimilarity between these two words are given by $|v_i - v_j|^2$ (e.g., sum of squared differences of the entries of the vectors).

Given these, you are to write the following functions:

- `int main(int argc, char *argv[])`: The main program asking the user to enter one or more words. Based on this input, the program will respond in the following manner:
  - If there is only one word entered, the program will output the number of times that word appears in the text. If it does not appear, the closest word in the dictionary and its occurrence will be output.
  - If there are more than one words entered, the program will print a histogram. Each line will have the word and its frequency on the input file. The occurrences are normalized so that a maximum number of characters (* for words with exact matches and + for

words with closest match) is limited to 20 characters. The output should have the scale information printed. If the frequency is less than or equal to 20, the scale should be 1 and not printed.

Sample outputs:

- The file includes the given (one) word:

```
Enter word(s): car
“car” appears in “input.txt” 2 times.
```

- The file does not include the given word but a close enough word is found. Use inverse of `dissimilarity` for closeness with a programmer defined threshold. Assuming truck and bus are close enough:

```
Enter word(s): truck
“truck” doesn’t appear in “input.txt” but “bus” appears 2 times.
```

- Assuming triangle has no close enough word in the dictionary:

```
Enter word(s): triangle
“triangle” doesn’t appear in “input.txt”.
```

- The file includes some of the given words but not all (assuming car and student appears 10 and 5 times, while bus (closest word car) appear:

```
Enter word(s): car student bus
“car”        **********
“student”    *****
“bus”        ++
```

- The similar situation as the previous one but with occurrences 40, 10 and 5 respectively.

```
Enter word(s): car student truck
Scale: 2
“car”        ********************
“student”    *****
“truck->bus” ++
```

- When there are no matches (triangle in the previous case):

```
Enter word(s): car student triangle
Scale: 2
“car”        ********************
“student”    *****
“triangle”   NO MATCHES
```

You can assume that the user can enter at most MAX_WORD_COUNT=100 words.

You can further assume that a word can be at most MAX_WORD_SIZE=12 characters.

- `int read_dict(const char * file_name, char dict[][MAX_WORD_SIZE])`: Read the given dictionary file (as described above) and return the number of words read. A negative return value indicates an error. The output `dict` is an array of strings that ends with the sentinel value “-”.
- `int read_text(const char * text_file, const char * ignore_file, char words[][MAX_WORD_SIZE])`: Read the given text file (as described above) return the number of words read. A negative return value indicates an error. The output `words` is an array of strings that ends with the sentinel value “-”. Use the following rules to read the input text file:

- Ignore white-spaces and the punctuation characters (‘.’, ‘:’, ‘;’, ‘,’, ‘\’’, ‘”’, etc.)
- Ignore any words that are in the “ignore_file”. Use exact string matching.

- `double dissimilarity(char * w1, char * w2, char dict[][MAX_WORD_SIZE], float threshold, int * )`: Calculate the dissimilarity between two given words using the formula given above. The inverse of this will be used for finding the closest word to a given word when it is not in the dictionary.
- `int histogram(const char words[][MAX_WORD_SIZE], const int occurrences[], const char hist[][MAX_WORD_SIZE+5+20])`: Given a list of words (words) and their occurrences in the input file (occurrences), this function creates a line for each word illustrating the histogram as described above. Note that the words and histogram arrays are sentinel controlled (also described above). The function returns the scale used to draw the histogram.

Of course, these functions should be used in your program whenever possible and needed. You should not have any code replicating the functionalities of these functions.

We have provided examples for the files to be processed. Test your implementation on these files. Beware that we will test your code on a different set of files (where the number of words and vector sizes will change).

General Rules:

1. We do not give you any function prototypes. We expect that you are experienced enough to understand when to use methods and name them. These will also be graded.
2. The program must be developed on given version of OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won’t be tolerated.
3. Note that if any part of your program is not working as expected, then you can get zero from the related part, even if it is working partially.
4. Zip your homework files before uploading them to MS Teams. The zip file must contain the C file with your solution and screenshots of the valid outputs of the program.
5. Put all the output screens in a pdf file by running your code.
6. You can ask any question about the homework by sending an email to zbilici@gtu.edu.tr or by using the homework channel on the MS Teams page of the course.

Assignment Specific Rules:

1. Do not use any functions from string.h other than strcmp.
2. Make sure to include appropriate comments and variable names in your code to make it easy to understand.