Assignment 1
DAA
21052030
CSE 27

MERGE SORT:

Pseudo Code:

```
MergeSort(arr, left, right):
    if left < right:
        middle = (left + right) / 2
        MergeSort(arr, left, middle)       // Recursively sort left subarray
        MergeSort(arr, middle + 1, right)  // Recursively sort right subarray
        Merge(arr, left, middle, right)    // Merge the two sorted subarrays

Merge(arr, left, middle, right):
    n1 = middle - left + 1
    n2 = right - middle

    // Create temporary arrays to hold the left and right subarrays
    leftArr[n1]
    rightArr[n2]

    // Copy data to temporary arrays
    for i = 0 to n1 - 1:
        leftArr[i] = arr[left + i]
    for j = 0 to n2 - 1:
        rightArr[j] = arr[middle + 1 + j]

    // Merge the temporary arrays back into the original array
    i = 0     // Initial index of left subarray
    j = 0     // Initial index of right subarray
    k = left  // Initial index of merged subarray

    while i < n1 and j < n2:
        if leftArr[i] <= rightArr[j]:
            arr[k] = leftArr[i]
            i = i + 1
        else:
            arr[k] = rightArr[j]
```

```
        j = j + 1
    k = k + 1

// Copy the remaining elements of leftArr[], if any
while i < n1:
    arr[k] = leftArr[i]
    i = i + 1
    k = k + 1

// Copy the remaining elements of rightArr[], if any
while j < n2:
    arr[k] = rightArr[j]
    j = j + 1
    k = k + 1
```

Function Call	Remark	Array
MergeSort(arr, left, right):	Algorithm Called	Arr = [5,4,2,7], left=1, right=4
if left < right:	Check l<r	1<4
middle = (left + right) / 2	Determine mid point of array	Middle = (1+4)/2 = 2
MergeSort(arr, left, middle)	Recursively call first half	arr=[5, 4] -> arr=[5] & arr=[4]
MergeSort(arr, middle + 1, right)	Recursively call second half	arr=[2, 7] -> arr=[2] & arr=[7]
Merge(arr, left, middle, right)	Merge algorithm called	([5,4,2,7], 1, 2, 4)-> ([5,4], 1, 1, 2)-> ([5], 1, 1,1) ([4], 2, 2, 2) ([2,7], 3, 3, 4)-> ([2], 3, 3, 3) ([7], 4, 4, 4)
n1 = middle - left + 1	Left sub array size	(n1=2)-> (n1=1)-> (n1=1) (n1=1) (n1=1)-> (n1=1) (n1=1)

n2 = right - middle	Right Sub Array size	(n2=2)-> (n2=1)-> (n2=0) (n2=0) (n2=1)-> (n2=0) (n2=0)
leftArr[n1]	Left Arr	
rightArr[n2]	Right Arr	
for i = 1 to n1 : leftArr[i] = arr[left + i]	Fill Temp Left Array	(lefArr=[5,4])-> (lefArr=[5])-> (lefArr=[5]) (lefArr=[4]) (lefArr=[2])-> (lefArr=[2]) (lefArr=[7])
for j = 1 to n2 : rightArr[j] = arr[middle + 1 + j]	Fill Temp Right Array	(rightArr=[2,7])-> (rightArr=[4])-> (rightArr=[]) (rightArr=[]) (rightArr=[7])-> (rightArr=[]) (rightArr=[])
i = 1	Initialize iterator of left	
J = 1	Initialize iterator of right	
k = left	Initialize iterator of merged array	

while i < n1 and j < n2:
if leftArr[i] <= rightArr[j]:
 	
arr[k] = leftArr[i]
i = i + 1
else:
arr[k] = rightArr[j]
j = j + 1
k = k + 1
	Compare and Merge all the elements from left and right into the merged array	[4, 5] [2, 7]

while i < n1: arr[k] = leftArr[i] i = i + 1	Copy all remaining elements from left arr	[2, 4, 5, 7]

k = k + 1		
while j < n2: **arr[k] = rightArr[j]** **j = j + 1** **k = k + 1**	**Copy all remaining elements from right arr**	**[2, 4, 5, 7]**