3D5A: Data Structures
Assignment 1: Hash Tables

Rahul Seth
17302557

Task 1

Our objective for the first task was to design a hash table to store the frequency of Surnames given to us in the CSV file. We had to design the program in such a way that the user should be able to check the frequency of any name available in the list. We had to display the number of collision, number of terms in table and the load factor for the hash table. I made use of various functions in my code to break it down for simplification. I used functions for calculating load factor, frequency of names, storing data in the hash table and for printing the output.
The task was started by designing a CSV parser to read in the file given to us.
I made use of the function *getdata_csv* to store in the data from the csv file into the hash table. There was a separate function to calculate the frequency of each name which was called inside the hash table function. I also made use of built in C functions like "strcmp" and "strcpy" throughout the program. I included some global variables to calculate the hash table data like the number of collisions, load factor and the number of terms.

```
int hash1(char* s){
   int hash = 0;
   while(*s){
      hash = (hash + *s);
      s++;
    }
   return (hash % ARRAY_SIZE)
}
```

Listing 1: Hash 1 Function from Task 1

Task 2

Our objective for Task 2 was to create another hash function to reduce the number of collisions from the previous task.
Following is the hash function used for Task 2 called "hash 2" :

```
Int hash2(char* s){
     Int hash=0;
     Int i=0;
     For(i=0;  s[i] != '\0'; i++){
           hash = hash + (*s + (i+1));
    }
  return (hash % ARRAY_SIZE);
}
```

Listing 2: Hash 2 Function from Task 2

In the above function, I am running a *for loop* inside the function where I am adding an iteration counter to my hash variable. On incorporating the hash 2 function in our code for Task 1 it was noticed that the number of collisions decreased significantly as compared to the first task. This meant that more unique keys were returned by the hash 2 function over the hash 1 function thus, making the program more efficient and faster.

Task 3

Our objective for Task 3 was to introduce another hash function and use it with the code for Task 2 and to implement the concept of double hashing over linear probing. Following is a snippet of the hash 3 function used.

```
Int hash3(char *s, int i){
      int hash = 0;
      int f, g, k;
      while(*s){
         k = hash + *s;
         f = k % ARRAY_SIZE;
         g = 1 + (k % (ARRAY_SIZE-1));
         hash = (f + i * g) % ARRAY_SIZE;
         s++;
     }
   return hash;
}
```

Listing 3: Hash 3 Function from task 3

In this Hash function, *" i "* is the iterations counter while *" g "* is the auxiliary hash function used instead of the linear probing strategy. In linear probing we would have had *hash = (f + i) % ARRAY_SIZE*.
There were a few observations I made while implementing double hashing over linear probing:

1. Double hashing reduces clustering
2. Linear probing is faster than double hashing

Task 4

Our aim for the last task was to store a list of surnames in the hash table. We were supposed implement a linked list with a hash function for this task.
I made use of struct to store all the variables in the given csv file and use pointers to call them and store information in the list. Since, pointers are essential for the implementations of linked list I made use of the in-built malloc function to assign memory so the program does not overload.

References

1. https://stackoverflow.com/questions/30136328/how-to-search-using-double-hash-in-c
2. https://www.geeksforgeeks.org/double-hashing/
3. https://softwareengineering.stackexchange.com/questions/278459/what-are-the-advantages-of-linear-probing-over-separate-chaining-or-vice-versa-w
4. https://www.geeksforgeeks.org/hashing-set-3-open-addressing/
5. https://stackoverflow.com/questions/14534431/how-to-reduce-hash-collisions
6. https://stackoverflow.com/questions/59513175/create-hash-table-with-linked-list-in-c
7. 3D5A: Data Structures & Algorithms lecture notes