AP COMPUTER SCIENCE A – HOMEWORK #10

Using `RecursiveBinarySearchStub.java` as a starting point, write a program `RecursiveBinarySearch.java` which performs a binary search, in a recursive fashion, on a list of sorted numbers. The file `RecursiveBinarySearchStub.java` is listed below.

```
import IO.ConsoleInput;
import IO.FileInput;
/**/
public class RecursiveBinarySearch {
  /**/
  private int numToFind, indexToFind, numIterations;
  private boolean numFound;
  private int [] list;
  /**/
  public static void main ( String [] arg ) {
    /**/
    int num;
    ConsoleInput ci;
    RecursiveBinarySearch rbs;
    String inputFile;
    /**/
    System.out.println();
    ci=new ConsoleInput();
    inputFile=ci.readString("input file name = ? ");
    num=ci.readInt(" number to find = ? ");
    ci.close();
    /**/
    rbs=new RecursiveBinarySearch(inputFile,num);
    rbs.recursiveBinarySearch();
    /*
     *  Put code here to print out the results of the search
     *  as shown on the homework handout.
     */
    return;
  }
  /**/
  public RecursiveBinarySearch ( String inpF, int n ) {
    /**/
    numToFind=n;
    indexToFind=-1;
    numFound=false;
    numIterations=0;
    readList(inpF);
  }
  /**/
  public void recursiveBinarySearch ( ) {
    /**/
    int len=list.length;
    /**/
    if ( ( numToFind < list[0] ) || ( numToFind > list[len-1] ) ) {
      return;
    }
    /**/
    recursion(0,len-1);
    /**/
    return;
  }
  /**/
  public boolean numberFound () {
    return numFound;
  }
  /**/
  public int getIndex() {
    return indexToFind;
  }
  /**/
  public int getNumIterations () {
    return numIterations;
  }
  /**/
```

AP COMPUTER SCIENCE A – HOMEWORK #10

```
  private void readList ( String fn ) {
    /**/
    int len;
    FileInput fi;
    /**/
    fi=new FileInput(fn);
    len=fi.readLineInt();
    list=new int [len];
    fi.readLine();
    for ( int i=0; i<len; ++i ) list[i]=fi.readLineInt();
    fi.close();
    /**/
    return;
  }
  /**/
  private void recursion ( int b, int e ) {
    /**/
    ++numIterations;
    /*
     *  Put code here to perform the search.
     *  Note: this method should call itself once.
     */
  }
}
```

1) Add the needed code to method `recursion ( int b, int e )`. Follow the procedure explained in the notes in the `4_recursive_search` folder. In particular, you should:

- **a)** first define $m = (b + e) / 2$
- **b)** then check if `list[m]` is the number we are looking for (in which case the process is done: be sure to set `numFound` and `indexToFind`)
- **c)** then check if the number has not been found (and the process is over), *i.e.*, check `if ( b == m ) || ( m == e )`
- **d)** Finally redefine `b` or `e`, and then call `recursion(b,e)`

2) Add code to where indicated in the `main` method to print out the results of the search. You should use the file `list_rbs.txt` as input to your program, and its output should look something like, *e.g.*,

```
input file name = ? list_rbs.txt
 number to find = ? 273

              number = 273
number of iterations = 5
        number found = true
               index = 27
```

or

```
input file name = ? list_rbs.txt
 number to find = ? 275

              number = 275
number of iterations = 7
        number found = false
               index = -1
```

where bold print is user input.