**Java Search And Sort Algorithms -L2-Assignment**

1. **Task to implement binary search using Recursive Algorithm :**

**package** Sample;

**public** **class** MyRecursiveBinarySearch {

**public** **static** **int** recursiveBinarySearch(**int**[]

sortedArray, **int** start, **int** end, **int** key)

{

**if** (start < end) {

**int** middle = start + (end - start) / 2;

**if** (key < sortedArray[middle]) {

**return** *recursiveBinarySearch*(sortedArray, start, middle, key);

} **else** **if** (key > sortedArray[middle]) {

**return** *recursiveBinarySearch*(sortedArray, middle+1, end , key);

} **else** {

**return** middle;

}

}

**return** -(start + 1);

}

**public** **static** **void** main(String[] args) {

**int**[] arr1 = {2,45,234,567,876,900,976,999};

**int** index = *recursiveBinarySearch*(arr1,0,arr1.length,45);

System.*out*.println("Found 45 at "+index+" index");

index = *recursiveBinarySearch*(arr1,0,arr1.length,999);

System.*out*.println("Found 999 at "+index+" index");

index = *recursiveBinarySearch*(arr1,0,arr1.length,876);

System.*out*.println("Found 876 at "+index+" index");

}

}

**Output:**

Found 45 at 1 index

Found 999 at 7 index

Found 876 at 4 index

1. **Task to implement merge sort in Java:**

**package** Sample;

**public** **class** MyMergeSort

{

**private** **int**[] array;

**private** **int**[] tempMergArr;

**private** **int** length;

**public** **static** **void** main(String a[]){

**int**[] inputArr = {45,23,11,89,77,98,4,28,65,43};

MyMergeSort mms = **new** MyMergeSort();

mms.sort(inputArr);

**for**(**int** i:inputArr){

System.*out*.print(i);

System.*out*.print(" ");

}

}

**public** **void** sort(**int** inputArr[]) {

**this**.array = inputArr;

**this**.length = inputArr.length;

**this**.tempMergArr = **new** **int**[length];

doMergeSort(0, length - 1);

}

**private** **void** doMergeSort(**int** lowerIndex, **int** higherIndex) {

**if** (lowerIndex < higherIndex) {

**int** middle = lowerIndex + (higherIndex - lowerIndex) / 2;

doMergeSort(lowerIndex, middle);

doMergeSort(middle + 1, higherIndex);

mergeParts(lowerIndex, middle, higherIndex);

}

}

**private** **void** mergeParts(**int** lowerIndex, **int** middle, **int** higherIndex) {

**for** (**int** i = lowerIndex; i <= higherIndex; i++) {

tempMergArr[i] = array[i];

}

**int** i = lowerIndex;

**int** j = middle + 1;

**int** k = lowerIndex;

**while** (i <= middle && j <= higherIndex) {

**if** (tempMergArr[i] <= tempMergArr[j]) {

array[k] = tempMergArr[i];

i++;

} **else** {

array[k] = tempMergArr[j];

j++;

}

k++;

}

**while** (i <= middle) {

array[k] = tempMergArr[i];

k++;

i++;

}

}

}

**Output:**

4 11 23 28 43 45 65 77 89 98

1. **Task to implement bubble sort in java :**

**package** Sample;

**public** **class** MyBubbleSort {

**public** **static** **void** bubble\_srt(**int** array[]) {

**int** n = array.length;

**int** k;

**for** (**int** m = n; m >= 0; m--) {

**for** (**int** i = 0; i < n - 1; i++) {

k = i + 1;

**if** (array[i] > array[k]) {

*swapNumbers*(i, k, array);

}

}

*printNumbers*(array);

}

}

**private** **static** **void** swapNumbers(**int** i, **int** j, **int**[] array) {

**int** temp;

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

**private** **static** **void** printNumbers(**int**[] input) {

**for** (**int** i = 0; i < input.length; i++) {

System.*out*.print(input[i] + ", ");

}

System.*out*.println("\n");

}

**public** **static** **void** main(String[] args) {

**int**[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };

*bubble\_srt*(input);

}

}

**Output:**

2, 4, 6, 0, 1, 9, 12, 23, 34,

2, 4, 0, 1, 6, 9, 12, 23, 34,

2, 0, 1, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,

0, 1, 2, 4, 6, 9, 12, 23, 34,