Project Milestone-3

Group: CS311S20PID-G58

Group Members:

1-Muhammad Raza (2018-CS-60)

2-Ghamees Ul-Mohsin (2018-CS-99)

Project Name: Plagiarism Checker Tool

Language: C# (Desktop Application)

Algorithm: Rabin-Karp (with some of our modification + using hash tables)

Algorithm Analysis is given on below pages

Pseudo Code

string ReadFile()

```
{
        int N = total.number.of.files ;          //Number of Files
        string file_array [N] = { } ;            //array for storing read file
        for ( int i = 0 ; i-to-N ; i++ )     //Reading files from 1-to-N.files
        {
                file_array[i] = file.i ;
        }
        return file_array[];
}
```

int GenerateHashTables(string array[N])

```
{

        HT_array[N] ;                                   //array for storing hashtables
        for( int i=0  ;  i-to-file_array  ; i++ )       //for loop upto  N-files
        {
                hashtable HT(i) = new hashtable() ;     //creating hashtables
                HT_array[i] = HT(i) ;
        }
        return HT_array[] ;
}
```

string MatchingData(int HT_array)

```
{
int var = 0 ;                           //variable for indexing matching array
string MatchedArray[] ;                 //string array for storing matched words
string miniTable = HT.Smallest  ;
                /* using smallest file hashtable to compare the matching text
                of it with every other file hash table */
for(int x=1  ;  x-to-last.file  ; x++ )         //loop goes upto all files
        {
                for( int i=0 ; i-to-end.of.file(x) ; i++ )
```

```
                                        /*loop goes upto all words of file x */
                {

                        if( word[x] == word.of.hashtable[i] )
                                {
                                MatchedArray[var] = word[x] ;
                                                /*if the word matches then it is
                                                stored in the matchedarray  */
                                var++ ;
                                }
                        else
                                continue ;
                }
        }
        return MatchedArray[] ;
}
```

Algorithm Correctness and Analysis

string ReadFile()

1.Initialization:

We are using a string function to give an input file/s to our code.

2.Maintenance:

If the invariant holds at the beginning of an arbitrary loop iteration, then it must also hold at the end of that iteration. It means that we simply load that number of file which are present in our folder

3. Termination:

The loop always terminates.

4. Correctness:

Whenever the loop invariant and the loop exit condition both hold, then P must hold because every times we load files the loop meets its ends.

int GenerateHashTables(string array[N])

1.Initialization:

We are giving an array of words to our function.It generates hash tables according to by creating their hash functions .

2. Maintenance:

The loop invariant varies according to our need of the input file which is to be saved in hash table.

3 . Termination:

After adding every word it checks the loop invariant and if terminates at the end.

4. Correctness:

It holds correct because every time we create an hashtable it return us the hashtable and all the words processed.

string MatchingData(int HT_array)

Initialization: There is an array for the matching content of the files data (which should be string data)

2. Maintenance: If the file exists and its not empty then loop must check the matching data. Otherwise move to next file.

3. Termination: Nested loop is terminated after examining the all files through it.

4. Correctness: Final array of the matching content should be returned and it holds the common data among all the files.