CS 111, Programming Fundamentals II
Project 2

Computer Science

For this project you will write code to search in a binary file. You will implement recursive binary search and use RandomAccessFile to open the binary file and search through it. This project is very similar to your lab #8, refer to it for hints if needed. Review the lecture slides about RandomAccessFile class and recursive binary search. You are given a binary file and two template classes, which contain comments for the implementation. You will add code to those files, each file can be implemented correctly with about 50-60 lines of code including comments and line breaks.

Getting Started

To get started download and save the java files **BinaryRecursiveSearch** and **Driver**. Also download the binary file named **BinFileShort**. There are 5000 "shorts" in the binary file. In Java a short takes up 2 bytes, so the size of the file is 10,000 bytes. This means you will have to read the first 2 bytes to get the first number, and then the 2 following bytes to get the second number.

The java file **BinaryRecursiveSearch** has a method declaration for ***binarySearch*** already completed, don't modify the header of the method. You will add code to the method ***binarySearch***, you don't need to add any other methods to this class. Take a look at the comments in the file. The java file **Driver**, has the main method already declared. Take a look at the comments in the file, this will be where you will test your recursive binary search implementation.

Refresh the lecture on **RandomAccessFile** and how the file pointer works. You will have to use the ***seek*** method in this assignment.

I. BinaryRecursiveSearch

In the file **BinaryRecursiveSearch**, inside of the method named ***binarySearch***, implement the recursive binary search. Refer to the lecture slides about the recursion for sample implementation. The method will take in a reference to a **RandomAccessFile (***file***)**, two bound positions (***left*** and ***right***), value to search for (***key***) and an integer value to count number of calls made (***callNumber***) to the method recursively. Initially the ***callNumber*** will be passed in as 0. As mentioned before, don't modify the header of the method.

In the method ***binarySearch***, print out the value for ***callNumber*** and then increment it by 1. Then calculate the middle point, adjust for 2 bytes per entry. For example if you wanted to get the 10^{th} number, then you would say that the index location is 10*2 and you would move the file pointer to byte number 20. Using the method ***seek***, move the file pointer to the middle index location and read the value at that location with the method ***readShort***. At this point you just need to add a few statements to complete the recursive binary search implementation. If the ***key*** is not in the file, then the return value should be -1. Refer to the lecture slides for the implementation.

Compile the class, fix any syntactical errors before moving on. If the file compiles successfully, you can run it. It will not do anything as there is no ***main*** method in this file.

II. Driver

The **Driver** class has the ***main*** method with some comments. This will be the driver for this application. You will add code to prompt the user and ask for a value to search for (***key***). Then you will call the method ***binarySearch*** and retrieve an index location. You will print out the index location of the ***key***, and prompt the user to see if they want to check for another value. The following is similar to the comments in inside of the file.

First create an instance of **Scanner** and an instance of **RandomAccessFile**. Open the file in read mode. You will need the **Scanner** to prompt the user for input. Add imports for those classes as needed. Also create an instance of **BinaryRecursiveSearch**, the class you have just finished implementing.

Add a do-while to the ***main*** method. Inside of the loop, prompt the user to enter a value to search for. Retrieve users input and call the method ***binarySearch***. When calling the ***binarySearch*** method, pass the reference to **RandomAccessFile** (***file***), ***left*** and ***right*** bounds, the ***key*** and 0 for the ***callNumber*** parameter. The ***left*** bound should be 0 and the ***right*** bound should 4999 (the "index" location of the last number).

Check if the returned value from the search algorithm, is a valid index location. Add print statements to display the index location of the ***key*** or to notify the user that the number (***key***) was not found. Then prompt the user again to see if they want to check for another ***key***. The user should be prompted regardless of the previous key being found or not. Have a look at the sample output in **Figure 1,** for a better understanding of the required print statements and how the do-while should operate.

Compile and run the program, search for the following keys in the file:

- 3 – index location 1
- 12 – not in the file
- 1234 – index location 423
- 7398 – index location 2499
- 11193 – index location 3749
- 12345 – not in the file

When running your code and searching for the keys mentioned above, take a screenshot of the output and save it. You will be required to upload a screenshot of your program. Your screenshot may contain other input values (***keys***) but must include at least three of the ones mentioned above, for example (3, 12, 1234) or (12, 11193, 12345).

III. Sample Output

```
 ----jGRASP exec: java Driver
Enter a value to search for :
11193
Binary Search call #1
Binary Search call #2
The value 11193 is as index location: 3749.

**********
Look for another value ? (Y/N)
Y
Enter a value to search for :
7398
Binary Search call #1
The value 7398 is as index location: 2499.

**********
Look for another value ? (Y/N)
Y
Enter a value to search for :
10
Binary Search call #1
Binary Search call #2
Binary Search call #3
Binary Search call #4
Binary Search call #5
Binary Search call #6
Binary Search call #7
Binary Search call #8
Binary Search call #9
Binary Search call #10
Binary Search call #11
Binary Search call #12
Binary Search call #13
The value 10 is not in the file.

**********
```

```
**********
Look for another value ? (Y/N)
Y
Enter a value to search for :
11
Binary Search call #1
Binary Search call #2
Binary Search call #3
Binary Search call #4
Binary Search call #5
Binary Search call #6
Binary Search call #7
Binary Search call #8
Binary Search call #9
Binary Search call #10
The value 11 is as index location: 3.

**********
Look for another value ? (Y/N)
N

 ----jGRASP: operation complete.
```

Figure 1 : Sample Output

IV. Upload your work to Canvas

Make sure that you upload the following files to the Project 2 assignment in your Canvas account:

Driver.java
BinaryRecursiveSearch.java
Screenshot of output containing at least 3 different keys (1 must be found in the file)

There will be additional files (the .class files, that you've generated when compiling your code) in your project 2 folder, but don't upload them to Canvas. Those files are not graded; they are just the byte code files that are used by the Java Virtual Machine.

Rubric

File / Lab	**Points**
Method *binarySearch* implemented recursively and to specification	30
The *main* method is implemented correctly and to specification	30
Both files compile/run and produce expected output	20
A screenshot of the program contains the requirements	10
Code is commented/formatted and variable names are descriptive	10
Total	100