ASSIGNMENT-1
DATA STRUCTURES
(CSU33D05)

SUBMITTED TO:
Dr. Mélanie Bouroche

SUBMITTED BY:
PRACHI SINGHROHA
(Student ID: 21355131)

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Declaration concerning plagiarism

I have read and I understand the plagiarism provisions in the General Regulations of the *University Calendar* for the current year, found at http://www.tcd.ie/calendar

I have completed the Online Tutorial in avoiding plagiarism 'Ready, Steady, Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write

STUDENT NUMBER: 21355131

SIGNED:

DATE: 13.10.2021

Task 1:

I have used the skeleton given on the blackboard. I updates the functions there, changed the value of *array_size* and *max_string_size*.

My approach is the same as in the skeleton. It retrieves the data from the file and stores it in the array. It then indexes it on the basis of linear probing in the insert function with the use of hash function given in the skeleton. The main function do the following jobs:

1. Prints the value of collision, capacity, numbers and load
2. Take a string as input and print the number of times the string occurred in data using *NumberOfOccurences* function.

Task 2:

I used the hash function given on page 118 in the book *The C Programming Language*.

Collisions in task 1: **32**
Collisions in task 2: **15**

Clearly the hash function I've used here is better than the one used in task 1 because it doesn't take string character order in consideration. To make it more clear lets call the following functions.
hash_function("ct");
hash_function("tc");
Both of them will return the same value which does not happen with the hash function used in task2, it will return two different values for these strings.

Task 3:

The approach for this task is same as the other tasks. It just indexes with double hashing instead of linear probing by sing two hash functions.

It also reduces the number of collisions because there is no clustering in double hashing.

The hash function given in the assignment is good enough because:

1. It never evaluates to 0.
2. Probes all the values of the cell.

3. Less uniform distribution.
4. Prime number added is added to the other hash function to create this one.

References:

1. *Skeleton code provided*
2. *Jacob Sorber: Understanding and Implementing Hash Tables in C*. (Youtube)
3. GeeksForGeeks
4. *The C Programming Language*
5. https://cseweb.ucsd.edu//~kube/cls/100/Lectures/lec16/lec16-23.html