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();

```
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

        }
    }
```

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.

```
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.

```java
/*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 PhoneBoo
k.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

```java
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.

```java
    private int asciiVal(String word){//get the totalasciivalues (int values) of  words
        int asciiVal=0;int i;
        for(i=0;i<word.length();i++){
            asciiVal=asciiVal+word.charAt(i);
        }
        return asciiVal;
    }

    private int hashFunction(int asciiVal){ //getting hash codes of words to facilitate hashing in hash tables
        return(0*asciiVal) % SIZE ;
    }

```

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.

```java
    public void insert(String words){//insert list of words to the hash table
        count++;
        int asciiVal = asciiVal(words);
        int hash = hashFunction(asciiVal);
        int i = hash;
        do{
            if(key[i]==0){
                key[i]=asciiVal;value[i]=words;
                count++;
                return;
            }
            i=(i+1)%SIZE;
        }while (i != hash);
    }

    public void printAnagrams(){//print listed words together
        for(int i=0; i<SIZE; i++){
            int temp = 0;int j;
            if (key[i] != 0){
                for(j=i+1; j<SIZE; j++){
                    if(key[i] == key[j]){//check if the values of two words are same
                        char wordArr1[] = value[i].toCharArray();
                        Arrays.sort(wordArr1);
                        char wordArr2[] = value[j].toCharArray();
                        Arrays.sort(wordArr2);//if same put them into these two arrays

                        if(Arrays.equals(wordArr1, wordArr2)){//then if those arrays are equal
                            if(temp == 0){
                                System.out.print(value[i] + " ");
                            }
                            System.out.print(value[j] + " ");//print them in a row
                            key[j] = 0;
                            temp++;
                        }
                    }
                }
            }
            if(temp > 0){
                System.out.println();//put a space after  one anagram set printed
            }
        }
    }
}
```

```java
class CatchAnagrams{
    public static void main(String[] args) throws FileNotFoundException{

        HashTable word = new HashTable();   //call a hashtable   function
        File file = new File("wordlist.txt");
        Scanner input = new Scanner(file);//input the wordlist.txt file

        while(input.hasNext()){
            word.insert(input.next());//insert all the words
        }
        input.close();
        word.printAnagrams();// call print angram function and display output

    }
}
```

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

Inputs and Output:

CatchAnagrams.java | wordlist.txt

wordlist.txt

```
acre
airmen
alarm
algorithm
arc
architecture
asleep
baby
baker
beard
blanket
bored
brake
bread
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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 | Price.java

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

```java

    private int hashNumber(String price,int size)
    {
        int key =  Integer.parseInt(price);
                return key % size;
    }

```

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

```java

    public void input(String price,int m)
    {
        int key = hashNumber(price,m);
        int k = key;
        int h = 1;

        do{
            if(prices[k] == null){
                prices[k] = price;
                break;
            }

            k = (k + 1) % m;
        }while(k != key);

    }

```

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

```java
public void sum(int price,int m)
{
    int c=0;
    int n2;

    for (int i = 0; i < m; i++) {
        n2 = price - Integer.parseInt(prices[i]);

        if(n2 <= 0){

            continue;
        }
        String n2String = Integer.toString(n2);

        int n2Hash = hashNumber(n2String,m);

        int remInt = Integer.parseInt(prices[n2Hash]);


        if(remInt == n2){
            c = c+1;
            break;
        }
    }
    if(c==1){
        System.out.println("1");
    }else{
        System.out.println("0");
    }

}
}
```

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

Output:

```
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
```