

IS 2110 - Data Structures & Algorithms II
Group Assignment 01 – Hashing
Group 25

Name	Index number	E-mail address	Contribution
Gunathilaka W.H.M	18020275	2018is027@stu.ucsc.cmb.ac.lk	Question 3
Piyathilake S.A.V.S	18020585	2018is058@stu.ucsc.cmb.ac.lk	Question 1
Weerarathna T.M.P	18020917	2018is091@stu.ucsc.cmb.ac.lk	Question 2

Question 01

This phonebook program uses a **hash table** to store **names as keys** and phone numbers as values. **Chaining technique** has been used to avoid collisions. The hash table, represented as a linked list of linked lists (known as array of linked lists); which means each element of the main linked list (*linkhash*) has been assigned as another linked list.

```
linkhash[i] = new LinkedList();
```

The hash value of the key has been calculated via **polynomial accumulation and Horner's rule**. Depending upon the position in the word, it multiplies each letter by increasing power. We can decrease the chance of collisions by this.

```
class HashTable {

    private int pos=0;
    int size = 10;
    LinkedList[] linkhash = new LinkedList[size];

    //Creates hash table - Creates a linkedlist of
    public void hashTable(){
        for (int i=0; i<size ; i++){
            linkhash[i] = new LinkedList();//const
        }
    }
}
```

```
public int computeHash(String key, int val){
    int hash = 0;

    for (int i = 0; i < key.length(); i++) {
        hash = 31 * hash + key.charAt(i);
    }

    hash %= size;

    if (hash < 0) {
        hash += size;
    }

    this.pos = hash;
    put(key, val);
    return hash;
}
```

```
/*putting to hash table*/
public void put(String owner, int pno){
    for (int i=0; i<size ; i++){
        if(i==pos){
            if (linkhash[pos].isEmpty()){
                linkhash[i].addFirst(owner); //
                linkhash[i].add(pno); //Add at
            }else{
                pos++;
                put(owner, pno);
                break;
            }
        }
    }
}
```

The **key** calculated by the hash function will map to the **beginning of the linked lists** created for each element of *linkhash* using **addFirst()**. The data(phone numbers) will be stored at the end of the linked list.

```

/*search for the owner of the number*/
public void search(int num){
    int flag=0;
    for (int i=0; i<size ; i++){
        if (linkhash[i].contains(num)){
            System.out.println(linkhash[i].getFirst());
            flag=1;
        }
    }
    if(flag != 1){
        System.out.println(0);
    }
}
}

```

When looping through the linked list, if *linkhash[i]* contains the user given phone number, it prints the first element/key/name otherwise the program displays a 0.

Test Cases

[1]

Input : 997768
Output: Mother

[2]

Input : 112234
Output: 0

Following are the outputs for the given test cases.

```

E:\UGVLE\year_2\sem_2\IS2110 - Data Structures and Algorithms II\Assignments\final answers>javac PhoneBook.java
E:\UGVLE\year_2\sem_2\IS2110 - Data Structures and Algorithms II\Assignments\final answers>java PhoneBook
*****
Enter the Phone Number Below. Owner of the no will be displayed in the next line if it's in the directory
. Otherwise 0 will be displayed.
*****
997768
Mother
E:\UGVLE\year_2\sem_2\IS2110 - Data Structures and Algorithms II\Assignments\final answers>java PhoneBook
*****
Enter the Phone Number Below. Owner of the no will be displayed in the next line if it's in the directory
. Otherwise 0 will be displayed.
*****
112234
0

```

Question 02

```

import java.util.*;
import java.io.*;

class HashTable{//Create a class for hashtable-Initialize
    static int SIZE=250;int[] key;String[] value;int count=0;

    public HashTable(){//construct a hashtable
        key = new int[SIZE];
        value = new String[SIZE];
    }
}

```

In this case what we had to do is get inputs from the given list of words in the text file, and implement a hash table-based algorithm to determine all the groups of anagrams in the file. Here what I have done is first implemented a hash table by initializing keys and values of the table. declared The table size as 250. because the maximum words of the file is not exceed 150.

```

12 private int asciiVal(String word){ //get the total ascii values (int values) of words
13     int asciiVal=0;int i;
14     for(i=0;i<word.length();i++){
15         asciiVal=asciiVal+word.charAt(i);
16     }
17     return asciiVal;
18 }
19
20 private int hashFunction(int asciiVal){ //getting hash codes of words to facilitate hashing in hash tables
21     return(0*asciiVal) % SIZE ;
22 }
23

```

Then implemented a AsciiValue() function to take the Ascii values of the words according to the length and characters.

Then used a hashFunction() to take hash codes of words to facilitate hashing in the hash table.

After getting the ascii values of the words, insert them to the hash table by using the hash function and if null value detected set it to return. Used linear probing to position the values when the hash table is close to full.

Then implemented a printAnagrams() function. Here checks a word with all the rest of words using two for loops and if the ascii value of those words are the same, put them in to two arrays using toCharArray () and when the arrays are equal, print them in a row and then go to new line before get another same ascii values. Like that we can identify all anagrams in the text file and ignore random words.

```

24 public void insert(String words){ //insert list of words to the hash table
25     count++;
26     int asciiVal = asciiVal(words);
27     int hash = hashFunction(asciiVal);
28     int i = hash;
29     do{
30         if(key[i]==0){
31             key[i]=asciiVal;value[i]=words;
32             count++;
33             return;
34         }
35         i=(i+1)%SIZE;
36     }while (i != hash);
37 }
38
39 public void printAnagrams(){ //print listed words together
40     for(int i=0; i<SIZE; i++){
41         int temp = 0;int j;
42         if (key[i] != 0){
43             for(j=i+1; j<SIZE; j++){
44                 if(key[i] == key[j]){ //check if the values of two words are same
45                     char wordArr1[] = value[i].toCharArray();
46                     Arrays.sort(wordArr1);
47                     char wordArr2[] = value[j].toCharArray();
48                     Arrays.sort(wordArr2); //if same put them into these two arrays
49
50                     if(Arrays.equals(wordArr1, wordArr2)){ //then if those arrays are equal
51                         if(temp == 0){
52                             System.out.print(value[i] + " ");
53                         }
54                         System.out.print(value[j] + " "); //print them in a row
55                         key[j] = 0;
56                         temp++;
57                     }
58                 }
59             }
60         }
61         if(temp > 0){
62             System.out.println(); //put a space after one anagram set printed
63         }
64     }
65 }
66

```

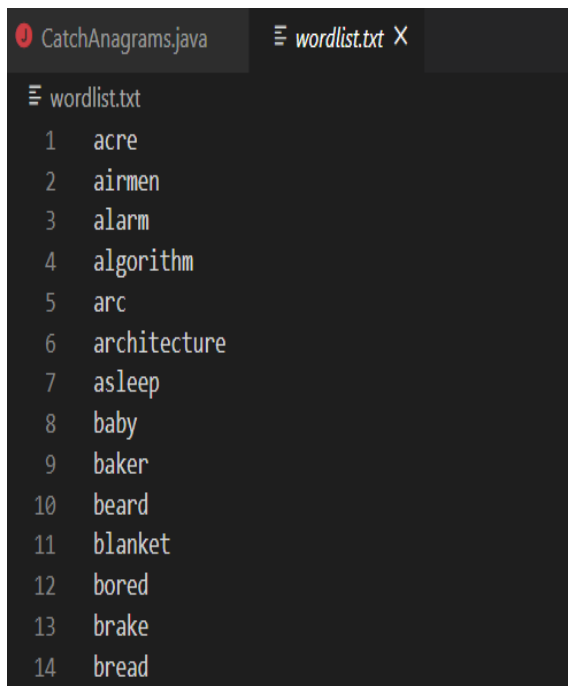
```

67 class CatchAnagrams{
68     public static void main(String[] args) throws FileNotFoundException{
69
70         HashTable word = new HashTable(); //call a hashtable function
71         File file = new File("wordlist.txt");
72         Scanner input = new Scanner(file); //input the wordlist.txt file
73
74         while(input.hasNext()){
75             word.insert(input.next()); //insert all the words
76         }
77         input.close();
78         word.printAnagrams(); // call print anagram function and display output
79     }
80 }
81

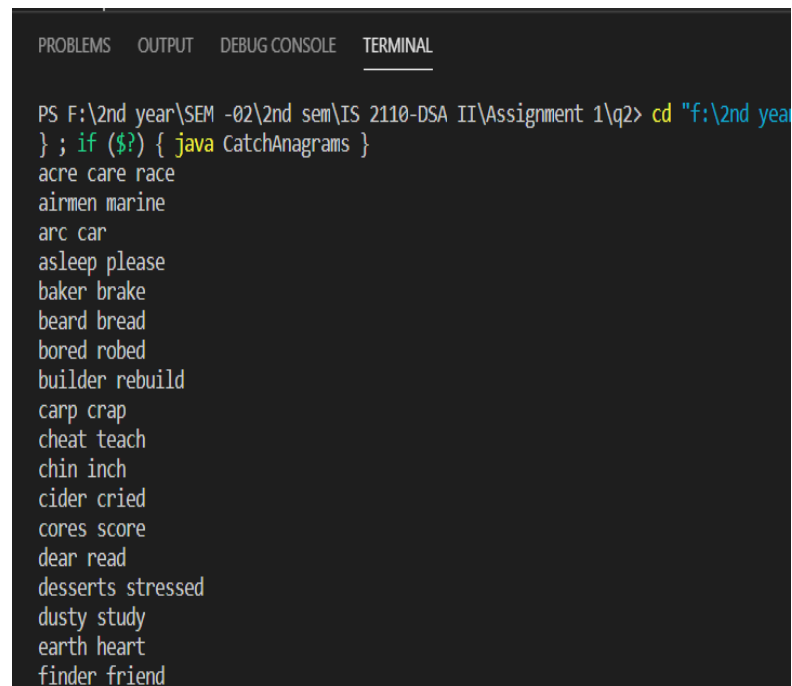
```

After implementing these functions, call them in CatchAnagrams() main method and get the "wordlist.txt" file as input. Then get the expected Output.

Inputs and Output:

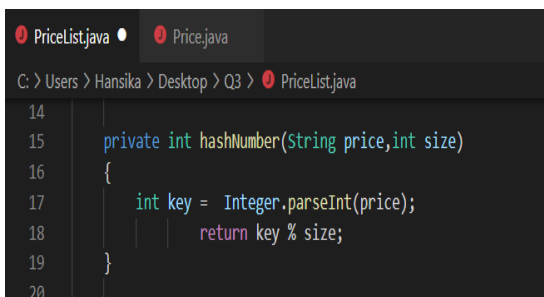


```
wordlist.txt
1 acre
2 airmen
3 alarm
4 algorithm
5 arc
6 architecture
7 asleep
8 baby
9 baker
10 beard
11 blanket
12 bored
13 brake
14 bread
```



```
PS F:\2nd year\SEM -02\2nd sem\IS 2110-DSA II\Assignment 1\q2> cd "f:\2nd year
"; if ($?) { java CatchAnagrams }
acre care race
airmen marine
arc car
asleep please
baker brake
beard bread
bored robed
builder rebuild
carp crap
cheat teach
chin inch
cider cried
cores score
dear read
desserts stressed
dusty study
earth heart
finder friend
```

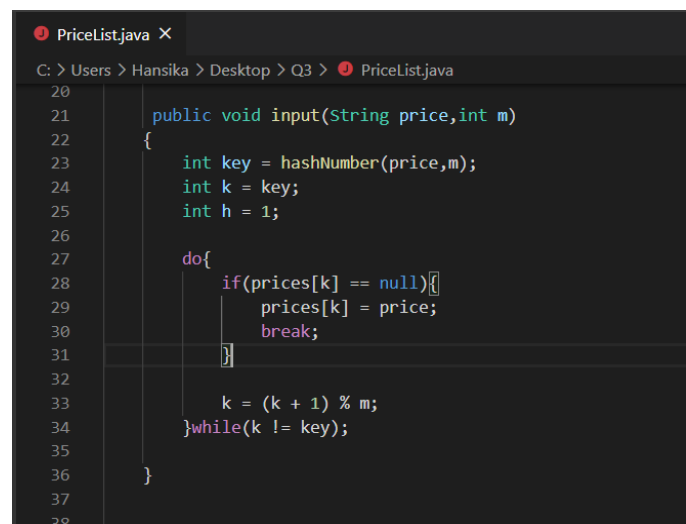
Question 03



```
PriceList.java
C:\Users\Hansika\Desktop> Q3 > PriceList.java
14
15 private int hashNumber(String price,int size)
16 {
17     int key = Integer.parseInt(price);
18     return key % size;
19 }
20
```

The program is built using the java language knowledge. This pricelist program uses a hash table to store outputs as keys and price list as values. Inside the main method it is counted how many prices are there in the PriceList.txt. It executes the overall program. Following code segment shows how Hashing the key process takes place.

Following code segment shows the insertion into the hash table. A separate text file called *PriceList.txt* contains a list of contact details including the pricelist. The program reads it's content line by line using Java Scanner class and splits the line to get price values(String) and output(int)separately.



```
PriceList.java
C:\Users\Hansika\Desktop> Q3 > PriceList.java
20
21 public void input(String price,int m)
22 {
23     int key = hashNumber(price,m);
24     int k = key;
25     int h = 1;
26
27     do{
28         if(prices[k] == null){
29             prices[k] = price;
30             break;
31         }
32
33         k = (k + 1) % m;
34     }while(k != key);
35
36 }
37
38
```

Following code segment shows the comparison of the prices with the offer given.

Sample Test Cases

No.	Test Case	Expected Output	Obtained Output	Pass/Fail
1	331552	1	1	Pass
2	534756	0	1	Fail
3	796873	1	1	Pass
4	248219	1	1	Pass
5	726312	1	1	Pass
6	561237	0	1	Fail

```

59
60
61 public void sum(int price,int m)
62 {
63     int c=0;
64     int n2;
65
66     for (int i = 0; i < m; i++) {
67         n2 = price - Integer.parseInt(prices[i]);
68
69         if(n2 <= 0){
70             continue;
71         }
72         String n2String = Integer.toString(n2);
73
74         int n2Hash = hashNumber(n2String,m);
75
76         int remInt = Integer.parseInt(prices[n2Hash]);
77
78         if(remInt == n2){
79             c = c+1;
80             break;
81         }
82     }
83     if(c==1){
84         System.out.println("1");
85     }else{
86         System.out.println("0");
87     }
88 }
89
90
91
92 }

```

Output:

```

C:\> Command Prompt
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Hansika>cd Desktop/Q3

C:\Users\Hansika\Desktop\Q3>javac PriceList.java

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:331552
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:534756
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:796873
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:248219
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:726312
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:561237
1

C:\Users\Hansika\Desktop\Q3>java PriceList
Input:111122
1

```