LOAD FACTOR	HASH FUNCTION	COLLISION HANDLING	COLLISION COUNT	INDEXING TIME	AVG. SEARCH TIME	MIN. SEARCH TIME	MAX SEARCH TIME
α=50%	SSF	LP	18123852	33084ms	97580630ns	56950349ns	43948220ns
		DH	4530963	7877ms	19516150ns	19191200ns	19841100ns
	PAF	LP	48423037	55851ms	862719000ns	56942450ns	89462310ns
		DH	12807679	11190ms	19254200ns	17687300ns	20821100ns
α=80%	SSF	LP	27754755	39980ms	76305490ns	54239500ns	54998400ns
		DH	5550951	7996ms	18806550ns	18355500ns	19257600ns
	PAF	LP	63756916	42560ms	104653945ns	43595450ns	709565630ns
		DH	15939229	10912ms	20923800ns	17859700ns	23987900ns

Warning : Linear Probing takes significantly more time than Double Hashing but it gives correct results .

Warning : Code uploaded as **ready** for **Double Hashing (PAF , %80)** (Linear Probing parts in comment lines) in HashTable.java class

Comment Lines for :

DOUBLE HASHING - LINEAR PROBING - PAF –SSF AND LOAD FACTOR choosing

- Inside of HashTable.java class

These comment lines should be open for the required operation preference.

For **example** : For double hashing remove the **comment lines** given below. U should also have comment lines on Linear Probing parts .

//

DOUBLE HASHING comment lines

- **HashTable.java class**

69-77 and 100-113 and 213-226

69 – 77 region choose one of the if condition **50** or **80 (** load factor**)** other one should be closed (**Close for Linear**)

```
67          //WHEN DOUBLE HASHING USING, OPEN THIS COMMAND LINES AS YOUR LOAD FACTOR PREFERENCE
68
69          // Load factors to %50 resizing of hash table
70  //        if (count>=(table.length*0.5)){
71  //            reSizeDouble();
72  //        }
73          // Load factors to %80 resizing of hash table
74          if (count>=(table.length*0.8)){
75              reSizeDouble();
76          }
77
```

100 –113 region should be **open** for **Double Hashing (close for Linear P.)**

```
98          //DOUBLE HASHING
99
100         if (table[hash]!=null){
101             int secondHash;
102             int j=0;
103             while (table[hash]!=null){
104                 secondHash=13-(sHashFunction(key)%13);//I cant find anything to put here
105                 if (secondHash==0)
106                     secondHash=1;
107                 hash=(hash+(j*secondHash))%table.length;
108                 j++;
109                 collisionCount++;
110             }
111             table[hash]=entryToInsert;
112             count++;
113         }
```

213-226 region should be open for Double Hashing (close for Linear P.)

```
211         //Search for double hashing
212
213         int j=0;
214         int index=pHashFunction(key);
215         //int index=sHashFunction(key);
216         int actualIndex=13-(sHashFunction(key)%13);
217         while (table[index]!=null&&!table[index].getKey().toString().equals(key)){
218             if (actualIndex==0)
219                 actualIndex=1;
220             index=(index+j*actualIndex)%table.length;
221             j++;
222         }
223         if (table[index]!=null&&table[index].getKey().toString().equals(key))
224             return index;
225         else
226             return -1;
```

//

LINEAR PROBING Comment Lines

- **HashTable.java class**

56-66 and 86-97 and 228-241 comment lines should be open

56-66 region choose one of the if condition **50** or **80 (**load factor**)** other one should be **closed**

(close for double hashing)

```
/*
//WHEN LINEAR PROBING USING, OPEN THIS COMMAND LINES (57-66 region choose one of load if condition  AS YOUR LOAD FACTOR PREFERENCE
// Load factors to %50 resizing of hash table
if (count>=(table.length*0.5)){
    reSize();
}
// Load factors to %80 resizing of hash table
//  if (count>=(table.length*0.8)){
//      reSize();
//  }
*/
```

86-97 region should be **open** for **Linear Probing** (**close for double hashing**)

```
//LINEAR PROBING
/*
if (table[hash]!=null){
    int px=0;
    while (table[hash]!=null) {
        hash = (pHashFunction(key) + px) % table.length;
        px++;
        collisionCount++;
    }
    table[hash]=entryToInsert;
    count++;
}
*/
```

228-241 region should be **open** for **Linear Probing** (**close for double hashing**)

```
/*
//Search for LINEAR PROBING
int index=pHashFunction(key);
//int index=sHashFunction(key);
int i=0;
while (table[index]!=null&&!table[index].getKey().toString().equals(key)) {
    index=(pHashFunction(key)+i)%table.length;
    i++;
}
if (table[index]!=null&&table[index].getKey().toString().equals(key))
    return index;
else
    return -1;
*/
```

//

PAF comment lines

79 single line these will be open even if its DH or LP

```
int hash = pHashFunction(key);// Calculate hash value
```

pHash

143 single line this will be open even if its DH or LP

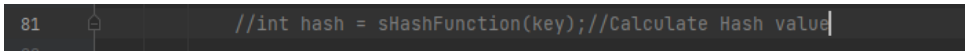
180 single line this will be open even if its DH or LP

214 single line (open if Double Hashing and PAF)

230 single line (open if Double Hashing and PAF)

SSF comment lines

81 single line   these will be open even if its DH or LP



sHASH

144 single line   these will be open even if its DH or LP

181 single line   these will be open even if its DH or LP

215 single line (open if Double Hashing and SSF)

231 single line (open if Double Hashing and SSF)

//

Data Structures

LinkedList :

I used Linkedlist as a value in the hashmap because the same word can be found in more than one file, there is no certain limit, I used it because we can add words as they come with linkedlist.

ArrayList : I used the arraylist because it has the removeAll function so i can use that function to delete stop words and delimeters from the words in the files we read from the txts.

HashTable: For storing cleaned data as keys and required value data's and for searching operations.