Name: Anjali Pingle Roll no:30

Class: MSc CS Part I Subject: Algorithm

Algorithm Mini Project

Index

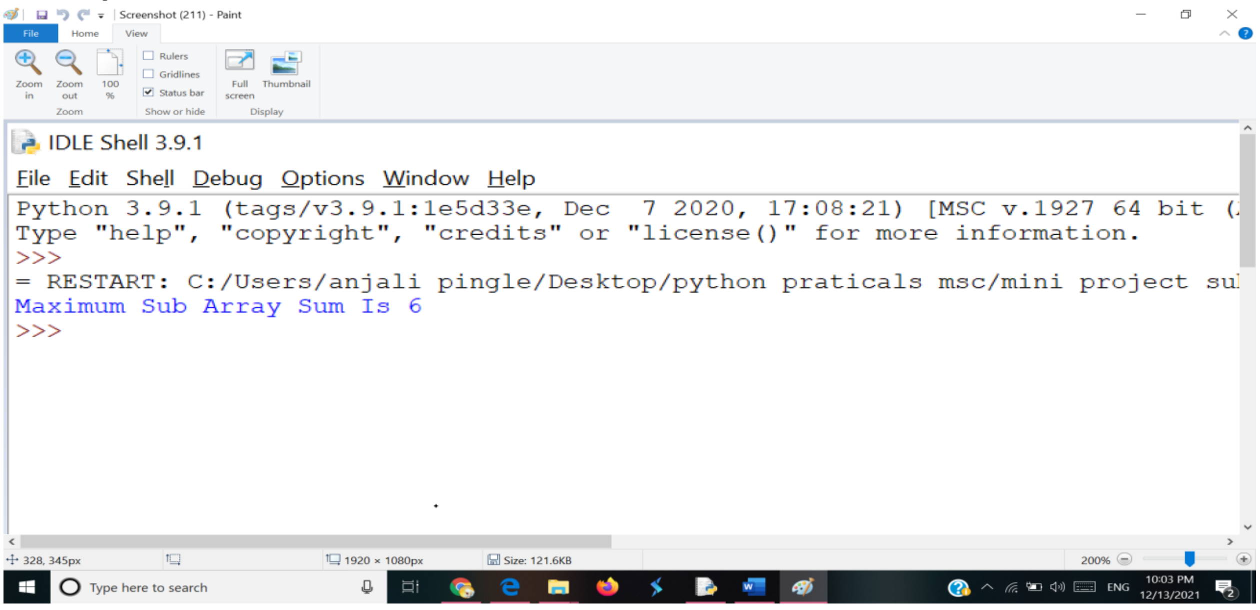
Maximum Subarray Problem

Aim: Write a Python program to implement the maximum subarray problem

Input:

```
def maxSubArraySum(arr,size):

    max_till_now = arr[0]
    max_ending = 0

    for i in range(0, size):
        max_ending = max_ending + arr[i]
        if max_ending < 0:
            max_ending = 0


        elif (max_till_now < max_ending):
            max_till_now = max_ending

    return max_till_now

arr = [-2, -3, 4, -1, -2, 5, -3]
print("Maximum Sub Array Sum Is" , maxSubArraySum(arr,len(arr)))
```

Output:



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7 2020, 17:08:21) [MSC v.1927 64 bit (
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/anjali pingle/Desktop/python praticals msc/mini project su
Maximum Sub Array Sum Is 6
>>>
```

Merge Sort

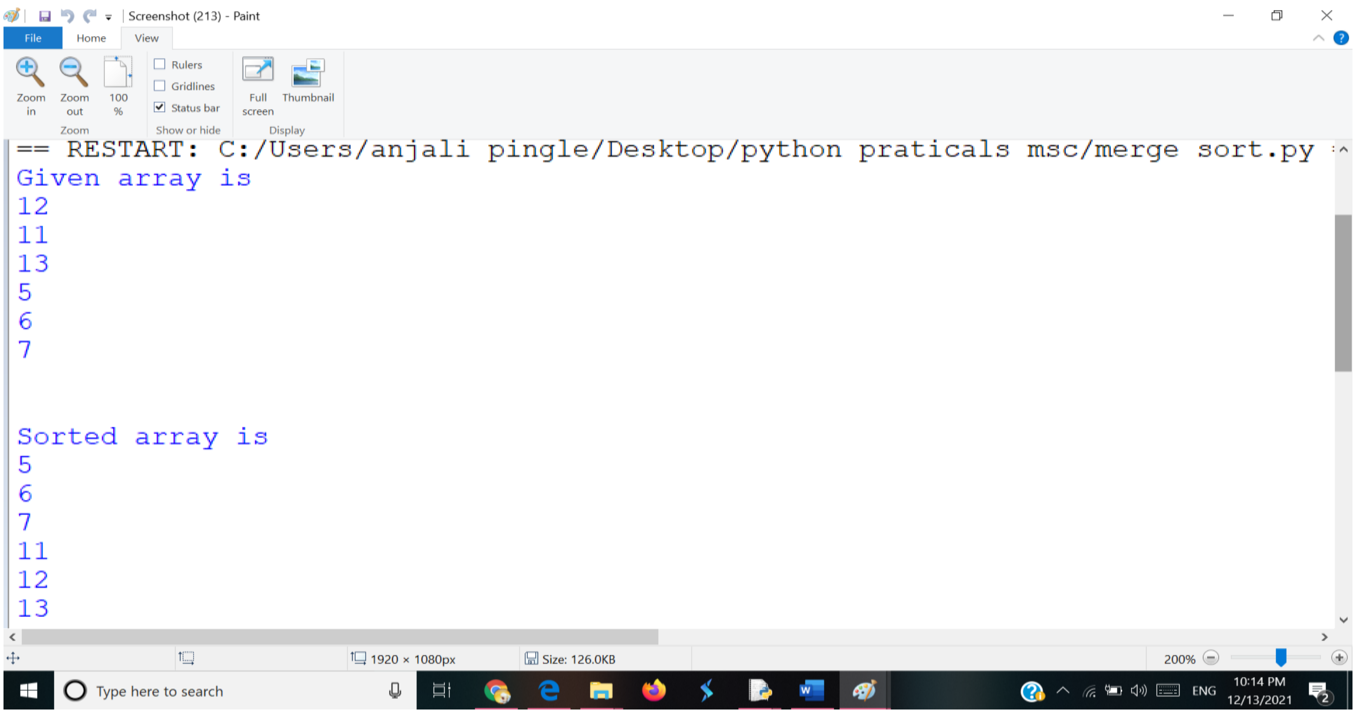
Aim: Write a Python program to implement merge sort.

Input:

```
# Python program for implementation of MergeSort

# Merges two subarrays of arr[].
# First subarray is arr[l..m]
# Second subarray is arr[m+1..r]


def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m

    # create temp arrays
    L = [0] * (n1)
    R = [0] * (n2)

    # Copy data to temp arrays L[] and R[]
    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]
    i = 0     # Initial index of first subarray
    j = 0     # Initial index of second subarray
    k = l     # Initial index of merged subarray

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    # Copy the remaining elements of L[], if there
    # are any
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    # Copy the remaining elements of R[], if there
    # are any
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

# l is for left index and r is right index of the
# sub-array of arr to be sorted


def mergeSort(arr, l, r):
    if l < r:

        # Same as (l+r)//2, but avoids overflow for
        # large l and h
        m = l+(r-l)//2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)



# Driver code to test above
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
```

```
print("Given array is")
for i in range(n):
    print("%d" % arr[i]),

mergeSort(arr, 0, n-1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i]),

# This code is contributed by Mohit Kumra
```

Output:



```
== RESTART: C:/Users/anjali pingle/Desktop/python praticals msc/merge sort.py
Given array is
12
11
13
5
6
7


Sorted array is
5
6
7
11
12
13
```