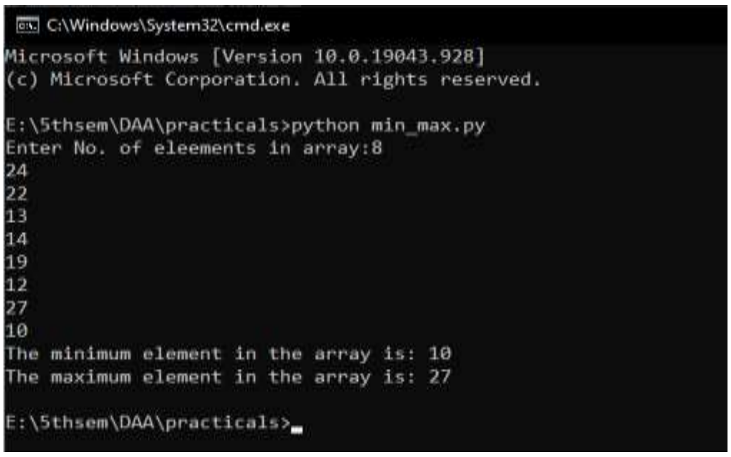
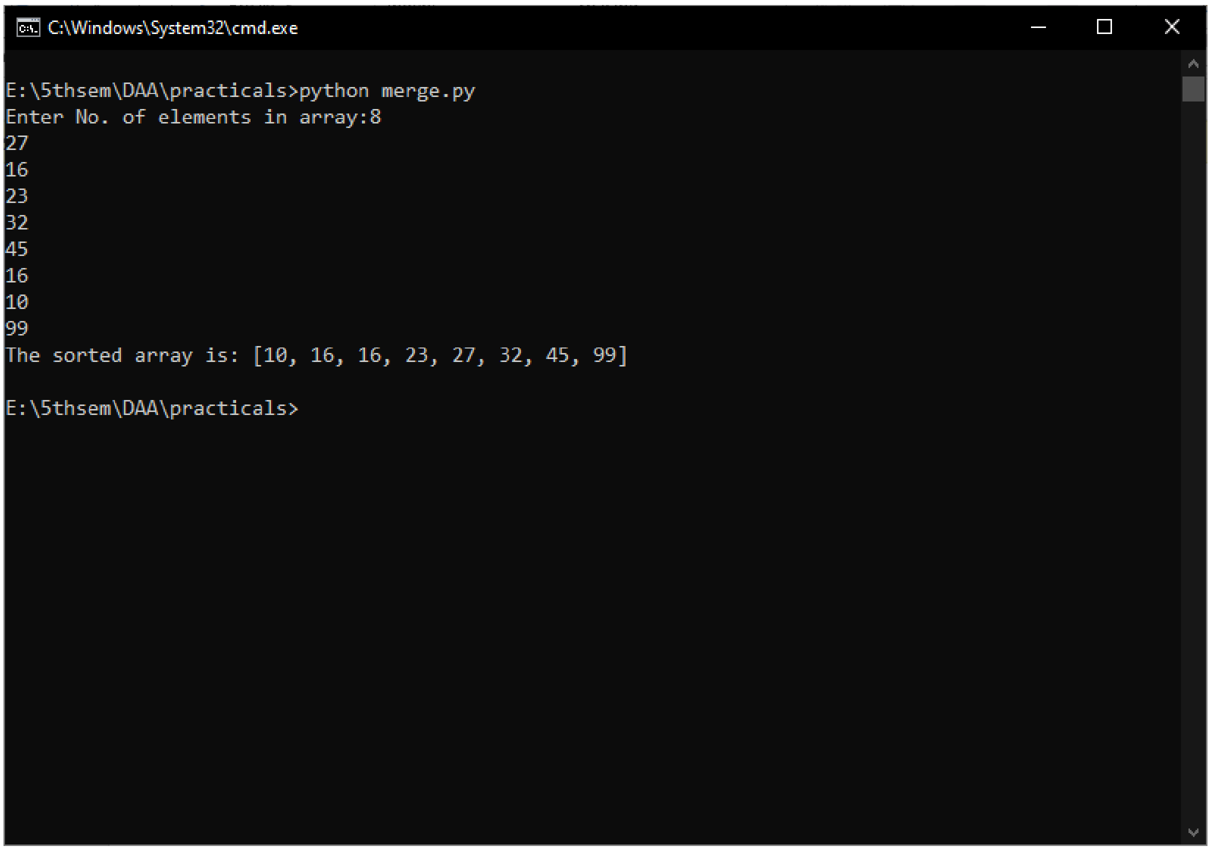
```
def min_max(arr, left, right):
    # Base case: when the subarray has only one element
    if left == right:
        return arr[left], arr[left]
    # Base case: when the subarray has two elements
    if right == left + 1:
        if arr[left] < arr[right]:
            return arr[left], arr[right]
        else:
            return arr[right], arr[left]

    # Find the middle index of the array
    mid = (left + right) // 2

    # Recursively find the min and max in the left and right halves
    min_left, max_left = min_max(arr, left, mid)
    min_right, max_right = min_max(arr, mid + 1, right)

    # Combine the results
    final_min = min(min_left, min_right)
    final_max = max(max_left, max_right)

    return final_min, final_max

def find_min_max(arr):
    if not arr:
        raise ValueError("Array is empty")
    return min_max(arr, 0, len(arr) - 1)

# Example usage
if __name__ == "__main__":
    n=int(input('Enter No. of eleements in array:'))
    array =[]
    for i in range(n):
        element=int(input())
        array.append(element)
    min_val, max_val = find_min_max(array)
    print(f"The minimum element in the array is: {min_val}")
    print(f"The maximum element in the array is: {max_val}")
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

E:\5thsem\DAA\practicals>python min_max.py
Enter No. of eleements in array:8
24
22
13
14
19
12
27
10
The minimum element in the array is: 10
The maximum element in the array is: 27

E:\5thsem\DAA\practicals>
```

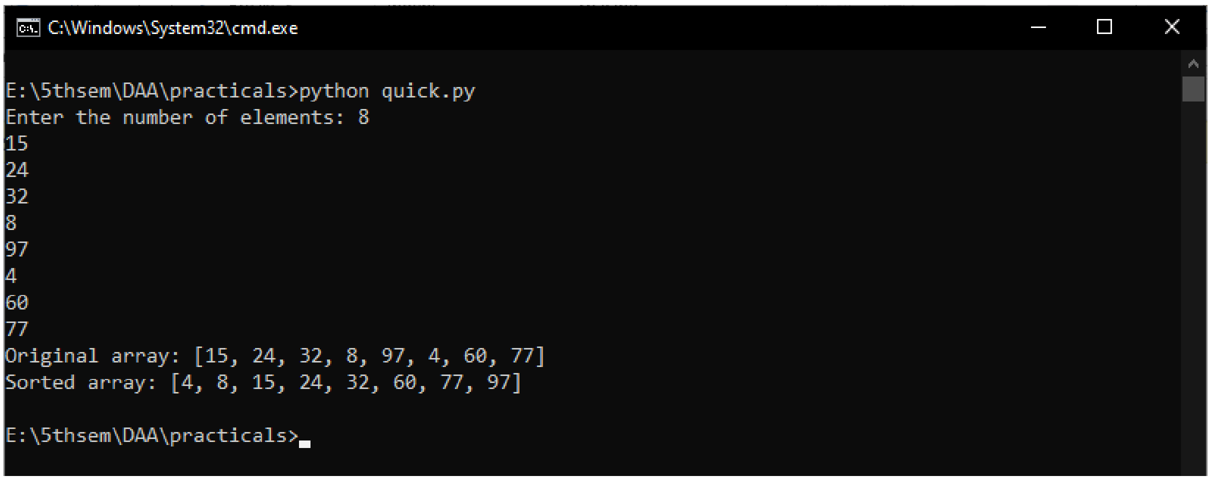
```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    # Find the middle point of the array
    mid = len(arr) // 2

    # Recursively sort the two halves
    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])

    # Merge the sorted halves
    return merge(left_half, right_half)

def merge(left, right):
    sorted_array = []
    i = j = 0

    # Merge the arrays while both have elements
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            sorted_array.append(left[i])
            i += 1
        else:
            sorted_array.append(right[j])
            j += 1

    # If there are remaining elements in left array
    while i < len(left):
        sorted_array.append(left[i])
        i += 1

    # If there are remaining elements in right array
    while j < len(right):
        sorted_array.append(right[j])
        j += 1

    return sorted_array

# Example usage
if __name__ == "__main__":
    n=int(input('Enter No. of elements in array:'))
    array = []
    for i in range(n):
        element=int(input())
        array.append(element)
    sorted_array = merge_sort(array)
    print(f"The sorted array is: {sorted_array}")
```



C:\Windows\System32\cmd.exe

```
E:\5thsem\DAA\practicals>python merge.py
Enter No. of elements in array:8
27
16
23
32
45
16
10
99
The sorted array is: [10, 16, 16, 23, 27, 32, 45, 99]

E:\5thsem\DAA\practicals>
```

```
def quick_sort(arr, low, high):
    if low < high:
        # Partition the array and get the pivot index
        pi = partition(arr, low, high)

        # Recursively apply quick_sort to the sub-arrays
        quick_sort(arr, low, pi - 1)
        quick_sort(arr, pi + 1, high)

def partition(arr, low, high):
    # Choose the pivot element, here we choose the last element in the array
    pivot = arr[high]
    i = low - 1

    # Rearrange the array by placing elements less than the pivot before the pivot
    for j in range(low, high):
        if arr[j] < pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]

    # Place the pivot element in the correct position
    arr[i + 1], arr[high] = arr[high], arr[i + 1]

    return i + 1

# Example usage
if __name__ == "__main__":
    n = int(input("Enter the number of elements: "))
    sample_array = []
    for i in range(n):
        element = int(input())
        sample_array.append(element)
    print("Original array:", sample_array)
    quick_sort(sample_array, 0, len(sample_array) - 1)
    print("Sorted array:", sample_array)
```



```
E:\5thsem\DAA\practicals>python quick.py
Enter the number of elements: 8
15
24
32
8
97
4
60
77
Original array: [15, 24, 32, 8, 97, 4, 60, 77]
Sorted array: [4, 8, 15, 24, 32, 60, 77, 97]

E:\5thsem\DAA\practicals>
```