Documentation for 1st part and 2nd Part (using grep and python)

-Hemanth Reddy Vennapusa
-15CS10051

First install perl by running the command -> `sudo apt install perl`
*****Preprocessing on documents -

```
perl -pi -e 's/\n/ /g' *
perl -pi -e 's/\r/ /g' *
```

the following two commands were used to replace all new lines and return carriage to space to help use grep
Document name format - 'GX[0-9]\{3\}\-[0-9]\{2\}\-[0-9]\{7,8\}'
if the query is 'W1 W2 W3 W4 W5' the following grep command can be issued to search in all documents and output the file names

```
grep W1 * -i | grep W2 -i | grep W3 -i | grep W4 -i | grep W5 -i | grep 'GX[0-9]\{3\}\-[0-9]\{2\}\-[0-9]\{7,8\}' -o
```

****The `output.txt,query.txt,alldocs(folder)'s` PATH is default been set to as in the programs folder
but if its different we can change it in the global variables benchm , queries ,loc in starting of the source code .
No of queries is also been defaulted to 82 in the global variable noquer.

The program first opens the query.txt file then for each query in the file it executes the grep command and appends the output to the
results.txt file and also counts the no of results for each query and total no of results for all queries.
It also calculates the time of execution for each query and stores it.
Then it opens the output.txt and for each query it maintains a set containing the relevant document ids

Then it opens the `results.txt` and compares the output and the results to calculate the precision and recall
The result is printed in the `performance.txt` file

To run the program make sure the directory of program is as follows

```
current dir /
```

-> searcher.cpp
-> makefile
-> alldocs/ (folder)
-> output.txt
-> query.txt
-> presenter.cpp (this one just formats the output of grep to results.txt)

if not change the path variables as stated above

After above step open a terminal and type -> `make clean`
-> `make run`

Using Postings List Matching -

******The preprocessing used in the grep program for the input files is also necessary for this program**

Unlike above the postings list method has only one program inverted_index.py
First it loops all files and adds each word in it to dictionary and appending the filename to the corresponding list of the word.
Then for each query , first it stemmatizes the words then it merges all the posting lists of the words in the query ,then it appends the document name to the list corresponding to the query id in a dictionary.
I unit tested all these parts Successfully
when i run the program by indexing only a few files all the program is working correctly but when it tries to index all the input files then it is taking a lot of time(lot) .When i put debugging statements in the indexing loop , it shows only 20 ~ 25 files where indexed per minute , so i couldn't run the program completely (I completed the program on wednesday but i didn't test it for all documents untill today.
I even implemented a threaded version with 2 new threads for indexing , yet the speed stays the same.
If you want to check make sure you have the nltk tools and also Snowball dataset with it
For nltk - `sudo pip install -U nltk`
Then make sure the current directory is as follows

```
current dir /
 -> inverted_index.py
 -> inverted_indexmt.py (only if you want to run the multi threaded
version)
 -> alldocs/ (folder)
 -> output.txt
 -> query.txt
```

If any of the above is in different folder you can specify it in the following global variables in the program starting.
path = 'alldocs/'
output = 'output.txt'
queries = 'query.txt'
We can run it by typing `python inverted_index.py > performance2.txt`

You can replace performance2.txt by any other name