Brief Report on Assignment 1:

The assignment consists of two files: **assign1.py**(invert.py) and **test.py**. I used python programming language for the implementation of the assignment. The first file as stated in the outline is in charge of the preprocessing and creating the data structures of the document corpus and saving them as files.

- Step 1: we read the cacm.all file and save each document and its relevant details in a Dictionary. The structure (keys) of this dictionary is like below:

  Full_list ={ID, Abstract, Date, Authors}

- Step 2: we start creating the posting list. The data structure used for the posting list is **a list of lists**. Each term in the corpus has an index or number assigned to it and that same number is its place in the posting list (sorted according to the alphabet). Now each entry of the posting list is also a list, which is the list of the postings of that term. Each posting Is an object of class “posting” that has the following instances:

  Posting =[positionsList, termFrequency, documentID]

  For example, if the first entry of the posting list is for the word ‘binary’. The first entry contains a list of all the postings of the term “binary”, which is every document that has this word in it and it’s TF and positions.

- Step 3: In this step, we create the dictionary of document frequency of terms. Which is very simple now the DF of each word is the **length of its list in the posting list**!

- Step 4: we have to save the dictionary and posting list in files. I save the dictionary as a **.csv** file and the posting list as a **.txt** file. For the dictionary, we have this format: (term, DF). we start from the top of the posting list and save each posting in one line with this format: [TF,pose1, pose2,...,poseN,docID]. With this organization when we are reading the posting_list.txt file we know how to parse each line into meaningful variables.

Here are the steps of the test.py algorithm:

- Step 1: We ask the user for an input query, we ask if they want the query to be stemmed, if they want stop word removal in the document corpus and if they want stemming in the document corpus.

- Step 2: we read the cacm.all file, just the way we did with the first step of the assign1.py algorithm.

- Step 3: we read the dictionary and with the value corresponding to each key before our input query we calculate from-to which line of the posting list we have the documents of the input query.

- Step 4: we present the user with the documents in which the query was present and we print a brief part of the document with the query being highlighted. We also print TF, document ID, TItle, and in the end, we print the calculated execution time.

Manual on how to run the algorithm:

First, we should make sure that the cacm.all file is present in the directory of the two python files. Then we run the test.py script.

1. First, the program is gonna ask the user for the input query.
2. It asks the user if they want the query stemmed.
3. It asks if we want the stop-word removal component to work (enter ‘y’ or ‘n’).
4. It asks if we want the stemming component to work (enter ‘y’ or ‘n’).

```
In [168]: runfile('D:/Ryerson/fall 2019/CPS 842/assignment 1/test.py',
wdir='D:/Ryerson/fall 2019/CPS 842/assignment 1')
Reloaded modules: assign1

Enter a word: domain

Do you want your query to be stemmed(y/n): n

Do you want stop word removal(y/n): y

Do you want stemming(y/n): n
```

Here are some examples of what output the program is gonna provide us with:
In this case, we entered the term ‘domain’. We did not want the query to be stemmed. We wanted stop-word removal but not stemming:

```
Enter a word: domain

Do you want your query to be stemmed(y/n): n

Do you want stop word removal(y/n): y

Do you want stemming(y/n): n
section 2...
first part out
section 3...
section 4...
domain:
This term was seen in 4 documents.
----------------------------------------

Document number 1346 :
This term is seen in this document 1 times.
Title:
application process equalization

. however , few results known domain show efficient algorithm can obtain
positions : 22
----------------------------------------

Document number 1692 :
This term is seen in this document 1 times.
Title:
numerical solution thin plate heat transfer problem

, boundary value problem , thin domain , successive overrelaxation ( sor
positions : 24
----------------------------------------

Document number 2217 :
This term is seen in this document 2 times.
Title:
experiments automatic learning multipurpose heuristic program

improvements solutions problems which were same domain those used
collection learning data
positions : 134,165
----------------------------------------

Document number 2956 :
This term is seen in this document 1 times.
Title:
some ideas data types high-level languages

1 n notation ) , use domain array data type , need
positions : 45
----------------------------------------

execution time in seconds: 65.8359999657
```

In the second example, we entered the word binary and we wanted the query to be stemmed, we didn't want stop-word removal but we did want to stem the documents (because the output was really big, I only show the beginning and the end of it):

```
Enter a word: binary

Do you want your query to be stemmed(y/n): y

Do you want stop word removal(y/n): n

Do you want stemming(y/n): y
section 2...
first part out
section 3...
section 4...
binari:
This term was seen in 65 documents.
----------------------------------------

Document number 23 :
This term is seen in this document 1 times.
Title :
binari and truth-funct oper on

 binari and truth-funct oper on
positions : 0
----------------------------------------

Document number 33 :
This term is seen in this document 1 times.
Title :
a program binari counter for the ibm type 650 calcul

a program binari counter for the ibm type
positions : 2
----------------------------------------

Document number 40 :
This term is seen in this document 10 times.
Title:
finger or fist ? ( the choic of decim or binari represent )

( the choic of decim or binari represent )
the binari number system offer mani advantag
positions : 10,14,36,44,62,73,134,151,169,173
----------------------------------------

Document number 63 :
This term is seen in this document 3 times.
Title:
octal diagram of binari concept and

octal diagram of binari concept and
paper date back the genesi of binari concept circa 5000 year ago
positions : 3,13,40
----------------------------------------

Document number 76 :
```

```
Document number 3095 :
This term is seen in this document 2 times.
Title:
a select travers algorithm for binari search tree

a select travers algorithm for binari search tree
of select data item from a binari search tree accord to a
positions : 5,16
-----------------------------------------

Document number 3096 :
This term is seen in this document 1 times.
Title:
an optim method for delet in one-sid height-balanc tree

a one-sid height-balanc tree is a binari tree in which everi node
positions : 15
-----------------------------------------

Document number 3115 :
This term is seen in this document 5 times.
Title:
orderli enumer of nonsingular binari

orderli enumer of nonsingular binari
nonsingular binari matric of order n ,
positions : 4,6,54,71,130
-----------------------------------------

Document number 3143 :
This term is seen in this document 1 times.
Title:
reason about array

by inform descript of algorithm for binari search and merg , and
positions : 69
-----------------------------------------

Document number 3163 :
This term is seen in this document 1 times.
Title:
an optim insert algorithm for one-sid

a one-sid height-balanc ( oshb ) binari search tree is present .
positions : 19
-----------------------------------------

execution time in seconds: 121.053999901

In [170]:
```