```
Script started on 2019-02-26 14:23:08-0800
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# exit
\007pwd\033[Kcat Anagram.java
/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# ls -1
\033[0m\033[01;32mAnagram.java\033[0m
\033[01;32mFindAnagrams.java\033[0m
\033[01;32mHashTable.java\033[0m
\033[01;32mNoteToGrader.txt\033[0m
\033[01;32mREADME.txt\033[0m
\033[01;32mpa3submissionfile.txt\033[0m
\033[01;32mwordList.txt\033[0m
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# cat README.txt
...\CMPS101S18PA3

README.txt- a short file which lists all the files in the directory and describes what they
 are.

NoteToGrader.txt - a short note in which you describe your approach.
                    It also shows a commit log in where I stored
                     my programs in a remote server on gitLab.

Anagram.java - An Angram Type that could represent an anagram of another word.
               Uses Polynomial Accumulation in order to generate hash values.

HashTable.java - Contains a HashTable that stores a dictionary of keys that contain Anagram
s.
                  Uses LinkedList to handle collisions.

FindAnagrams.java - Demonstrates HashTable.java and displays execution time to check O(1) r
unning time.

wordList.txt - The list of words that will be stored in HashTable.java

\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# cat NoteToGrader
.txt
Approach:

My approach to the problem involved using polynomial accumulation in order to generate a
series of unique number and mod it by a prime number times the size of the wordList. The ba
se
for my polynomial accumation is 5 and is raised to the power of the letter ranking in 0-25.
 I found
the following concept in the notes below
http://www.orcca.on.ca/~yxie/courses/cs2210b-2011/htmls/notes/05-hashtable.pdf

I also used LinkedList in order to handle my collisions and stored my set of anagrams in an
 array inside
an array of LinkedList.

gitLab commits:

Author: jwang358 <jwang358@ucsc.edu>
Date:   Tue Feb 26 14:04:04 2019 -0800

    Finished program

Author: jwang358 <jwang358@ucsc.edu>
Date:   Thu Feb 21 23:02:35 2019 -0800

   Started Lab3 (Implemented structure of HashTable and Anagram)
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# cat Anagram.java
```

```
//Programmer: Jeffrey Wang
//Date: 02/06/19
//Class: CMPS101-db

/*sa
 * - a constructor which uses a String, made up of alphabetic characters either
 *   upper or lower case as an input argumen
 * - a constructor which uses a char array, made up of alphabetic characters
 *   either upper or lower case as an input argument.
 * - a method for printing.
 * - a method for comparing two Anagram variables that returns true or false.
 * - a method that returns the word part of an Anagram variable.
 * - do not allow user to get the code part of an Anagram variable.
 */


public class Anagram
{
        private String satellite;               //A word;
        private long hashValue;                 //The hash Value to the word

        /**
          A constructor which a word into an Anagram type and assigns a
          hash value to it.
          @param String value: The word that needs to turn into an anagram..
         */
        public Anagram(String value)
        {
                satellite = value.toLowerCase();
                hashValue =  createHashValue(satellite);
        }

        /**
          A constructor which a word into an Anagram type and assigns a
          hash value to it.
          @param Character[] value: The characters in a word that needs to
                                                                turn into an anagram..
         */
        public Anagram(Character[] value)
        {
                StringBuilder str = new StringBuilder();
                for (int i = 0; i < value.length; i++)
                        str.append(value[i]);
                satellite = str.toString().toLowerCase();
                hashValue = createHashValue(satellite);
        }

        /**The print method for Anagram which prints the word*/
        public void print()
        {
                System.out.println(satellite);
        }

        /**
                This method compares this word and a following word
                to see if they are indeed Anagrams.
                @param Anagram word: The word in which this Anagram
                                                        is being compared to.
                @return boolean: whether or not the words are Anagrams
        */
        public boolean compare(Anagram word)
        {
                // Only continue to compare if the words have the same hasValues(collision)
                if(hashValue == word.getHashValue())
                {
                        //Assign an Integer array that represents a histogram of letters in
 words9
```

```
                        int[] letter1 = new int[26];
                        int[] letter2 = new int[26];

                        for(int i = 0; i < satellite.length(); i++){
                                int alphabet = satellite.charAt(i) - 97;
                                if(alphabet >= 0 && alphabet <=26)
                                        letter1[alphabet]++;
                        }

                        for(int i = 0; i < word.getAnagram().length(); i++){
                                int alphabet = word.getAnagram().charAt(i) - 97;
                                if(alphabet >= 0 && alphabet <=26)
                                        letter2[alphabet]++;
                        }

                        for(int i = 0; i < 26; i++)
                        {
                                if(letter1[i] != letter2[i])
                                        return false;
                        }
                        return true;
                }
                else ;
                        return false;
        }

        /*Returns word of Anagram*/
        public String getAnagram()
        {
                return satellite;
        }

        /*Returns hashValue*/
        public long getHashValue()
        {
                return hashValue;
        }

        /**
         *This function creates a hash value through polynomial
         *accumulation in order to generate a hash value. It uses
         *a base 5 in order to generate a unique value.
     *@param satellite: The string;
         *@return hashValue: A unique hash value
         */
        private long createHashValue(String satellite)
        {
                long hashValue = 0;
                for (int i = 0; i < satellite.length(); i++)
                        hashValue += Math.pow(5,((satellite.charAt(i) - 98)));
                return hashValue;
        }
}

\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# cat HashTable.ja
va
import java.io.*;
import java.util.*;

//Programmer: Jeffrey Wang
//Date: 02/26/19
//COMPS-101-db

/**
 * The following class generates a Hash Table from a list of Anagram words
 * from a textfile. The Table with be sized 10 times the available list and will * use a li
nkedList to store sets of Anagram.
```

```
 */

public class HashTable
{
	private LinkedList[] anagramSets;	//A Hash Table that utilizes a LinkedList t
o
						//Handle collisions.
	private int n;				//Size of list

	/**
	* HashTable Constructor: Intializes a HashTable with a file.txt. It must
	* be initialized with a file.
	* @param File wordList: A file that contains the words of anagrams.
	*
	*/
	public HashTable(File wordList) throws IOException
	{
		//Implement File reader to tranverse through list of words
		FileReader file = new FileReader(wordList);
		Scanner scan = new Scanner(file);
		System.out.println("Reading wordList.txt...");

		//Set the size of the HashTable
		n = 2017 * (Integer.parseInt(scan.nextLine()));
		anagramSets = new LinkedList[n];

		//Counting number of collisions
		int count = 0;

		//Inputs the keys into the hash table
		while(scan.hasNextLine())
		{
			//Generates the key associated with the hash table
			//By taking the mod of the size of array
			Anagram temp = new Anagram(scan.nextLine());
			int key = (int)(temp.getHashValue() % n);

			//Flag for when the Anagram has be inserted into Hash Table
			Boolean flag = true;

			//Labels the first and next LinkedList within a space in anagram
			LinkedList head = anagramSets[key];
			LinkedList next = head;

			//Transerve through loop till Anagram type is inserted
			while(flag)
			{
				if(next != null){
				}
				//If there is no nextLinkedList
				if(next == null)
				{
					//Input a newLinkedList with set of Angrams to arra
y
					if(head == null)
					{
						anagramSets[key] =  new LinkedList(temp);
						flag = false;
					}
					//Sets the new LinkedList to point as the next one
					//if there was a LinkedList before it.
					else
					{
						head.setNext(new LinkedList(temp));
						flag = false;
						count++;
					}
				}
```

```
                                //Checks whether or not the Anagram belongs in a set of
                                //Anagrams
                                else if(temp.compare(next.getHead()))
                                {
                                        next.add(temp);
                                        flag = false;
                                }
                                //Point to the next set of LinkedLists
                                else
                                {
                                        head = next;
                                        next = next.getNext();
                                }
                        }
                }
                System.out.println("Handling " + count + " collisions...");

                //Close filereader
                file.close();
                System.out.println("Closing wordList.txt...");

        }

        /**
        * This method searchs through a Hash Table and returns an
    * array of Strings that are anagrams of a word.
        * @param Anagram focus: The word that we are trying to
        *                        search in the hash Table.
        * @return String[]: Returns an array of string that is an anagram
        *                   of focus.
        */
        public String[] search(Anagram focus)
        {
                LinkedList sameKey = anagramSets[(int)(focus.getHashValue() % n)];
                while(sameKey != null)
                {
                        //Returns LinkedList holding Anagrams
                        if(focus.compare(sameKey.getHead()))
                                return sameKey.getAnagrams();
                        sameKey = sameKey.getNext();
                }
                return new String[] {""};
        }

        /**
         * The following private class will define a set of 'Anagram(object)' that
         * are actual Anagrams.
         */
        private class LinkedList
        {
                private Anagram head = null;            //Points to the first Anagram of th
e set
                private LinkedList next = null;         //Points to the next set of Anagram
s with the same hash value.
                private ArrayList<Anagram> set;         //Creates an ArrayList that stores
values of set.

                /**
                * Required Constructor which takes in the first Anagram word in a set of An
agrams.
                * @param Anagram first: The first word in a set of Anagrams.
                */
                public LinkedList(Anagram first)
                {
                        head = first;
                        set = new ArrayList<>();
                        set.add(first);
                }
```

```
                /**Accessor for Head*/
                public Anagram getHead()
                {
                        return head;
                }

                /**Mutator for next set of Anagrams*/
                public void setNext(LinkedList next)
                {
                        this.next = next;
                }

                /**Accessor for next set of Anagrams*/
                public LinkedList getNext()
                {
                        return next;
                }

                /**adds an Anagram to the set*/
                public void add(Anagram newEle)
                {
                        set.add(newEle);
                }

                /**Returns and array of strings that in the Anagram set*/
                public String[] getAnagrams()
                {
                        String[] a = new String[0];
                        ArrayList<String> wordAnagrams = new ArrayList<>();
                        for(Anagram word : set){
                                wordAnagrams.add(word.getAnagram());
                        }
                        return(wordAnagrams.toArray(a));
                }
        }
}

\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# javac\033[Kc Anag
ram.java HashTabl
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# ls -1
\033[0m\033[01;32mAnagram.class\033[0m
\033[01;32mAnagram.java\033[0m
\033[01;32mFindAnagrams.java\033[0m
\033[01;32m'HashTable$LinkedList.class'\033[0m
\033[01;32mHashTable.class\033[0m
\033[01;32mHashTable.java\033[0m
\033[01;32mNoteToGrader.txt\033[0m
\033[01;32mREADME.txt\033[0m
\033[01;32mpa3submissionfile.txt\033[0m
\033[01;32mwordList.txt\033[0m
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# cat FindAnagrams
.java
import java.util.*;
import java.io.*;

//Programmer: Jeffrey Wang
//Date: 02/26/19
//COMPS-101-db

/**
 *      This test program demonstrates storing a list of words in a
 *      Hash Table with an emphasis on these points:
 *      - compiles without errors and is a serious attempt at a solution
 *  - only reads word file from disk once.
```

```
 *      - for comparisons uses an algorithm faster than brute force,
 *      letterby letter comparison for each word in the word list.T(wordsize) =O(1).
 *      - correctly identifies anagrams for the input string
 *      - does NOT print out the input string as a possible anagram
 *      if ithappens to be a word (the input string does not need to be a word)
 *      - allows for multiple input strings
*/

public class FindAnagrams
{
        //Analyze running time of program.
        private static long startTime;
        private static Long endTime;

        public static void main(String[] args) throws IOException
        {
                //Opens file in order to
                File file = new File("wordList.txt");

                //Analyze running time of creating hashTable
                startTime = System.nanoTime();
                HashTable list = new HashTable(file);
                endTime = System.nanoTime();
                System.out.println("HashTable Generate Time in milliseconds: " + (endTime -
 startTime)/1000000 + "\n");

                Scanner keyboard = new Scanner(System.in);

                StringTokenizer inputs;                 //Tokenizer to take in multiple inp
uts
                String sentinel = " ";                  //Sentinel value in order to exit a
nagram loop

                //Tihs loop interates through multiple inputs from user and displays anagra
m
                while(!sentinel.equals("-1"))
                {
                        System.out.println("Type in a word to find the anagrams or type in
-1 to quit");
                        inputs = new StringTokenizer(keyboard.nextLine());

                        System.out.println("\n");

                        //Iterates through multiple inputs within a single line
                        while(inputs.hasMoreElements())
                        {
                                sentinel = inputs.nextToken();
                                Anagram word = new Anagram(sentinel);
                                System.out.print("Word: ");
                                word.print();

                                System.out.println("Anagrams: ");

                                //Analyze the running time of hash search
                                startTime = System.nanoTime();
                                String[] anagra = list.search(word);
                                endTime = System.nanoTime();

                                //Prints set of anagrams in Linked List except the word its
elf
                                for (int i = 0; i < anagra.length; i++)
                                {
                                        if(!word.getAnagram().equals(anagra[i]))
                                                System.out.println(anagra[i]);
                                }

                                System.out.println("Search Time in milliseconds: " + (endTi
me - startTime)/1000000);
```

```
                                System.out.println("\n\n");
                        }

                }

                        // Testing character constructor of Anagram
                        System.out.println("Testing Character Constructor: ");
                        Character[] characters = new Character[]{'d', 'o', 'g'};
                        Anagram dog = new Anagram(characters);
                        String[] doglist = list.search(dog);

                        System.out.println("Angrams of " + dog.getAnagram() + ":");
                        for (int i = 0; i < doglist.length; i++)
                        {
                                if(!dog.getAnagram().equals(doglist[i]))
                                        System.out.println(doglist[i]);
                        }

        }
}
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# javac FindAnagra
ms.java
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# ls -1
\033[0m\033[01;32mAnagram.class\033[0m
\033[01;32mAnagram.java\033[0m
\033[01;32mFindAnagrams.class\033[0m
\033[01;32mFindAnagrams.java\033[0m
\033[01;32m'HashTable$LinkedList.class'\033[0m
\033[01;32mHashTable.class\033[0m
\033[01;32mHashTable.java\033[0m
\033[01;32mNoteToGrader.txt\033[0m
\033[01;32mREADME.txt\033[0m
\033[01;32mpa3submissionfile.txt\033[0m
\033[01;32mwordList.txt\033[0m
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3#  java Find\007An
agrams\033[K
Reading wordList.txt...
Handling 11894 collisions...
Closing wordList.txt...
HashTable Generate Time in milliseconds: 3298

Type in a word to find the anagrams or type in -1 to quit
items


Word: items
Anagrams:
emits
metis
mites
smite
stime
times
Search Time in milliseconds: 0



Type in a word to find the anagrams or type in -1 to quit
appre


Word: aeprs
Anagrams:
apers
apres
```

```
asper
pares
parse
pears
prase
presa
rapes
reaps
spare
spear
Search Time in milliseconds: 0



Type in a word to find the anagrams or type in -1 to quit
 aelst opst aekst expect


Word: aelst
Anagrams:
least
setal
slate
stale
steal
stela
taels
tales
teals
tesla
Search Time in milliseconds: 0



Word: opst
Anagrams:
opts
post
pots
spot
stop
tops
Search Time in milliseconds: 0



Word: aekst
Anagrams:
skate
stake
steak
takes
teaks
Search Time in milliseconds: 0



Word: expect
Anagrams:
except
Search Time in milliseconds: 0



Type in a word to find the anagrams or type in -1 to quit
```

```
Type in a word to find the anagrams or type in -1 to quit
123123


Word: 123123
Anagrams:

Search Time in milliseconds: 0



Type in a word to find the anagrams or type in -1 to quit
lollipop


Word: lollipop
Anagrams:
Search Time in milliseconds: 0



Type in a word to find the anagrams or type in -1 to quit
-1


Word: -1
Anagrams:

Search Time in milliseconds: 0



Testing Character Constructor:
Angrams of dog:
god
\033]0;root@LAPTOP-52K1L0AJ: /mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3\007roo
t@LAPTOP-52K1L0AJ:/mnt/c/Users/Jeffrey/Desktop/CMPS101S18PA/CMPS101S18PA3# exit
exit

Script done on 2019-02-26 14:27:38-0800
```