Report for Quiz 2 Object Oriented Programming C

Member list:

1. Andi Muhammad Rafli (5025201089)
2. Marsyavero Charisyah Putra (5025201122)
3. Maisan Auliya (5025201137)

1.1 We were told to build a base prototype for predicting text that.
In the first part of the question, we created a class that was named PredictivePrototype and wordToSignature method in the same class. This method will convert the input to the numeric signature and replace any non-alphabetic character with a whitespace. In this method we used StringBuffer class instead of the ordinary String class because String is immutable, if we try to alter their values, another object gets created, whereas StringBuffer is mutable so they can change their values which makes StringBuffer is faster and more efficient than String.

1.2 We created signatureToWords method that will revert back the numeric signature to set of possible words from the given dictionary (words.txt file) Where the returned list must not have duplicates and each word should be lower-case. Here instead of storing the dictionary in our java program, we connect our java program with the words.txt file.

We also created Boolean function called isValidWord that will ignore lines with non-alphabetic characters.

We used Set<String> function because the expected output is in set form

1.3 For the last part of the first question, we created Words2SigProto class with wordToSignature method and Sigs2WordsProto class with signatureToWords method.

The expected output for the first questions will be:

- If we run Word2SigProto by inputting [home,hello, world, my, name, is]

The output should be: 4663 43556 96753 69 6263 47 (as intended in question).

- If we run Sigs2WordsProto by inputting 4663 43556 96753 69 6263 47

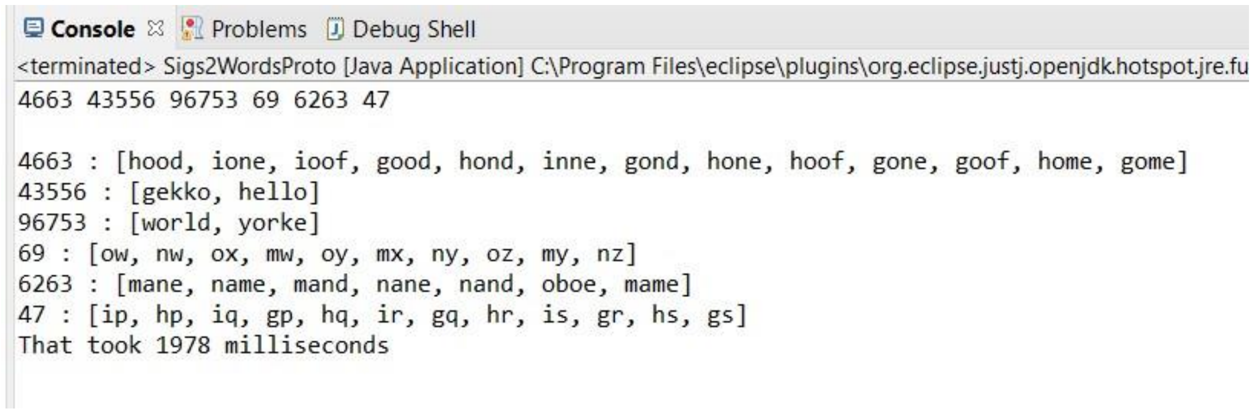
The output should be: 4663 : [hood, ione, ioof, good, hond, inne, gond, hone, hoof, gone, goof, home, gome] 43556 : [gekko, hello] 96753 : [world, yorke] 69 : [ow, nw, ox, mw, oy, mx, ny, oz, my, nz] 6263 : [mane, name, mand, nane, nand, oboe, mame] 47 : [ip, hp, iq, gp, hq, ir, gq, hr, is, gr, hs, gs] (as intended in question).

2. in Storing and searching a dictionary we create a responsive input text where the input text reads the entire dictionary from the disk. Each time a word is being looked up there will be a slight noticeable delay. this program will read and store the dictionary in memory as a list of a pairs We first defined the WordSig object to store the pairs. It implements Comparable. It contains Strings: word and signature. To make the object simpler to sort and comparable, we overrode the compareTo method to only compare the signatures of two WordSig objects.
   In DictionaryListImpl, we define the arraylist List to store the pairs. In the constructor we store linearly the words and its signature using wordToSignature, then sort it after with Collections.sort. In signatureToWords, we implement our own Linear Search algorithm, then after getting the index, we move the index to the first instance of matching word, then adds each next word until it no longer matches.

   To record the execution time, we used System.currentTimeMillis to get the elapsed time in millisecond scale.

   The comparison between the first program prototype (question no.1) and the improved program (question no.2)

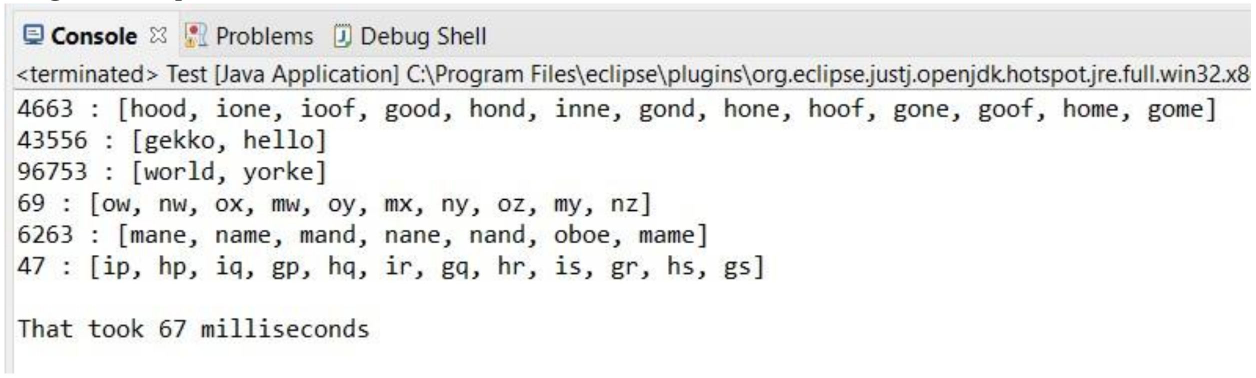
   Program for question number one's recorded time:



```
<terminated> Sigs2WordsProto [Java Application] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
4663 43556 96753 69 6263 47

4663 : [hood, ione, ioof, good, hond, inne, gond, hone, hoof, gone, goof, home, gome]
43556 : [gekko, hello]
96753 : [world, yorke]
69 : [ow, nw, ox, mw, oy, mx, ny, oz, my, nz]
6263 : [mane, name, mand, nane, nand, oboe, mame]
47 : [ip, hp, iq, gp, hq, ir, gq, hr, is, gr, hs, gs]
That took 1978 milliseconds
```

Program for question number two’s recorded time:



Console ☒ Problems Debug Shell

```
<terminated> Test [Java Application] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
4663 : [hood, ione, ioof, good, hond, inne, gond, hone, hoof, gone, goof, home, gome]
43556 : [gekko, hello]
96753 : [world, yorke]
69 : [ow, nw, ox, mw, oy, mx, ny, oz, my, nz]
6263 : [mane, name, mand, nane, nand, oboe, mame]
47 : [ip, hp, iq, gp, hq, ir, gq, hr, is, gr, hs, gs]

That took 67 milliseconds
```

It was proven that the second program is more time efficient.

For the 3rd and 4th questions, our team member couldn’t find the solusions for the problems

Distribution of work :
For every problem, we decide to finish our work together with conclusion of our own opinion

Andi Muhammad Rafli 33,3%
Maisan Auliya 33,3%
Marsyavero Charisya Putra 33,3%