Part A. Questions on Hash Tables

1 &2.

Briefly, hash functions use some generic function to generate an index for the data. Hashing can be used for several purposes. Some of them can be listed like that:
It can be used to compare large amounts of data. If we create the hashes for the data, it makes easy to compare the data with the help of comparing hashes.

It is also used for indexing.If we want to find our data faster than normal methods we can use hashing again. In this case, first we should calculate the hash of the data then we can directly go to the record by using related hash record

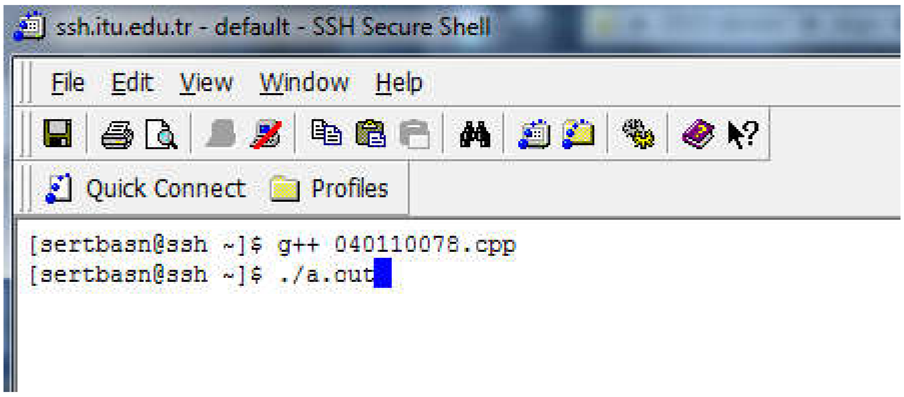
Furthermore,

1. Cryptographic applications
2. Finding duplicate records
3. Finding similar records
4. Finding similar substrings
5. Geometric hashing

Part B. Implementation

In this project I just tried to implement linear probing algorithm. Example compile results are given below.

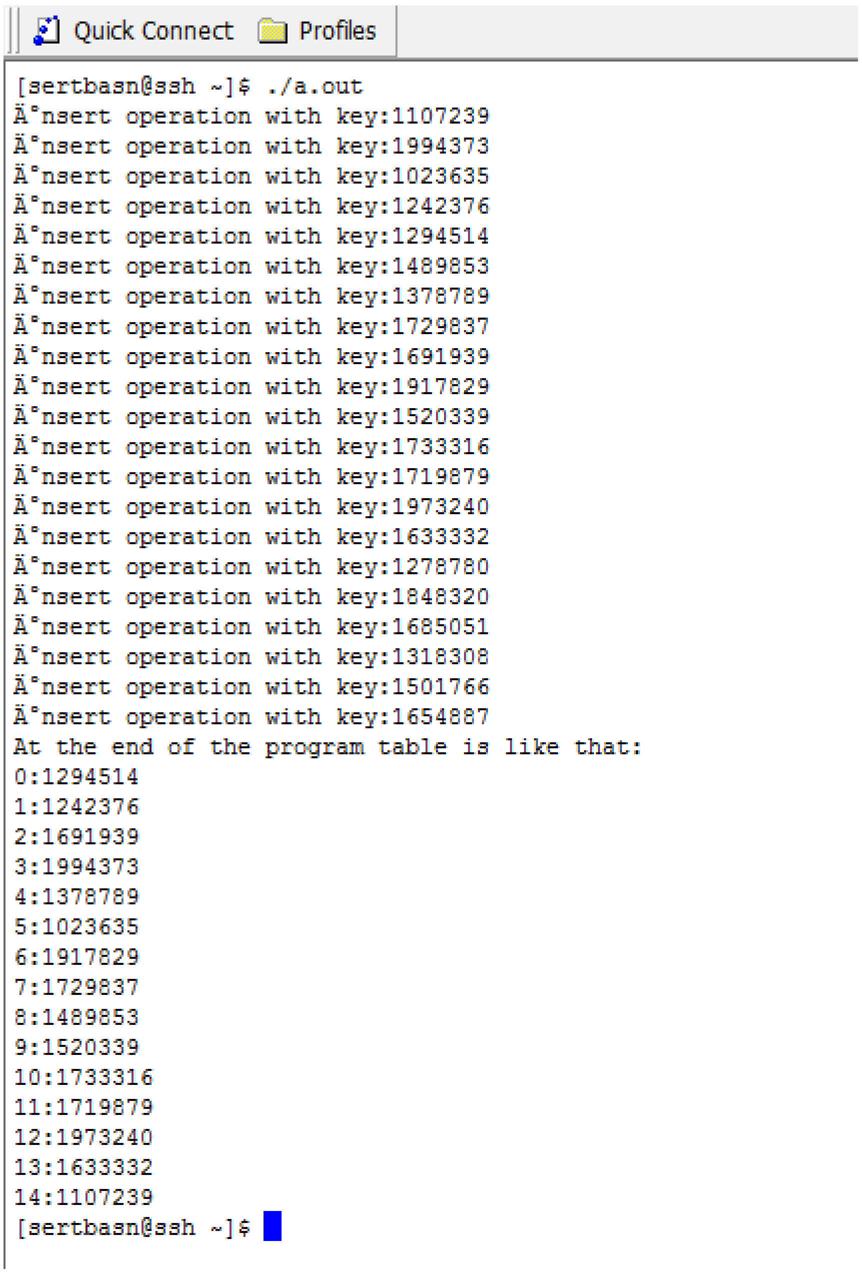
Also all code is compiled and run over SSH platform.


ssh.itu.edu.tr - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

```
[sertbasn@ssh ~]$ g++ 040110078.cpp
[sertbasn@ssh ~]$ ./a.out
```

Example of implementation:

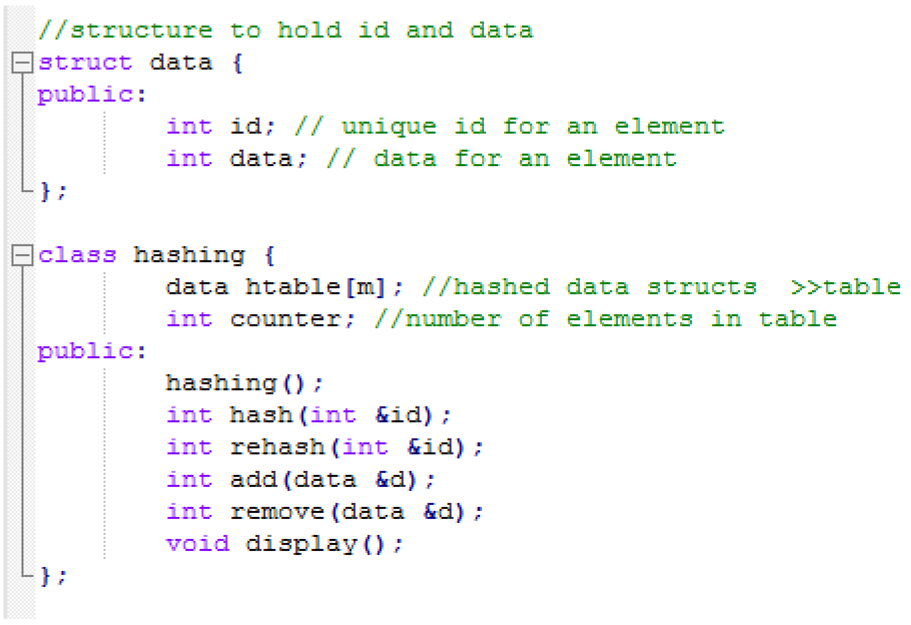
First 30 lines of the file is read and m=20 sized linear hashing results are given in the below picture in order to show its operation of the code:



```
[sertbasn@ssh ~]$ ./a.out
Ä°nsert operation with key:1107239
Ä°nsert operation with key:1994373
Ä°nsert operation with key:1023635
Ä°nsert operation with key:1242376
Ä°nsert operation with key:1294514
Ä°nsert operation with key:1489853
Ä°nsert operation with key:1378789
Ä°nsert operation with key:1729837
Ä°nsert operation with key:1691939
Ä°nsert operation with key:1917829
Ä°nsert operation with key:1520339
Ä°nsert operation with key:1733316
Ä°nsert operation with key:1719879
Ä°nsert operation with key:1973240
Ä°nsert operation with key:1633332
Ä°nsert operation with key:1278780
Ä°nsert operation with key:1848320
Ä°nsert operation with key:1685051
Ä°nsert operation with key:1318308
Ä°nsert operation with key:1501766
Ä°nsert operation with key:1654887
At the end of the program table is like that:
0:1294514
1:1242376
2:1691939
3:1994373
4:1378789
5:1023635
6:1917829
7:1729837
8:1489853
9:1520339
10:1733316
11:1719879
12:1973240
13:1633332
14:1107239
[sertbasn@ssh ~]$
```

Note: Above screen shot is from an example input file. It is just builded to show how the program works.

I have one struct and one class to implement the algorithm



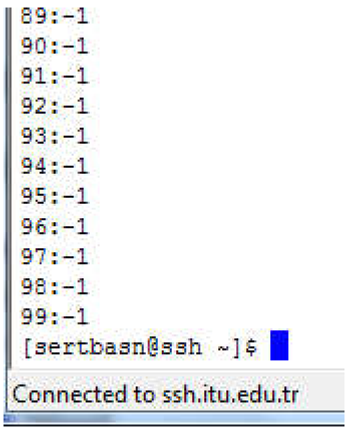
```
//structure to hold id and data
struct data {
public:
        int id; // unique id for an element
        int data; // data for an element
};

class hashing {
        data htable[m]; //hashed data structs  >>table
        int counter; //number of elements in table
public:
        hashing();
        int hash(int &id);
        int rehash(int &id);
        int add(data &d);
        int remove(data &d);
        void display();
};
```

Note: rehash function is written in case of any collision

After defining the structure I created a table with given size m then initialized it with the default values.**(-1 if there is no any element is inserted)** After that, I just tried to implement given functions on it by using key values.

Note: If there is no any inserted element or if insert and delete operations are both performed its value should be -1 like an initial case. It is represented in below figure.



```
89:-1
90:-1
91:-1
92:-1
93:-1
94:-1
95:-1
96:-1
97:-1
98:-1
99:-1
[sertbasn@ssh ~]$
```

Connected to ssh.itu.edu.tr

a. Linear Probing

$$h(k,i) = (k+i) \bmod m \text{ and } i \in [0, m-1]$$

b. Double Hashing

$$h(k,i) = h_1(k) + i * h_2(k) \bmod m \text{ and } i \in [0, m-1]$$

$$h_1(k) = k \bmod m \quad \text{and} \quad h_2(k) = 1 + k \bmod (m-1)$$

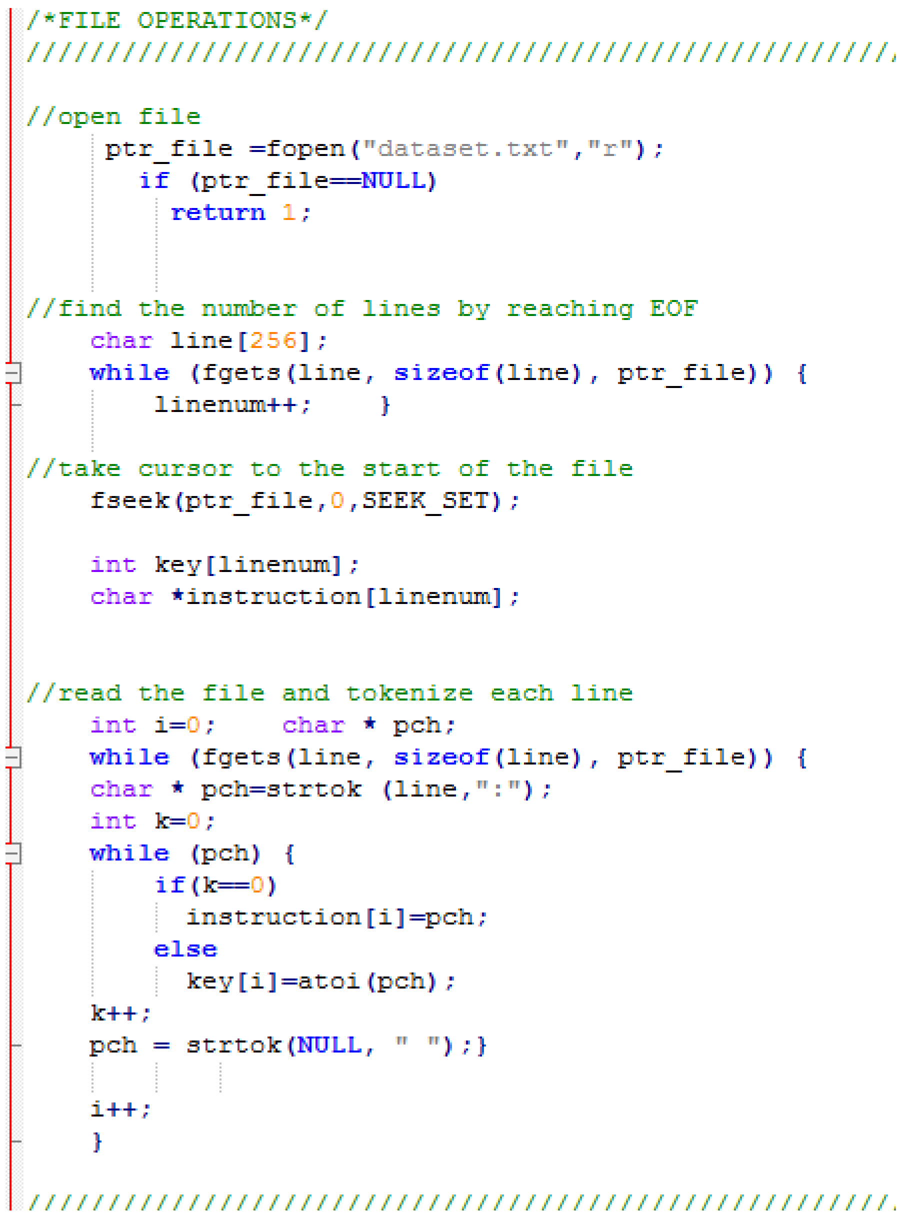
c. Quadratic Hashing

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m \text{ and } i \in [0, m-1]$$

$$h'(k) = k \bmod m$$

d. Reading 'dataset.txt' file

I read the file with below code.
Open the file in read mode (return with an error is it occurs)
Reach EOF to find number of lines
Define instruction and key arrays with specified size
Each line is hold with variable pch and its tokenized into two part and copied into related arrays.



```
/*FILE OPERATIONS*/
/////////////////////////////////////////////////////

//open file
     ptr_file =fopen("dataset.txt","r");
       if (ptr_file==NULL)
         return 1;


//find the number of lines by reaching EOF
    char line[256];
    while (fgets(line, sizeof(line), ptr_file)) {
        linenum++;     }

//take cursor to the start of the file
    fseek(ptr_file,0,SEEK_SET);

    int key[linenum];
    char *instruction[linenum];


//read the file and tokenize each line
    int i=0;     char * pch;
    while (fgets(line, sizeof(line), ptr_file)) {
    char * pch=strtok (line,":");
    int k=0;
    while (pch) {
        if(k==0)
          instruction[i]=pch;
        else
          key[i]=atoi(pch);
    k++;
    pch = strtok(NULL, " ");}

    i++;
    }

/////////////////////////////////////////////////////
```