

Algorithm and Data Structures

Assignment 2: Report

Name: Maeda Taishin

Student ID: 1W21CF15

Date: 25/04/2023

Mob Programming with Simon Choi 1W21CF07

Exercise 1-1: Linear Search using while-Loop

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.List;
6  import java.util.Scanner;
7  public class LinearS2 {
8      public static int seqSearch(int[] array, int n, int x){
9          int i = 0;
10         while(i<n){
11             if(array[i] == x){
12                 return i;
13             }
14             i++;
15         }
16         return -1;
17     }
18     public static void main(String[] args){
19         Scanner sc = new Scanner(System.in);
20
21         try {
22             Path file = Paths.get (first: "test-data-10.txt");
23             List<String> stringData = Files.readAllLines(file);
24
25             int[] array = new int[stringData.size()];
26             for (int i = 0; i<array.length; i++){
27                 array[i] = Integer.parseInt(stringData.get(i));
28             }
29             for (int i = 0; i<array.length; i++){
30                 System.out.print(array[i]+ " ");
31             }
32
33             System.out.println();
34             System.out.println(x: "Enter an element: ");
35             int x = sc.nextInt();
36             int n = array.length;
37             int index = seqSearch(array, n, x);
38             if (index == -1){
39                 System.out.println(x: "The element not found");
40             }else{
41                 System.out.println("The element found in array "+ index);
42             }
43         } catch (IOException e){
44             e.printStackTrace();
45         }
46     }
47 }
48
```

Code Explanation:


Initially, I would like to explain the code in the main function. As seen in the picture (line: 19), I start by setting up a scanner object called “sc” to get user input by importing the Scanner library. The explanation of reading a file from a text file will be discussed later. To help users understand what's happening in this program, I first print the elements inside the array from the text file using a for-loop to repeatedly print out elements based on the array size (Line: 29-31). I use “System.out.print” and add a space after printing each element, instead of using “System.out.println”, to keep the program output in one line. Next, I prompt users to input the element they want to find. I create a variable called “x” to store the integer input from the users, which will be used later when calling a searching function. I also create a variable called “n” to represent the size of the array, as the program starts counting the first element as the zeroth position. In line 37 of this program, I create a variable called “index” to call the searching function named “seqSearch”, passing the array “array”, the array's size “n”, and the input “x” (The function will be discussed later). Finally, I use an if-else statement (Line: 38-42) to check whether the condition for finding the element holds, based on the value of “index” returned from the

searching function. For instance, if the value of “index” is -1, indicating that the element is not found in the array, the program will execute “The element not found”. On the other hand, if the value of “index” is not -1, then the element is found in the array and will be printed out as “The element found in the array” followed by the position of the element located inside the array.

For an explanation about reading a text file, the idea is that we first assumed the text file, test-data-10.txt or test-data-100.txt in this case, are the same directory (Line: 22). Next, we needed to read all lines in the files provided and store data in a list if string(Line: 23). Then, we needed to convert the list that is recognized as a string in this case to an integer array (Lines: 25-28). In order to do that, several Java libraries are needed to be stored at the beginning of the code (Lines: 1-5). Lastly, the code inside the main function will be placed inside the try-catch exception which allows us to define blocks of codes to be tested or executed if an error occurs (Lines: 21 and 43-44).

To elaborate on the searching function “seqSearch” which is done by using a while-loop, I first initialized a variable “i” to 0 (Line: 9). Then, I created a while-loop with a condition that repeats as long as the value of “i” is less than or equal to the size of the array (line: 10). Inside the loop, I used an if-else statement to check if the element at position “i” in the array is equal to the input “x” that the user prompted (line: 11). If this condition is satisfied, then the value of “i” will be returned to the main function indicating the position of the element “x” in the array. However, if the condition is not satisfied, the value of “i” will be incremented until it becomes greater than “n” where “n” represents the size of the array indicating that the element “x” is not found in the array. From that being said, the function will return -1 to the main function meaning that the element “x” is not included in the array.

test-data-10.txt



```
9 0 1 13 8 2 8 1 8 3
Enter an element:
8
The element found in array 4
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The screenshot displays the results of searching for an element, “8” in this case, within an array that contains 10 elements. The result shows that the element is found at the 4th position in the array.

test-data-100.txt

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code % cd "/Users/chawinwaimaleongora-ek/Downloads/2nd year/Spring Semester/Algorithms and data structures/Code/" && javac LinearS2.java && java LinearS2
70 78 19 68 17 12 3 14 74 68 22 59 49 9 18 21 43 26 65 60 6 8 76 15 58 22 2 45 45 25 5 69 19 15 33 73 65 27 57 17 60 46 49 76 35 19 58 65 70 1 45 20 1 3 47 5 41 51 45 34 49 66 61 60 41 69 64 55 68 58 64 26 69 72 33 7 54 4 71 37
52 32 21 40 42 46 77 66 56 2 57 19 79 69 4 60 28 22 5 66
Enter an element:
50
The element found in array 46
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code % cd "
70 78 19 68 17 12 3 14 74 68 22 59 49 9 18 21 43 26 65
52 32 21 40 42 46 77 66 56 2 57 19 79 69 4 60 28 22 5
Enter an element:
50
The element found in array 46
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The screenshot displays the results of searching for an element, “50” in this case, within an array that contains 100 elements. The result shows that the element is found at the 46th position in the array.

Exercise 1-2: Linear search using for-Loop

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.List;
6  import java.util.Scanner;
7  public class LinearS1 {
8      public static int seqSearch(int[] array, int n, int x){
9          for(int i=0; i<n; i++){
10             if(array[i] == x){
11                 return i;
12             }
13         }
14         return -1;
15     }
16     public static void main(String[] args){
17         Scanner sc = new Scanner(System.in);
18
19         try {
20             Path file = Paths.get ("first: test-data-10.txt");
21             List<String> stringData = Files.readAllLines(file);
22
23             int[] array = new int[stringData.size()];
24             for (int i = 0; i<array.length; i++){
25                 array[i] = Integer.parseInt(stringData.get(i));
26             }
27             for (int i = 0; i<array.length; i++){
28                 System.out.print(array[i]+ " ");
29             }
30
31             System.out.println();
32             System.out.println(x: "Enter an element: ");
33             int x = sc.nextInt();
34             int n = array.length;
35             int index = seqSearch(array, n, x);
36             if (index == -1){
37                 System.out.println(x: "The element not found");
38             }else{
39                 System.out.println("The element found in array "+ index);
40             }
41         } catch (IOException e){
42             e.printStackTrace();
43         }
44     }
45 }
46
```

Code Explanation:

The main function of this program is the same as the previous program. However, I used a for-loop as the search function. The concept inside the for-loop inside the searching function is the algorithm runs until the value of an element in “i”th position inside the array is equal to the value of “x”. Firstly (Line: 9), “i” is initialized as zero, and it will increment itself until it reaches its limit which is less than the array’s size (“n”). Inside the for-loop (Line: 10-11), an if-else statement is created in order to check the condition of whether the array[i] is equal to “x”. If so, the program will return the value “i” which will result in “The element found in the array (index)”. If not, then the program will repeat itself until “i” is not less than “n”. Consequently, if array[i] is still not equal to “x”, then the program will return -1 which will result in “The element not found”.

test-data-10.txt

```
9 0 1 13 8 2 8 1 8 3
```

```
Enter an element:
```

```
8
```

```
The element found in array 4
```

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The output of this program is the same as the one with a while-loop.

test-data-100.txt

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code % cd "/Users/chawinwaimaleongora-ek/Downloads/2nd year/Spring Semester/Algorithms and data structures/Code/" && javac LinearSI.java && java LinearSI
70 78 19 68 17 12 3 14 74 68 22 59 49 9 18 21 43 26 65
52 32 21 40 42 46 77 66 56 2 57 19 79 69 4 60 28 22 5
Enter an element:
50
The element found in array 46
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code % cd "
70 78 19 68 17 12 3 14 74 68 22 59 49 9 18 21 43 26 65
52 32 21 40 42 46 77 66 56 2 57 19 79 69 4 60 28 22 5
Enter an element:
50
The element found in array 46
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The output of this program is the same as the one with a while-loop.

Exercise 2: Sentinel Search by using while-loop

```
1  import java.io.IOException;
2  import java.nio.file.Files;
3  import java.nio.file.Path;
4  import java.nio.file.Paths;
5  import java.util.List;
6  import java.util.Scanner;
7  public class SentinelS {
8      public static int senSeqSearch(int[] a, int n, int x) {
9          int[] arrayS = new int[n+1];
10         System.arraycopy(a, srcPos: 0, arrayS, destPos: 0, n);
11         arrayS[n] = x;
12
13         int i = 0;
14         while (arrayS[i] != x) {
15             i++;
16         }
17         if (i < n && arrayS[i] == x) {
18             return i;
19         } else {
20             return -1;
21         }
22     }
23     public static void main(String[] args){
24         Scanner sc = new Scanner(System.in);
25
26         try {
27             Path file = Paths.get (first: "test-data-10.txt");
28             List<String> stringData = Files.readAllLines(file);
29
30             int[] array = new int[stringData.size()];
31             for (int i = 0; i < array.length; i++){
32                 array[i] = Integer.parseInt(stringData.get(i));
33             }
34             for (int i = 0; i < array.length; i++){
35                 System.out.print(array[i]+ " ");
36             }
37
38             System.out.println();
39             System.out.println(x: "Enter an element: ");
40             int x = sc.nextInt();
41             int n = array.length;
42             int index = senSeqSearch(array, n, x);
43             if (index == -1){
44                 System.out.println(x: "The element not found");
45             }else{
46                 System.out.println("The element found in array "+ index);
47             }
48         } catch (IOException e){
49             e.printStackTrace();
50         }
51     }
52 }
53
```

Code Explanation:

The main function of this program is similar to the two previous programs. However, the algorithm in this search function is different from the other two. The name of the searching function is different in this case which is called “senSeqSearch”. The idea of sentinel search is that we add a prompted number to the last position of the array, and make a program to keep finding whether the prompted number is included in the array until it reaches the last element of the array. However, we can not add extra elements that increase an array’s size since an array can not be expanded after instantiation. This means that in order to make this function work it is necessary to create a new array with one larger array size and copy the whole smaller-sized array to the larger one. The aforementioned procedures can be written as codes which can be seen in line: 9 and line: 10. Then, the very last element of the array arrayS (position: array size) is set to be equal to the prompted number (adding sentinel). Next, in line 13 to line 15, a while-loop is created with a variable “i” that is initiated

as 1. For instance, the loop will keep running and increment the value of “i” if the element in “i”th position of the array is not equal to the prompted number. Eventually, the `arryS[i]`, in this case, will be equal to “x” since we already append a sentinel in the tail of the `arryS`. This will give us the current position in `arryS` where the element is the same as the prompted number which will be checked later whether the result satisfies our goal. Finally, the if-else statement is used in order to check if the prompted value is actually included in the array initially (Line: 19 to Line:21). The condition will be satisfied meaning that the prompted number is included when the value of “i” is less than “n” and (&&) the value of `arryS[i]` is equal to “x”. Otherwise, the program will be terminated and print “The element not found”. Note that, “i should be less than n” means the condition should not be satisfied at the last position of the array since it is the sentinel that we have added after. If we do not consider this case, then any prompted number that is not initially included in the array will give the outcome as a satisfied condition where the program will print “Element found in the array (the last position)”.

test-data-10.txt

```
9 0 1 13 8 2 8 1 8 3
Enter an element:
8
The element found in array 4
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The output of this program is the same as the two of the aforementioned codes.

test-data-100.txt

```
chawinwaimaleongora-ek@Chawins-MacBook-Air Code % cd "/Users/chawinwaimaleongora-ek/downloads/2nd year/Spring semester/Algorithms and data structures/Code/" && javac Sentinels.java && java Sentinels
70 78 19 68 17 12 3 14 74 68 22 59 49 9 18 21 43 26 65
52 32 21 40 42 46 77 66 56 2 57 19 79 69 4 60 28 22 5
Enter an element:
50
The element found in array 46
chawinwaimaleongora-ek@Chawins-MacBook-Air Code %
```

The output of this program is the same as the two of the aforementioned codes.

How did I ensure the correctness of my program:

By inputting every possible number in different cases can help me to make sure that my code covers every condition. For example, I could try inputting different numbers that have already been included in the array or the numbers that are not included. One more thing is that I have tried to input

the numbers that are located in the last position of the array to make sure the conditions “ $i < n$ ” in functions are correct.