**Searching Algorithms: Linear Search**

A Searching Algorithm will allow you to identify if a specific value exists within a given array of information. The Linear Sort is the simpler of the 2 we will be looking at. Either build your own array for these searches, 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 Linear 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.

|  |
| --- |
| static void linearSearch(int[] a) {  System.***out***.println("(Linear) Input a number to search: ");  int numIn = *input*.nextInt();    int i=0;  boolean indexIdentified = false;  int numAmt=0;    for(i = 0; i < a.length; ++i) { // runs through and identifies matching numbers and prints each one with its respective index  if(numIn == a[i]) {  System.***out***.printf("The number is at index %d or position %d\n", i, i+1);  indexIdentified = true;  ++numAmt; // number of instances  }  }    if(i == a.length && indexIdentified == false) { // if it reaches the end of the array and no numbers have been identified  System.***out***.println("That number is not in the array");  }  if(indexIdentified == true) { // if numbers have been identified, print the amount  System.***out***.printf("There are %d instance(s) of this number in the array", numAmt);  }  } |

1. Please add developer comments to the various elements of your search. These comments should briefly outline what each line of code is for, and what it accomplishes.
2. In the box below, please provide a written description of how this search moves through an array identifying the selected values. This Description should highlight the process, it does not need to outline every single adjustment that is made.

|  |
| --- |
| An integer value is inputted and then compared with every element of the array procedurally, if a match is found then it prints the index and adds one to the identified indexes counter. At the end it prints the amount of identified indexes |

1. Can you think of a situation when this searching method would be the most logical method to use, please explain why?

|  |
| --- |
| When used with strings, this can be helpful for identifying a name in a list of names to verify if that name is on the list. An example of this would be verifying login credentials |

**Searching Algorithms: Binary Search**

The Binary Search is slightly more complicated, as it will not work unless it has a fully sorted array. 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 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.

|  |
| --- |
| static int binarySearch(int[] a) {  // sorting    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;  }  }  }    System.***out***.println("\n");  for(int l = 0; l < a.length; ++l) {  System.***out***.print(a[l]+ " ");  }    // binary search with sorted array    System.***out***.println("\n\n(Binary) Input a number to search: ");  int numIn = *input*.nextInt();    int mid = 0;    int low = 0;  int high = a.length-1;  int count = 0;  while(low <= high) {    mid = (low + high)/2;    if(a[mid] == numIn) { // if a exact match is found, print its index  ++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;  }    if(a[mid] > numIn) {  high = mid-1; // if the target number is lower than the current index, set the middle of the array to be the highest available index effectively eliminating the upper half of the array  }  else {  low = mid+1; // vice versa eliminate the lower half of the array  }  }  return 0;  } |

1. Please add developer comments to the various elements of your search. These comments should briefly outline what each line of code is for, and what it accomplishes.
2. In the box below, please provide a written description of how this search moves through an array sorting it.   
     
   In order to identify multiple instances of the selected value, what did you need to add to the binary search?

|  |
| --- |
| A for loop was used to shift the index over and count from each side to the other side and stop when it reaches the end, then return the amount of instances counted. |

1. Can you think of a situation when this searching method would be the most logical method to use, please explain why?

|  |
| --- |
| Binary search works by sorting an array linearly and then comparing the number being searched to half of the array and eliminating one half of it and then doing the same process with the other half until the number is identified. In most cases this can be very efficient because It eliminates a large range of numbers in the first search and the proceeding searches which allows for few searches to identify the number. |