**Searching Algorithms: Recursive Binary Search**

Either build your own array for this search, consisting of 20 elements or use the following.**int** OriginalArray[] = {4,10,6,1,8,10,9,12,14,6,15,6,7,10,8,2,7,3,9,1};

1. Please develop a Recursive Binary Search Method in your eclipse file, and then copy and paste it into the box below. Identify the amount of instances of the selected value which exist within the array.

|  |
| --- |
| public class RecursiveSearchAssignment {  static Scanner *input* = new Scanner(System.***in***);    static int[] insertionSort(int[] a) { // array sorter  int itemToInsert, j;  boolean reset;  for(int k = 1; k < a.length; k++) {  itemToInsert = a[k];  j = k-1;  reset = false;  while((j >= 0) && !reset) {  if (itemToInsert < a[j] ) {  a[j + 1] = a[j];  j--;  if(j == -1)  a[0] = itemToInsert;  }  else {  reset = true;  a[j + 1] = itemToInsert;  }  }  }  // **XXX** debug  System.***out***.println("\n");  for(int n = 0; n < a.length; ++n) {  System.***out***.print(a[n]+ " ");  }  return a;  }    public static int recursiveBinary(int[] a, int low, int high, int numIn) {  int mid = 0;  int count = 0;    while(low <= high) {    //Find mid point  mid = (low + high)/2;    //is value at mid equal to target  //return mid  //index++;  if(a[mid] == numIn) {  ++count; // keeps track of number instances  for(int k = mid+1; k <= a.length-1; ++k) {  if(a[k] == numIn)  count++;  }  for(int k = mid-1; k > -1; --k) {  if(a[k] == numIn)  count++;  }  return count;  }    //is it greater?  //recursively call function but with low and mid-1 for high  else if(a[mid] > numIn) {  return *recursiveBinary*(a, low, mid-1, numIn);  }    //is it less  //recursively call function but with mid+1 for low and high  else {  return *recursiveBinary*(a, mid+1, high, numIn);  }  }  return 0;  }    public static void main(String[] args) {  int[] OriginalArray = {4,10,6,1,8,10,9,12,14,6,15,6,7,10,8,2,7,3,9,1};    System.***out***.println("Input number to search: "); // input target number  int numIn = *input*.nextInt();    int[] sortedArray = *insertionSort*(OriginalArray);  int count = *recursiveBinary*(sortedArray, 0, sortedArray.length-1, numIn); // new sorted array put into recursive search  // a low high target  System.***out***.println("\n\nThere are " + count + " instances of this number in the array");  } |

1. Please add developer comments to the various elements of your search. Outline how the elements which make it Recursive, and how it differs from the iterative Search you did previously.
2. In the box below, please provide a written description of how this search moves through an array sorting it. Explain in detail the elements which make it recursive. (base case, changing parameters, loops, etc.)  
     
   What is the benefit of setting up your search recursively, rather than iteratively? Please explain.

|  |
| --- |
| Doesn’t require as much code to loop back through and can just re-use itself to do more counting |