UCS1403 Design and Analysis of Algorithms
AY: 2021-22

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

DAA Lab – Assignment 4

Sorting Algorithms using Divide and Conquer strategy

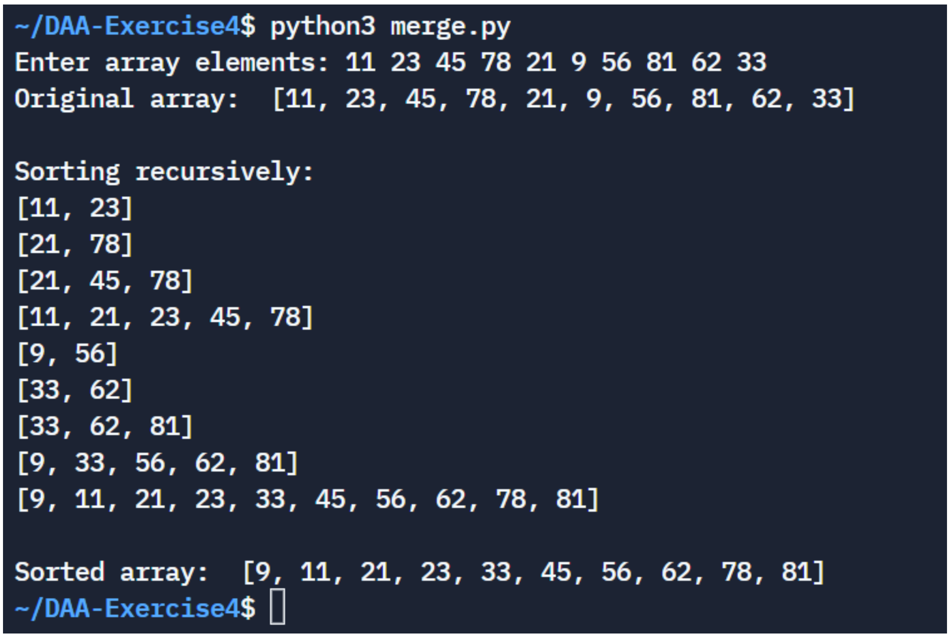
1. Implement Merge Sort and take a snapshot of the function calling stack and recursive depth.

Code:

```
#Python program to implement merge sort recursively

def mergeSort(arr):
	if (len(arr) > 1):

		mid = len(arr)//2
		left = arr[:mid]
		right = arr[mid:]

		mergeSort(left)
		mergeSort(right)

		i = j = k = 0
		l,r = len(left),len(right)

		while (i < l and j < r):
			if left[i] < right[j]:
				arr[k] = left[i]
				i += 1
			else:
				arr[k] = right[j]
				j += 1
			k += 1

		while i < l:
			arr[k] = left[i]
			i += 1
			k += 1

		while j < r:
			arr[k] = right[j]
			j += 1
			k += 1

		print(arr)

#input
arr = list(map(int,input("Enter array elements: ").split()))
print("Original array: ",arr,"\n")

#recursive calls
print("Sorting recursively: ")
mergeSort(arr)
print()

#output
```

```
print("Sorted array: ",arr)
```

Output:



```
~/DAA-Exercise4$ python3 merge.py
Enter array elements: 11 23 45 78 21 9 56 81 62 33
Original array:  [11, 23, 45, 78, 21, 9, 56, 81, 62, 33]

Sorting recursively:
[11, 23]
[21, 78]
[21, 45, 78]
[11, 21, 23, 45, 78]
[9, 56]
[33, 62]
[33, 62, 81]
[9, 33, 56, 62, 81]
[9, 11, 21, 23, 33, 45, 56, 62, 78, 81]

Sorted array:  [9, 11, 21, 23, 33, 45, 56, 62, 78, 81]
~/DAA-Exercise4$ 
```

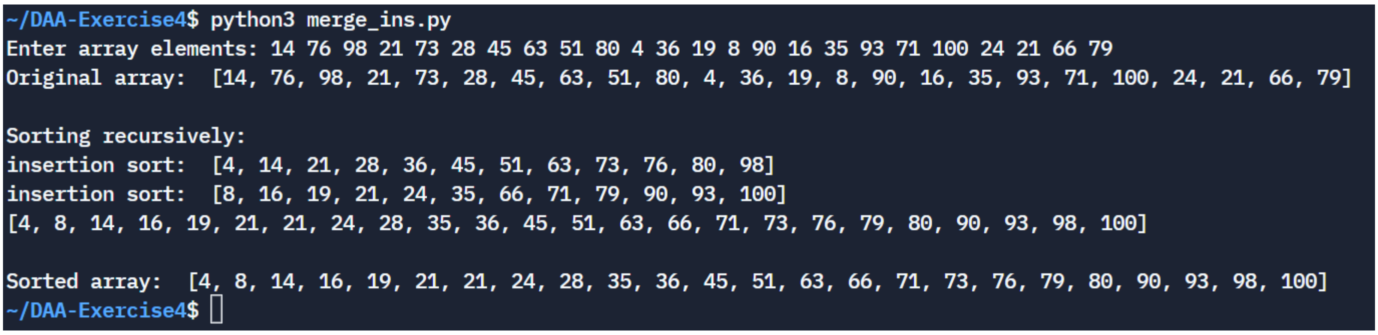
2. Implement Merge Sort and call insertion sort for n=12, instead of recursive calls.

Code:

```
#Python program to implement merge sort with insertion sort for n = 12

#insertion sort
def insertionSort(arr):
      for i in range(1,len(arr)):
        el = arr[i]
        j = i-1
        while (j>=0 and el<arr[j]):
          arr[j+1] = arr[j]
          j-=1
        arr[j+1] = el

#merge sort
def mergeSort(arr):
      if (len(arr) == 12):
           insertionSort(arr)
           print("insertion sort: ",arr)

      elif (len(arr) > 1):

           mid = len(arr)//2
           left = arr[:mid]
           right = arr[mid:]

           mergeSort(left)
           mergeSort(right)

           i = j = k = 0
           l,r = len(left),len(right)

           while (i < l and j < r):
                 if left[i] < right[j]:
                      arr[k] = left[i]
                      i += 1
                 else:
                      arr[k] = right[j]
                      j += 1
                 k += 1

           while i < l:
                 arr[k] = left[i]
                 i += 1
                 k += 1

           while j < r:
                 arr[k] = right[j]
                 j += 1
                 k += 1

           print(arr)
```

```
#input
arr = list(map(int,input("Enter array elements: ").split()))
print("Original array: ",arr,"\n")

#recursive calls
print("Sorting recursively: ")
mergeSort(arr)
print()

#output
print("Sorted array: ",arr)
```

Output:



```
~/DAA-Exercise4$ python3 merge_ins.py
Enter array elements: 14 76 98 21 73 28 45 63 51 80 4 36 19 8 90 16 35 93 71 100 24 21 66 79
Original array:  [14, 76, 98, 21, 73, 28, 45, 63, 51, 80, 4, 36, 19, 8, 90, 16, 35, 93, 71, 100, 24, 21, 66, 79]

Sorting recursively:
insertion sort:  [4, 14, 21, 28, 36, 45, 51, 63, 73, 76, 80, 98]
insertion sort:  [8, 16, 19, 21, 24, 35, 66, 71, 79, 90, 93, 100]
[4, 8, 14, 16, 19, 21, 21, 24, 28, 35, 36, 45, 51, 63, 66, 71, 73, 76, 79, 80, 90, 93, 98, 100]

Sorted array:  [4, 8, 14, 16, 19, 21, 21, 24, 28, 35, 36, 45, 51, 63, 66, 71, 73, 76, 79, 80, 90, 93, 98, 100]
~/DAA-Exercise4$ 
```

UCS1403 Design and Analysis of Algorithms
AY: 2021-22

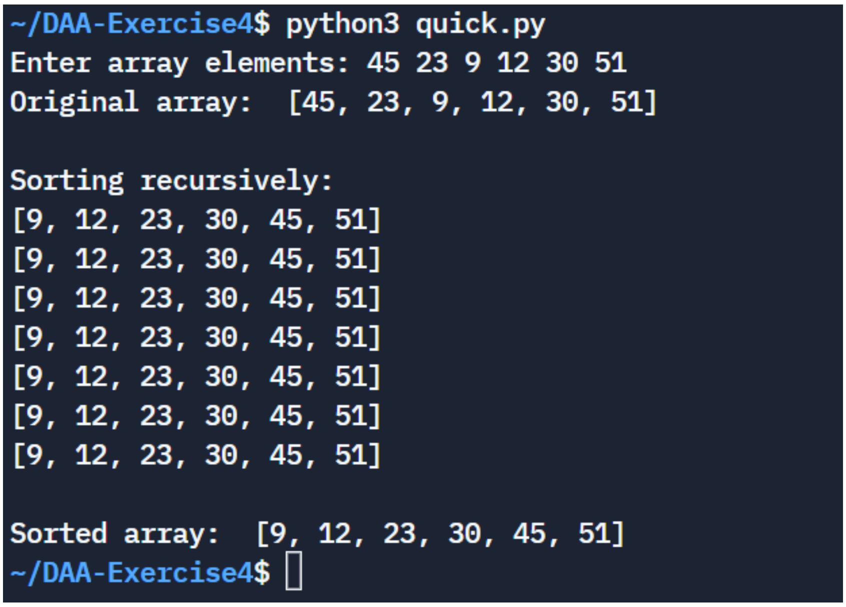
Name: *Krithika Swaminathan*
Reg. no.: *205001057*

3. Implement Quicksort.

Code:

```
#Python program implementing Quick Sort

def partition(arr,low,high):
      i = (low-1)
      pivot = arr[high]

      for j in range(low,high):
            if (arr[j] <= pivot):
                  i = i+1
                  arr[i], arr[j] = arr[j], arr[i]

      arr[i+1], arr[high] = arr[high], arr[i+1]
      return (i+1)

def quickSort(arr, low, high):
      if (len(arr) == 1):
            return arr

      if low < high:
            p = partition(arr, low, high)

            quickSort(arr, low, p-1)
            quickSort(arr, p+1, high)

      print(arr)

#input
arr = list(map(int,input("Enter array elements: ").split()))
print("Original array: ",arr,"\n")

#recursive calls
print("Sorting recursively: ")
quickSort(arr,0,len(arr)-1)
print()

#output
print("Sorted array: ",arr)
```

<u>*Output:*</u>



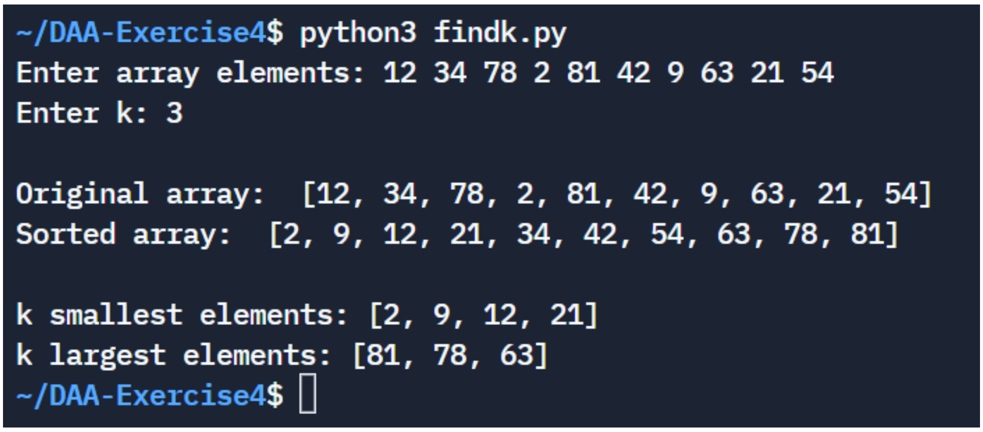
```
~/DAA-Exercise4$ python3 quick.py
Enter array elements: 45 23 9 12 30 51
Original array:  [45, 23, 9, 12, 30, 51]

Sorting recursively:
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]
[9, 12, 23, 30, 45, 51]

Sorted array:  [9, 12, 23, 30, 45, 51]
~/DAA-Exercise4$ 
```

4. Find the Kth Smallest/Largest Element in Unsorted Array.

Code:

```
#Python program to find the k smallest/largest elements in an unsorted array

def partition(arr,low,high):
      i = (low-1)
      pivot = arr[high]

      for j in range(low,high):
            if (arr[j] <= pivot):
                  i = i+1
                  arr[i], arr[j] = arr[j], arr[i]

      arr[i+1], arr[high] = arr[high], arr[i+1]
      return (i+1)

def quickSort(arr, low, high):
      if (len(arr) == 1):
            return arr

      if low < high:
            p = partition(arr, low, high)

            quickSort(arr, low, p-1)
            quickSort(arr, p+1, high)

#input
arr = list(map(int,input("Enter array elements: ").split()))
k = int(input("Enter k: "))

print("\nOriginal array: ",arr)

#sorting the array
quickSort(arr,0,len(arr)-1)
print("Sorted array: ",arr,"\n")

#finding k smallest elements
print("k smallest elements:",arr[:k+1])

#finding k largest elements
print("k largest elements:",arr[:-(k+1):-1])
```

Output:



```
~/DAA-Exercise4$ python3 findk.py
Enter array elements: 12 34 78 2 81 42 9 63 21 54
Enter k: 3

Original array:  [12, 34, 78, 2, 81, 42, 9, 63, 21, 54]
Sorted array:  [2, 9, 12, 21, 34, 42, 54, 63, 78, 81]

k smallest elements: [2, 9, 12, 21]
k largest elements: [81, 78, 63]
~/DAA-Exercise4$ 
```