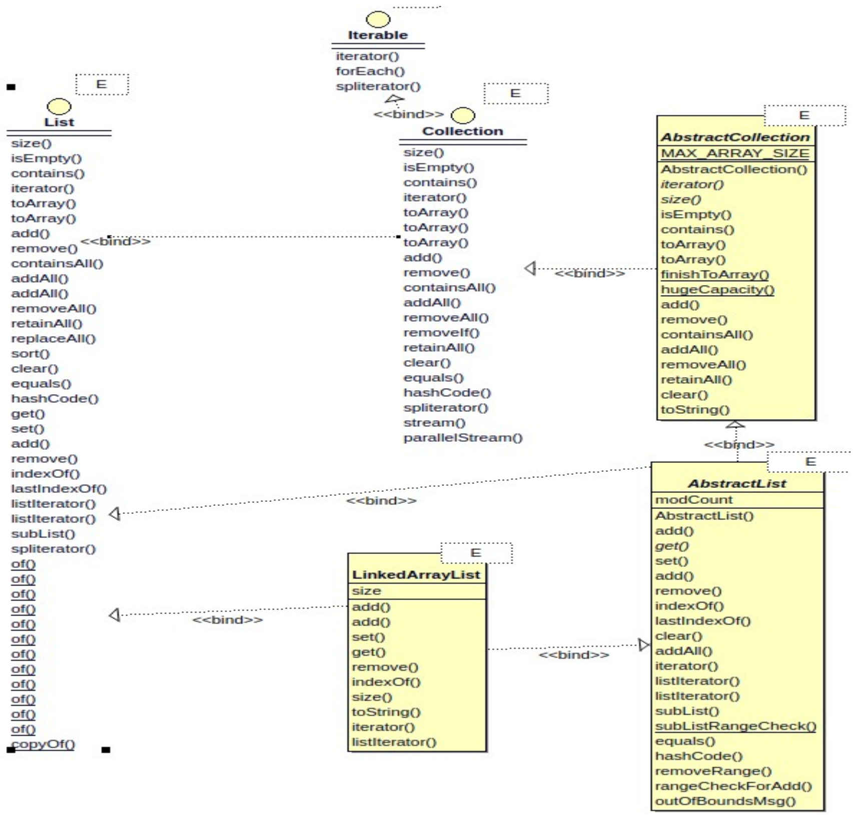
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 3 Report

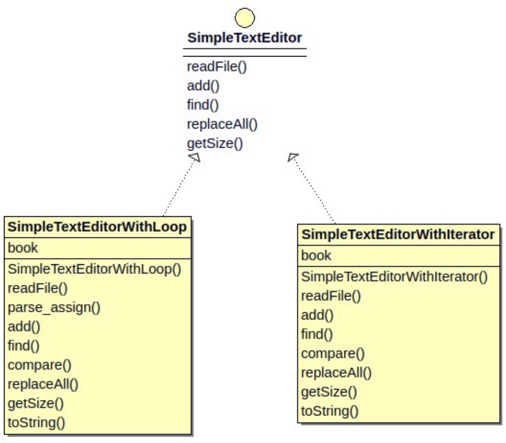
Muharrem Ozan Yeşiller
171044033

1. CLASS DIAGRAMS

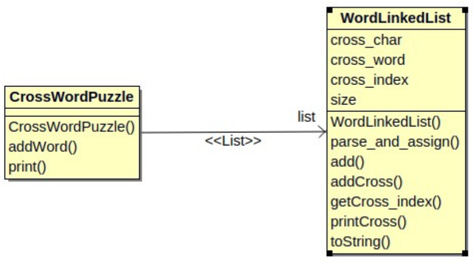
1.1(Question1)



1.2(Question 2)



1.3(Question 3)



2. PROBLEM SOLUTION APPROACH

2.1(Question1)

In this part, i implemented a linked array list structure. When doing these implementations, adding data to the structure requested from us was to extract data from the structure. There is a node structure that connects our data within classes. We have real data in this node structure. We kept this data in a fixed capacity array. When the user is adding data, the structure connects a new node, that is, a new array, to the end of the structure if the array is full. There is a problem in this section, what will we do if the user adds an index? In fact, this is the main problem, I have followed a way to solve this problem. If the object to be inserted by the user is between a full array, last element of array shifts from the right of the index it wants to adding element. If we have an object at the last hand (ie if the arrays are full), we connect the last element we want to shift to the structure with a new node. In removing, user can delete these elements by giving an index. We reach the index that the user wants to delete and scroll to the left from that index. The object can be sent to the method for removing. In this case, the indexed removing method is used in the index where the object is located.
So how does the program reach these indices? This program, which adds and removing with the help of Iterator, reaches these indices again using iterator. But this iterator is a slightly more intelligent iterator. This iterator doesn't just progress between nodes like normal iterators. It looks at the array elements first, while looking at the array elements, if there are no more elements to look at in that arrays, it moves to the next node. While doing this operation, the iterator created is kept in the class in which node and which index is looking. and this creates a iterator by correctly answering the question of which index of the array of which node, according to the index sent by the user. Now, when this iterator occurs, the program knows which object to perform the operations and performs these operations according to the method the user calls.

2.2(Question2)

In this part, I have implemented a text editor class. In this section, our class is created as a list object. The constructor takes two parameters, one is the file it will read, and one is the list object. This is because it can be used in many list structures. While creating the Editor object, each character received from the file is added to the end of the list structure. And then a character or character string can be added to this structure via this object, the searched word can be found in the text and a character in the text can be replaced by another character. While implementing these methods, I used two different structures. One with the help of iterator, and one with the loop. In general, only the loop has changed in the iterator used part. Because the algorithm is the same in both.

The algorithm of the read method works as follows:
Adds the character read from the file to the end of the list.

The algorithm of the add method works as follows:
The given string is parsed and inserted from the given index.

The algorithm of the find method works as follows:
When navigating the structure one by one, if the first letter of the incoming string matches, it compares that character string with the incoming character string, if it finds that character string, it returns the starting index. If not, the file returns -1 to the end.

The algorithm of the Replace method works as follows:
If it finds the character that the user chooses while traveling on the building one by one, he assigns the character he wants the user to replace.

2.3(Question3)

In this section, evaluate the words that the user enters into the puzzle system, and if there is a word that can match the letter, add these letters to the cross node of the word it coincides with. I have implemented the method of adding the letters that the words overlap if they are working so that they cross each other. In this part, it is important whether the word has a cross in that letter before. The solution is discussed in this section.

3. TEST CASES

3.1(Question1)

First, several elements are added to the structure using add (E e). Then, data is added to the structure using the index, that is, using add (int index, E e). Then, data is deleted from the structure using remove (int index) and remove (Object o). Then mixed and added is done. Perform this test piece was made with the end points, how to give reactions when added to the end or middle "Running command and results" are described in part.

3.2(Question2)

First, I handled situations that would show how methods work in a low-character file. Why low character file? Because it's easy to see how the methods work.

The tests of these methods are as follows:
1) The object is created. If the object is an exception throw, the reading process is successful.

2) A few words are added to the object. Random index, end, beginning, specific index. If the object is an exception throw, the reading process is successful.

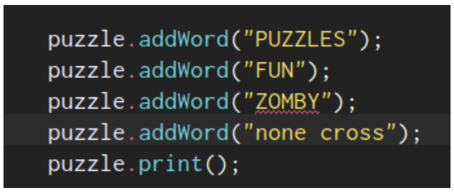
3) Several characters / words are searched in the text of the object. If the object is an exception throw, the reading process is successful.

4) Instead of some letters, other letters are used in the object. If the object is an exception throw, the reading process is successful.

Then the time complexity is calculated by creating objects with works iterator and objects with works loop, with linkedlist and arraylist. And Theese are compared experimentally and theoretically. These accounts are handled in larger files.

3.3(Question3)

It is a program with a user interface. But I made a test by adding a few words and print them on the screen. Here are the words I added:



```
puzzle.addWord("PUZZLES");
puzzle.addWord("FUN");
puzzle.addWord("ZOMBY");
puzzle.addWord("none cross");
puzzle.print();
```

Their comparisons are theoretically as follows:

List is an ArrayList and iterator is used:

READ: O(sizeof_characters x O(1))
O(sizeof_characters x O(1)) → adding(used index) up to character lentgh
(When the capacity is full, it processes O(1 x sizeof_characters) + O(n))

ADD: Assume that n = size – index, O(n)
O(n) → (If the index is not at the end, it is shifted from the added element.)
(When the adding last element, it processess O(1))

FIND: O(n) + (O(1) x O(n))
O(n) → create iterator, O(1) → advances between nodes, O(n) → compare

REPLACE: O(n) + O(1)
O(n) → creates iterator, O(1) → advances between nodes and set

List is an ArrayList and iterator is not used:

READ: O(sizeof_characters x O(1))
O(n) → creates iterator, O(sizeof_characters x O(1)) → adding(used index) up to character lentgh.

ADD: Assume that n = size – index, O(n)
O(n) → (If the index is not at the end, it is shifted from the added element.)

FIND: O(n) x O(string_size) = O(n x string_size)
O(n) → advance till arraylist size, O(string_size) → compare character or characters

REPLACE: O(n) x O(1)
O(n) → advance till arraylist size, O(1) → set data

List is a LinkedList and iterator is used:

READ: O(n) + O(sizeof_characters x 1)
O(n) → create iterator, O(sizeof_characters x O(1)) → adding(used next()) up to character lentgh.

ADD: O(n) + O(string_length x 1)
O(n) → create indexth iterator, O(string_length x 1) → adding up to string length

FIND: O(n) x O(string_size) = O(n x string_size)
O(n) → create iterator, O(n x string_size) → compare n times string

REPLACE: O(n) + O(1)
O(n) → create iterator, O(1) → advances between nodes and set

List is a LinkedList and iterator is not used:

READ: O(sizeof_characters x O(n) = O(n x sizeof_characters)
O(sizeof_characters) → till characters size of file, O(n) → adding last list

ADD: O(string_length x O(n))
O(string_length) → till string length, O(n) → Advances and adds up to index(n)

FIND: O(n) x O(string_size) = O(n x string_size)
O(n) → advances "n" element of list, O(string_size) → compare if there is a possible character

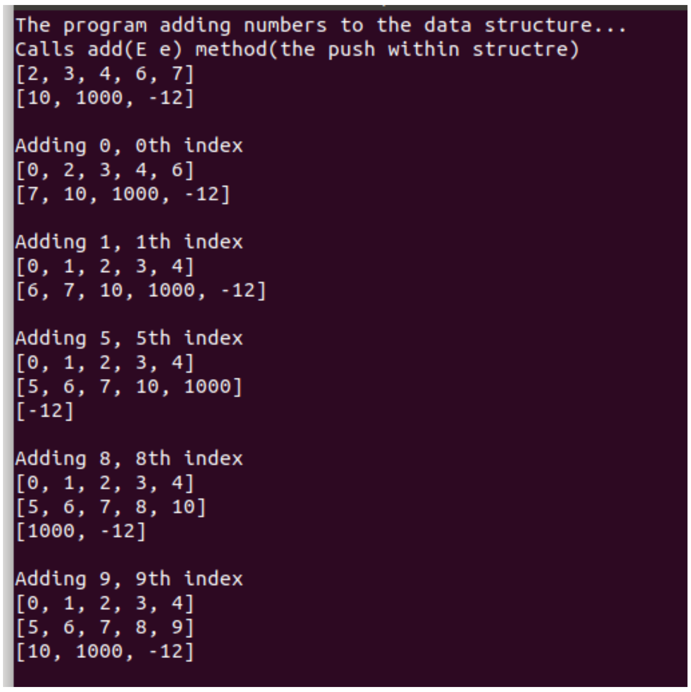
REPLACE: O(n) x O(index) = O(n x index)
O(n) → size of list, O(index) → indexth data set

Their comparisons are experimental performance as follows:

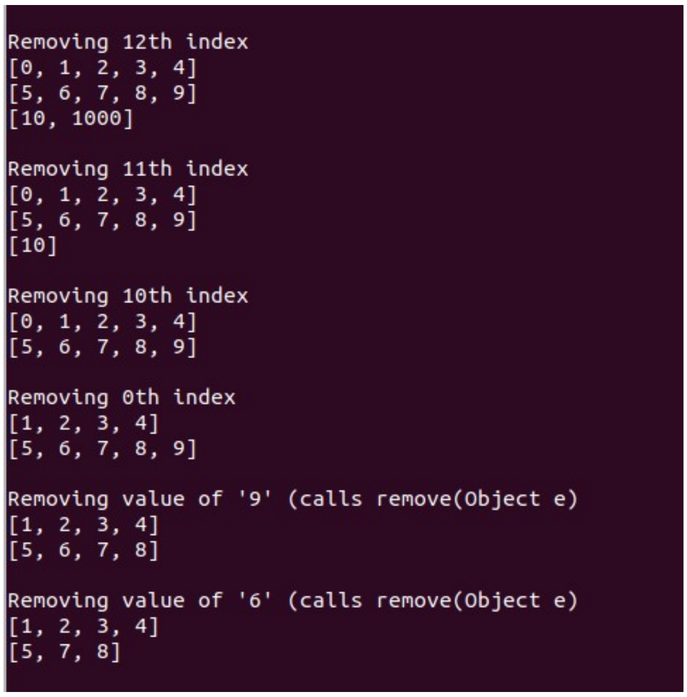
```
---------------------------------------------------------------------
TIMES FOR two.txt. two.txt has 1412 characters.
---------------------------------------------------------------------
The Linked list with loop:
------------------
READ: 1714890 ns
ADD: 60224 ns
FIND: 4054607 ns
REPLACE: 4406206 ns
------------------
The Linked list with iterator:
------------------
READ: 651641 ns
ADD: 16103 ns
FIND: 464112 ns
REPLACE: 570491 ns
------------------
The Array list with loop:
------------------
READ: 560716 ns
ADD: 14275 ns
FIND: 1092683 ns
REPLACE: 521800 ns
------------------
The Array list with iterator:
------------------
READ: 954771 ns
ADD: 21163 ns
FIND: 836776 ns
REPLACE: 514534 ns
------------------
---------------------------------------------------------------------
TIMES FOR three.txt. three.txt has 8267 characters.
---------------------------------------------------------------------
The Linked list with loop:
------------------
READ: 1369626 ns
ADD: 61684 ns
FIND: 56011400 ns
REPLACE: 127485783 ns
------------------
The Linked list with iterator:
------------------
READ: 899339 ns
ADD: 17312 ns
FIND: 2392068 ns
REPLACE: 2328750 ns
------------------
The Array list with loop:
------------------
READ: 1154657 ns
ADD: 8347 ns
FIND: 956600 ns
REPLACE: 1191900 ns
------------------
The Array list with iterator:
------------------
READ: 879822 ns
ADD: 10567 ns
FIND: 966297 ns
REPLACE: 1781170 ns
```

4. RUNNING AND RESULTS

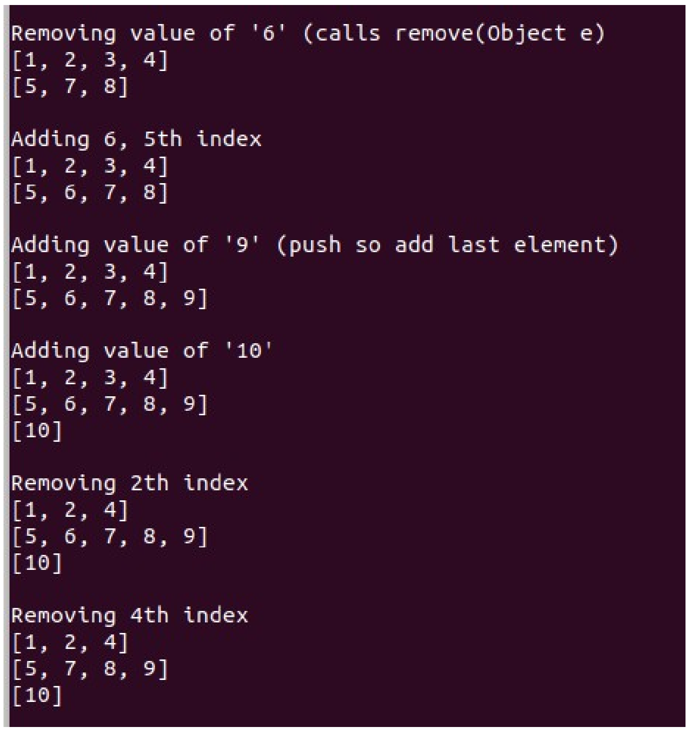
4.1(Question1)



```
The program adding numbers to the data structure...
Calls add(E e) method(the push within structre)
[2, 3, 4, 6, 7]
[10, 1000, -12]

Adding 0, 0th index
[0, 2, 3, 4, 6]
[7, 10, 1000, -12]

Adding 1, 1th index
[0, 1, 2, 3, 4]
[6, 7, 10, 1000, -12]

Adding 5, 5th index
[0, 1, 2, 3, 4]
[5, 6, 7, 10, 1000]
[-12]

Adding 8, 8th index
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 10]
[1000, -12]

Adding 9, 9th index
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[10, 1000, -12]
```



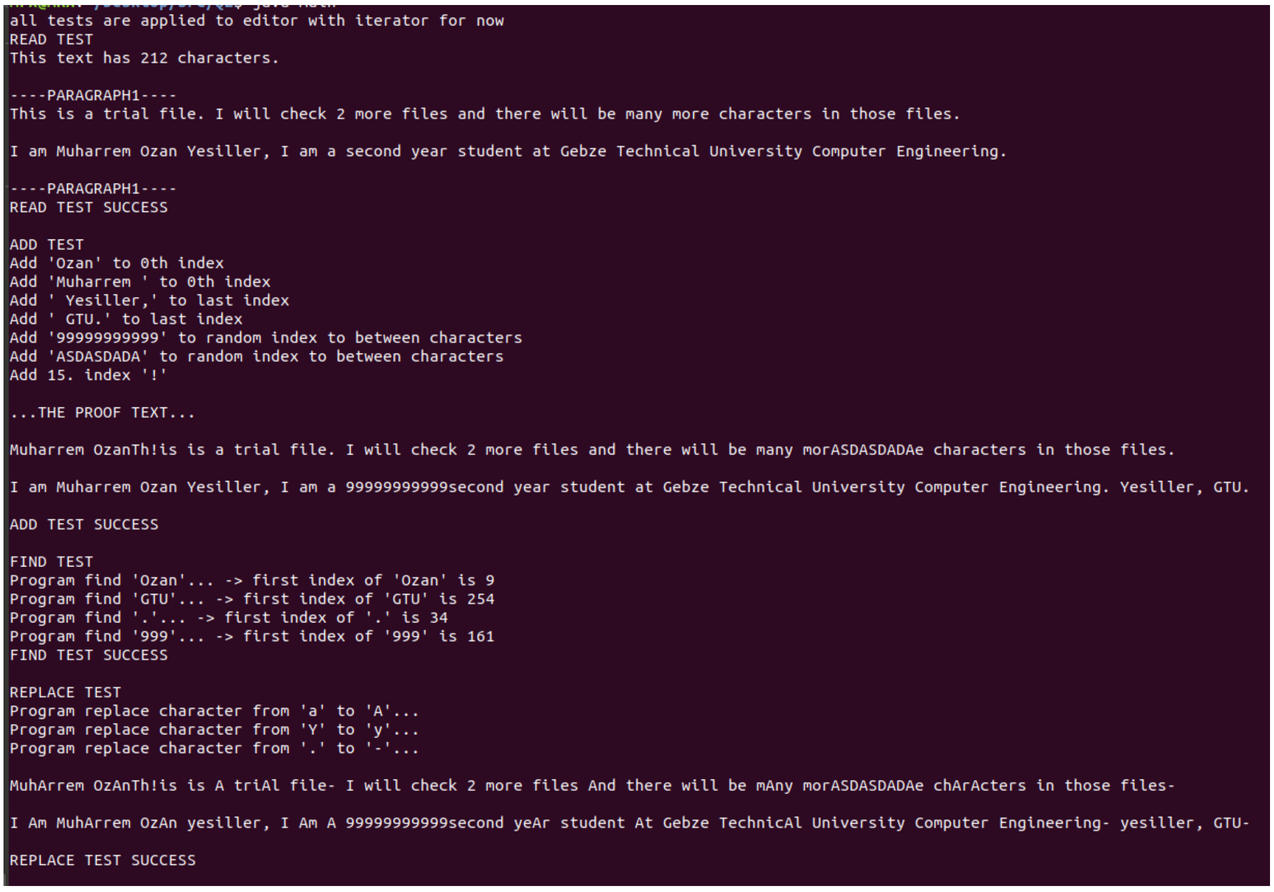
```
Removing 12th index
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[10, 1000]

Removing 11th index
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
[10]

Removing 10th index
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]

Removing 0th index
[1, 2, 3, 4]
[5, 6, 7, 8, 9]

Removing value of '9' (calls remove(Object e)
[1, 2, 3, 4]
[5, 6, 7, 8]

Removing value of '6' (calls remove(Object e)
[1, 2, 3, 4]
[5, 7, 8]
```



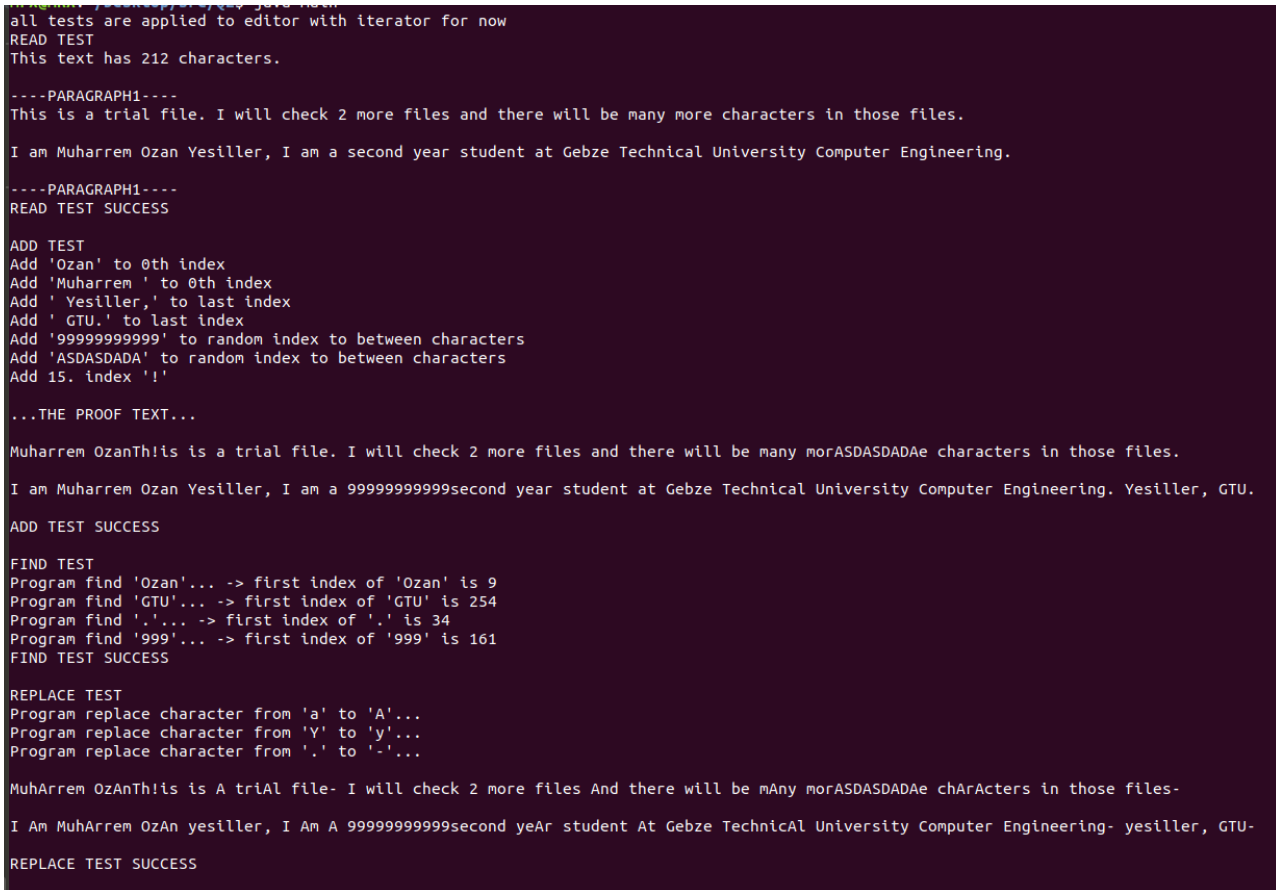
```
Removing value of '6' (calls remove(Object e)
[1, 2, 3, 4]
[5, 7, 8]

Adding 6, 5th index
[1, 2, 3, 4]
[5, 6, 7, 8]

Adding value of '9' (push so add last element)
[1, 2, 3, 4]
[5, 6, 7, 8, 9]

Adding value of '10'
[1, 2, 3, 4]
[5, 6, 7, 8, 9]
[10]

Removing 2th index
[1, 2, 4]
[5, 6, 7, 8, 9]
[10]

Removing 4th index
[1, 2, 4]
[5, 7, 8, 9]
[10]
```

4.2(Question2)

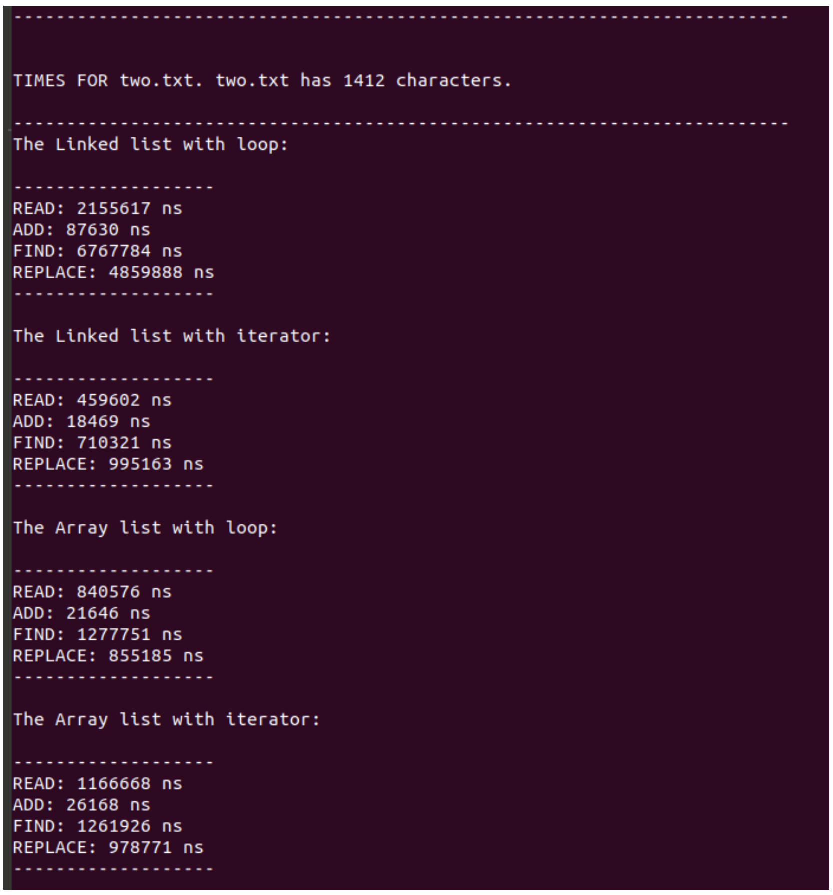
(THE PROGRAM PRINTS THESE RESULTS INTO A LOGFILE.)



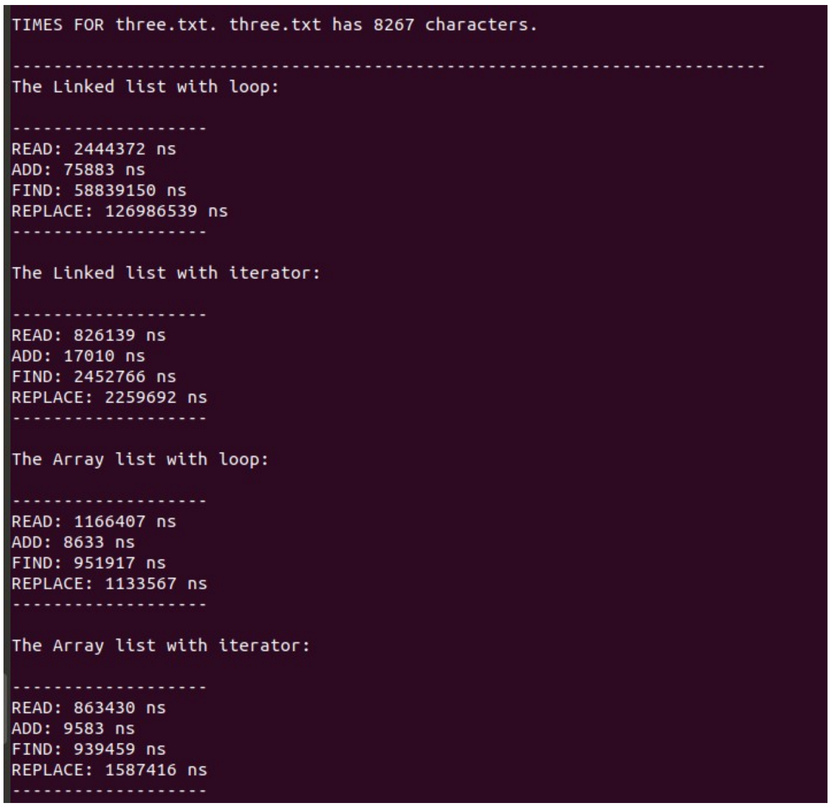
```
all tests are applied to editor with iterator for now
READ TEST
This text has 212 characters.

----PARAGRAPH1----
This is a trial file. I will check 2 more files and there will be many more characters in those files.

I am Muharrem Ozan Yesiller, I am a second year student at Gebze Technical University Computer Engineering.

----PARAGRAPH1----
READ TEST SUCCESS

ADD TEST
Add 'Ozan' to 0th index
Add 'Muharrem ' to 0th index
Add ' Yesiller,' to last index
Add ' GTU.' to last index
Add '99999999999' to random index to between characters
Add 'ASDASDADA' to random index to between characters
Add 15. index '!'

...THE PROOF TEXT...

Muharrem OzanTh!is is a trial file. I will check 2 more files and there will be many morASDASDADAe characters in those files.

I am Muharrem Ozan Yesiller, I am a 99999999999second year student at Gebze Technical University Computer Engineering. Yesiller, GTU.

ADD TEST SUCCESS

FIND TEST
Program find 'Ozan'... -> first index of 'Ozan' is 9
Program find 'GTU'... -> first index of 'GTU' is 254
Program find '.'... -> first index of '.' is 34
Program find '999'... -> first index of '999' is 161
FIND TEST SUCCESS

REPLACE TEST
Program replace character from 'a' to 'A'...
Program replace character from 'Y' to 'y'...
Program replace character from '.' to '-'...

MuhArrem OzAnTh!is is A triAl file- I will check 2 more files And there will be mAny morASDASDADAe chArActers in those files-

I Am MuhArrem OzAn yesiller, I Am A 99999999999second yeAr student At Gebze TechnicAl University Computer Engineering- yesiller, GTU-

REPLACE TEST SUCCESS
```



```
all tests are applied to editor with iterator for now
READ TEST
This text has 212 characters.

----PARAGRAPH1----
This is a trial file. I will check 2 more files and there will be many more characters in those files.

I am Muharrem Ozan Yesiller, I am a second year student at Gebze Technical University Computer Engineering.

----PARAGRAPH1----
READ TEST SUCCESS

ADD TEST
Add 'Ozan' to 0th index
Add 'Muharrem ' to 0th index
Add ' Yesiller,' to last index
Add ' GTU.' to last index
Add '99999999999' to random index to between characters
Add 'ASDASDADA' to random index to between characters
Add 15. index '!'

...THE PROOF TEXT...

Muharrem OzanTh!is is a trial file. I will check 2 more files and there will be many morASDASDADAe characters in those files.

I am Muharrem Ozan Yesiller, I am a 99999999999second year student at Gebze Technical University Computer Engineering. Yesiller, GTU.

ADD TEST SUCCESS

FIND TEST
Program find 'Ozan'... -> first index of 'Ozan' is 9
Program find 'GTU'... -> first index of 'GTU' is 254
Program find '.'... -> first index of '.' is 34
Program find '999'... -> first index of '999' is 161
FIND TEST SUCCESS

REPLACE TEST
Program replace character from 'a' to 'A'...
Program replace character from 'Y' to 'y'...
Program replace character from '.' to '-'...

MuhArrem OzAnTh!is is A triAl file- I will check 2 more files And there will be mAny morASDASDADAe chArActers in those files-

I Am MuhArrem OzAn yesiller, I Am A 99999999999second yeAr student At Gebze TechnicAl University Computer Engineering- yesiller, GTU-

REPLACE TEST SUCCESS
```

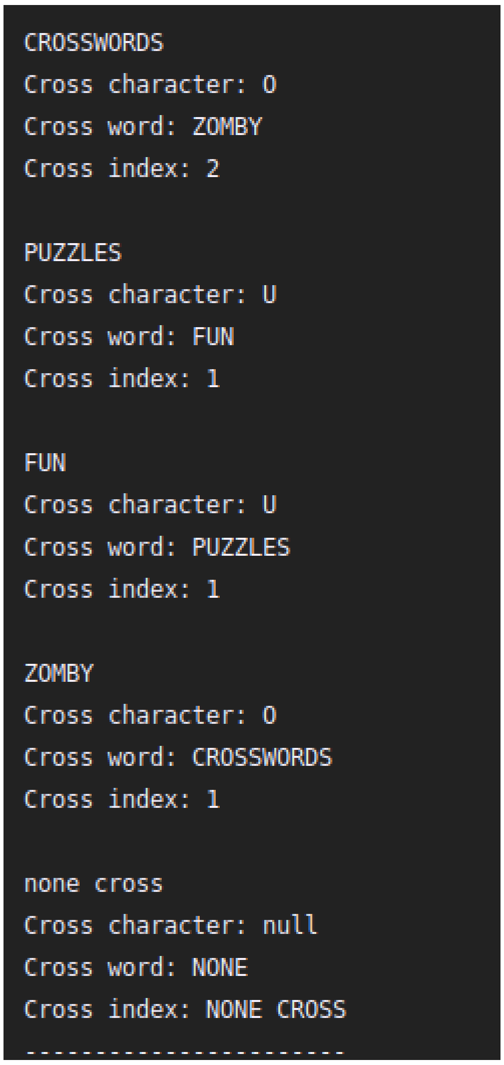


```
------------------------------------------------------------------------


TIMES FOR two.txt. two.txt has 1412 characters.

------------------------------------------------------------------------
The Linked list with loop:

-------------------
READ: 2155617 ns
ADD: 87630 ns
FIND: 6767784 ns
REPLACE: 4859888 ns
-------------------

The Linked list with iterator:

-------------------
READ: 459602 ns
ADD: 18469 ns
FIND: 710321 ns
REPLACE: 995163 ns
-------------------

The Array list with loop:

-------------------
READ: 840576 ns
ADD: 21646 ns
FIND: 1277751 ns
REPLACE: 855185 ns
-------------------

The Array list with iterator:

-------------------
READ: 1166668 ns
ADD: 26168 ns
FIND: 1261926 ns
REPLACE: 978771 ns
-------------------
```



```
TIMES FOR three.txt. three.txt has 8267 characters.

------------------------------------------------------------------------
The Linked list with loop:

-------------------
READ: 2444372 ns
ADD: 75883 ns
FIND: 58839150 ns
REPLACE: 126986539 ns
-------------------

The Linked list with iterator:

-------------------
READ: 826139 ns
ADD: 17010 ns
FIND: 2452766 ns
REPLACE: 2259692 ns
-------------------

The Array list with loop:

-------------------
READ: 1166407 ns
ADD: 8633 ns
FIND: 951917 ns
REPLACE: 1133567 ns
-------------------

The Array list with iterator:

-------------------
READ: 863430 ns
ADD: 9583 ns
FIND: 939459 ns
REPLACE: 1587416 ns
-------------------
```

4.3(Question3)



```
CROSSWORDS
Cross character: O
Cross word: ZOMBY
Cross index: 2

PUZZLES
Cross character: U
Cross word: FUN
Cross index: 1

FUN
Cross character: U
Cross word: PUZZLES
Cross index: 1

ZOMBY
Cross character: O
Cross word: CROSSWORDS
Cross index: 1

none cross
Cross character: null
Cross word: NONE
Cross index: NONE CROSS
-----------------------
```