Assignment # 3

Due Date: **Monday 23 Nov, 2020, 11:59 PM**

On Google Classroom

Objective: **File Input/Output, 1D/2D integer/character arrays**

NOTES:

1. Understanding the question is part of assignment, so do not ask anyone else to understand the
2. problem for yourself and tell you the solution. Do it yourself.
3. The name of cpp files should be l20XXXX_q_no.cpp where XXXX is your roll number.
4. Make a .zip file which will contain all cpp files and name that zip file as l20XXXX.zip and submit this zip file.
5. Always check the boundary values [Recall Problem solving].
6. Always use proper variable names and proper indentation for readability.
7. Always use comments to elaborate the code.
   you can comment in c++ through:
   // This is single line comment
   /* This is
      Multi line
      Comment
   */
8. Always follow the output and input format.
9. **DON'T involve in PLAGIARISM.**

You are given a file 'termids.txt':

- termids.txt– A file mapping a term found during tokenization to a unique integer, its TERMID. Each line should be formatted with a TERMID and token separated by a tab, as follows:
- **46**\t**array**.
- \t stands for tab space.

a file 'docids.txt':

- docids.txt– A file mapping a document to a unique integer, its DOCID. Each line should be formatted with a DOCID and document name separated by a tab, as follows:
- **8**\t**problems#7.txt**.
- \t stands for tab space.

a file 'doc_index.txt'.

- doc_index.txt– A forward index containing the position of each term in each file. One line of this file contains all the positions of a single token in a single document. Each line should contain a DOCID, a TERMID and a list of all the

positions of that term in that document (the first term has position 1, the second has position 2, etc.).

- The DOCID, TERMID and positions should be separated by tabs, as follows:
- **1**\t**39**\t2\t**7**\t**367**\t**420**\t**449**\t**586**\t**597****641**\t**666**\t**668**\t**783**
- **docID**\t**termID**\t**1st_position**\t**2nd_position**\t**3rd_position**\t**4th_position**\t**up to so on**
- \t stands for tab space.

PART-I:
[Filing → Make all possible terms with their IDs]:

Make a global 2D character array of rows = 1000 and cols = 20, name it as **unique_terms**.
Make a global 2D character array of rows = 50 and cols = 30, name it as **doc_list**.

1. Write a function **void read_docs_list();** which reads the "'docids.txt" and stores all the documents in the **doc_list** array. If the documents are less than 50, place null (\0) in the remaining indices.
2. Write a function **void read_unique_terms();** which reads the "'termids.txt" and stores all the terms in the **unique_terms** array. If the terms are less than 1000, place null (\0) in the remaining indices.

PART-II:
[1D Array → Positions of a term/token in a document]:
Make a global integer array of size 100 and name it as **term_document_frequency**.
Let's say you have a document named as "chapter0.txt" and the ID of the document is 0. So, in documentID=0, there are many terms.
The main function takes input a term from the user, calls the function **get_term_ID** to get the ID of this term and then calls the function **find_term_in_document,** which fills the global array. Now just call the function **printArray** to print the positions of the term in the document.
Functions you need to implement:

1. Write a function **void print1DArray(int array[], int size);** which takes an array and its size as parameters. The function prints the valid content of the array on the console. The valid content is "Non negative values". The invalid content will be -1.
2. Write a function **int get_term_ID(char term []);** that receives a term (char array) as a parameter and returns the **term_ID** of that term. This function converts the term into lower case and checks whether it is in stopwords list or not. If the term is valid then the function opens the "termids.txt" and finds the term_ID from the document.
3. Write a function **void find_term_in_document(int term_ID, int documentID = 0);** This function opens the "doc_index.txt", finds the **term_ID** in the **document_ID** and stores the positions of the term_ID in **term_document_frequency** array.

PART-III:
[2D Array → Positions of a term/token in a document]:

Make a global 2D integer array of rows = 10 and cols = 100, name it as **term_frequency_in_all_documents**.
Let's say you have a document named as "problems#0.txt" and the ID of the document is 0. So, in documentID=0, there are many terms. As there are many documents, but we consider we have at max 10 documents.
The main function takes input a term from the user, calls the function get_term_ID to get the ID of this term and then calls the function **find_term_in_document** for **ten times,** which fills the global array **term_frequency_in_all_documents**. Now just call the function **printArray** to print the positions of the term in the document.
Functions you need to implement:

1. Write a function **void print2DArray(int array[][100], int rows, int cols);** which takes an array and its rows and cols as parameters.
   The function calls the **print1DArray(array[i], frequency).** where the frequency is the count of valid values.

Main Function:

1. Call the function **read_docs_list()**
2. Call the function **read_unique_terms()**
3. Take a term from the user.
   a. Get the term_ID
   b. **Find the term frequency in document 0**
      Print all the positions through print1DArray
   c. **Find the term frequency in ten documents**
      Print all the positions through print2DArray