Sequential Search (Searching Supplement: 1 - 2)

A sequential search simply involves looking at each item in an array in turn until either the value being searched for is found or it can be determined that the value is not in the array. If the array is unsorted, then it is necessary to keep searching as long as the value is not found. However, if the array is sorted, it may be possible to quit searching without examining all of the elements in the array. If the array is sorted from low to high, the search can stop as soon as the current array value is greater than the value being searched for.

In the worst case, a sequential search will require looking at the whole array, so the time to search an array of size ***n*** is proportional to ***n***.

EXAMPLES

The following method searches through a list of data items and returns the position of the first occurrence of the *searchItem*, if found. The method returns -1 if not found. For this algorithm the array does not have to be in order.

```
public int indexOf (int list[], int searchItem) {
    for (int i = 0; i < list.length; i++)
        if (searchItem == list[i] )
            return i;
    return -1;
}
```

Searching a list that is known to be in order.

```
public int indexOf (int list[], int searchItem) {
    for (int i = 0; i < list.length && list[i] <= searchItem; i++)
        if (list[i] == searchItem)
            return i;
    return -1;
}
```

Calling A Method That Performs A Sequential Search

Calling a method that performs a searching function normally requires that you catch the return value in an independent variable that can then be tested to determine the results of the search. For example:

Given that we have the following variables declared and initialized as shown:

```
int[] list = { 3, 5, 8, 9, -14, 1, 17, 4, 22, 35, 6 };
int searchItem = <some integer value>;
```

The following program segment would search the array **list** for the value stored in **searchItem** and test the return value to determine if **searchItem** was found in the array:

```
int index = indexOf(list, searchItem);
if (index == -1)
    System.out.println(searchItem + " was found at index position " + index);
else
    System.out.println(searchItem + " was found at index position " + index);
```

If `searchItem` contained the value **9**, then `index` would be assigned the value **3** which is the index value where **9** is stored in the array `list`. If `searchItem` contained the value **2**, `index` would be assigned the value **-1** because `indexOf` returns **-1** when it doesn't find `searchItem` in the array `list`.

Lab 13 - ASSIGNMENT

Lab 13A - 70 points

OBJECTIVE

WAP to search through an array of integers and count the number of times an input values occurs. Input the list from a data file (**"lab13a.dat"**). Output the original array and the number of times the value was found. If the search value is not in the list output an appropriate message.

FIELD SUMMARY

- **int[ ] list** – an array of integers read from a data file.
- **int count** – the number of values stored in *list*.
- **int searchItem** – the value to be searched for.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `fileInput`, `kybdInput`, and `output`.
- **fileInput** – declare a Scanner object and read a series of integers from a data file storing them in an array of integers.
- **kybdInput** – declare a Scanner object and read an integer from the keyboard using an appropriate prompt.
- **find** – **Arguments:** an array of integers, the number of values stored in the array, and the value to search for. **Return:** an integer. Return the number of times the search item is found. This method should be **static**.
- **output** – output all the elements contained in ***list***, the item to be searched for, and the number of times the search item occurred in ***list***.
-

SAMPLE DATA FILE INPUT

6 4 8 5 8 4 9 3 1 7 2 4 8 9

SAMPLE KEYBOARD INPUT

Enter a number: **4**

SAMPLE OUTPUT

Original Data:
6 4 8 5 8 4 9 3 1 7 2 4 8 9
Search Item: 4
Number of occurrences: 3

Lab 13B - 80 points

OBJECTIVE

WAP to find the firsts occurrence of a value input from the keyboard in an array of integers and replaces it with a new value. Input the list from a data file (**"lab13b.dat"**). Input the search value and the replacement value from the keyboard. Output the original array, where the search item was found and the replacement value was inserted (if the search item was found) and the modified array. If the search value is not in the list output an appropriate message.

FIELD SUMMARY

- **int[ ] list** – an array of integers read from a data file.
- **int count** – the number of values stored in *list*.
- **int searchItem** – the value to be searched for.
- **int replaceValue** – the value to be used as a replacement value.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `fileInput`, `kybdInput`, and `output`.
- **fileInput** – declare a Scanner object and read a series of integers from a data file storing them in an array of integers.
- **kybdInput** – declare a Scanner object and read two integers from the keyboard using appropriate prompts.
- **output** – Output the original array, where the search item was found and the replacement value was inserted (if the search item was found) and the modified array . If the search value is not in the list output an appropriate message.
- **replace** – **Arguments:** an array of integers, the number of values stored in the array, the value to search for, and the replacement value. **Return:** the index position where the search value first occurred (and was replaced by the replacement value) or -1 if not found. This method should be **static**.

SAMPLE DATA FILE INPUT

1 2 4 5 4 5 5 4 4 7 6 4 8 4

SAMPLE KEYBOARD INPUT

Enter a number to replace: **4**
Enter a replacement value: **99**

SAMPLE OUTPUT

Original data: 1 2 4 5 4 5 5 4 4 7 6 4 8 4
The 4 found at position 2 was replaced with 99.
The modified list: 1 2 99 5 4 5 5 4 4 7 6 4 8 4

SAMPLE KEYBOARD INPUT
Enter a number to replace: **9** *Enter a replacement value:* **44**
SAMPLE OUTPUT
Original data: 1 2 4 5 4 5 5 4 4 7 6 4 8 4 **9 was not found in the list.**

Lab 13C - 90 points

OBJECTIVE

WAP to search through an array of characters and replace each occurrence of a character entered from the keyboard with a second character entered from the keyboard. Input the list from a data file (**"lab13c.dat"**). Output the original array, the number of times the search item was replaced with the replacement value and then output the array again after the changes are made.

FIELD SUMMARY

- **char[ ] list** – an array of integers read from a data file.
- **char searchItem** – the value to be searched for.
- **char replaceValue** – the new values that will replace the search item..

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `fileInput`, `kybdInput`, and `output`.
- **fileInput** – declare a Scanner object and read a series of characters from a data file storing them in an array of chars.
- **kybdInput** – declare a Scanner object and read two characters from the keyboard using appropriate prompts.
- **output** – output the original array, the item to be searched for, the value to use as a replacement value, and the altered array. If the search item is not in the array then display an appropriate message.
- **replaceAll** – **Arguments:** an array of characters, the number of values stored in the array, the value to be replaced and the replacement value. **Return:** an int. Return the number of times the search item was replaced with the replacement value. This method should be **static**.

SAMPLE DATA FILE INPUT

(*^$kv#$3#?+3*%94$9"}\#w38?)#4@**

SAMPLE KEYBOARD INPUT

Enter a search value: *****
Enter a replacement value: **-**

SAMPLE OUTPUT

Original Data:
(* ^ $ k v # $ 3 # ? + 3 * % 9 * * 4 $ 9 " } \ # w 3 8 ?) # 4 @
Search For: *
Replace With: _
Report: * was replaced 4 times with _.
Altered Data:
(_ ^ $ k v # $ 3 # ? + 3 _ % 9 _ _ 4 $ 9 " } \ # w 3 8 ?) # 4 @

NOTE: The String class contains a **toCharArray()** method that convert a string object into an array of chars. This method can be very useful when you need to read a collection of character from a data file or from the keyboard and store them in an array of characters.

```
char[] list;

Scanner reader = new Scanner…
list = reader.nextLine().toCharArray();
```

Notice that a loop is not required.

Lab 13D - 100 points

OBJECTIVE

WAP that reads in the name of a month from the keyboard and prints the number of days in that month in a non leap year. Use parallel arrays to store the two sets of data (the names of the months and the number of days in each corresponding month). If the name of the month entered from the keyboard is not valid, i.e. **Septober**, then output an appropriate message.

FIELD SUMMARY

- **String[ ] months** – an array of String storing the 12 months in a year.
- **int[] days** – an array of integer values storing the number of days in each month of the year. `months` and `days` are parallel arrays.
- **String month** – the month to search for.

METHOD SUMMARY

- **main** – instantiate an object of your class. Make method calls to `input`, and `output`.
- **input** – declare a Scanner object and read the name of a month from the keyboard using an appropriate prompt.
- **output** – output the number of days in the month input from the keyboard. If the user enters invalid data then display an appropriate message.
- **indexOf** – **Arguments:** an array of String (the months) and a String to search for (the month). The array is a full array so it is not necessary to pass the count as a parameter. **Return:** an int. Return the index position where the search string (month) was found in the array or -1 if it was not found. This method should be **static**.

NOTE: If you know the index number for any given month the number of days for that month can be obtained by referencing the parallel array containing the number of days. I.E. The index value for Feb is 1 and days[1] returns 28.

Months	Index	Days
"January"	0	31
"February"	1	28
"March"	2	31
"April"	3	30
"May"	4	31
"June"	5	30
"July"	6	31
"August"	7	31
"September"	8	30
"October"	9	31
"November"	10	30
"December"	11	31

SAMPLE KEYBOARD INPUT

Enter the name of a month: **September**

SAMPLE OUTPUT

September has 30 days

SAMPLE KEYBOARD INPUT

Enter the name of a month**: Septober**

SAMPLE OUTPUT

Septober is not a valid name of a month.