LAB#19

Q#1: Write a program to sort an unsorted array by using a merge sort algorithm.

Solution:

```
1
2   from array import *
3   def merge(left, right):
4       merged = []
5       left_index = right_index = 0
6
7
8       while left_index < len(left) and right_index < len(right):
9           if left[left_index] < right[right_index]:
10              merged.append(left[left_index])
11              left_index += 1
12          else:
13              merged.append(right[right_index])
14              right_index += 1
15
16
17      merged.extend(left[left_index:])
18      merged.extend(right[right_index:])
19
20      return merged
21
22  def merge_sort(arr):
23      if len(arr) <= 1:
24          return arr
25
26
```

```
27      mid = len(arr) // 2
28      left_half = arr[:mid]
29      right_half = arr[mid:]
30
31
32      left_half = merge_sort(left_half)
33      right_half = merge_sort(right_half)
34
35
36      return merge(left_half, right_half)
37
38
39
40  a1 = array('i', [23, 56, 12, 14, 5])
41
42
43  sorted_a1 = array('i', merge_sort(a1))
44
45
46  for x in sorted_a1:
47      print(x,end=' ')
48
```

Output:

```
5 12 14 23 56
```

Explanation:

a1=[23,56,12,14,5]

sorted_a1=array(merge_sort(a1))

1:

Def merge_sort([23,56,12,14,5])

If len(arr)<=1

return arr

X

mid=len(arr)//2

mid=5//2

mid=2

lefthalf=arr[:2]

righthalf=arr[2:]

1-left lefthalf=merge_sort[23,56]

1-right righthalf=merge_sort[12,14,5]

I return merge[left,right]

1-left:

def merge_sort[23,56]:

if len(arr)<=1

X

mid=len(arr)//2

mid=2//2

mid=1

lefthalf=arr[:1]

righthalf=arr[1:]

1-left-left left**half=mergesort[23]**

1-left-right **righthalf=mergesort[56]**

II return merge[lefthalf,righthalf]

1-left-left:

Def mergesort(23):

If len(23)<=1

Return 23

1-left-right:

```
Def mergesort(56):
    If len(56)<=1
     Return 56
```

II:

```
Def merge(23,56):
     Merged=[]
     leftindex=rightindex=0

    while leftindex<len(left) and rightindex<len(right):
              if left[0]<right[0]
                23<56
                Merged.append(23)
                Leftindex=1

While   1<1
    X
merged.extend(left(1))    X
merged.extend(right[0])
```

merged[23,56]

return [23,56]

The values will be returned to the 1-left in the function (1):

```
1:
Def merge_sort([23,56,12,14,5])
            If len(arr)<=1
                return arr
                    X
            mid=len(arr)//2
            mid=5//2
            mid=2
            lefthalf=arr[:2]
            righthalf=arr[2:]
1-left      lefthalf=merge_sort[23,56]
1-right     righthalf=merge_sort[12,14,5]
I           return merge[left,right]
```

Here we have:

1-left lefthalf=[23,56]

1-right righthalf=mergesort[12,14,5]

I: return merge[lefthalf,righthalf]

1-right :

Def merge([12,14,5])

If len(arr)<=1:

X

mid=len(arr)//2

mid=3//2

mid=1

lefthalf=arr[:1]

righthalf=arr[1:]

1-right-left: lefthalf=mergesort[12]

1-right-right: righthalf=mergesort[14,5]

III: return merge[lefthalf,righthalf]

1-right-left:

def mergesort[12]:

If len(arr)<=1:

Return(12)

1-right-right:

Def mergesort[14,5]:

If len(arr)<=1:

X

Mid=len(14,5)//2

Mid=2//2

Mid=1

Lefthalf=14

Righthalf=5

1-right-right-left: lefthalf=mergesort(14)

1-right-right-right: righthalf=mergesort(5)

IV: return merge[lefthalf,righthalf]

1-right-right-left:

def mergesort(14):

if len(arr)<=1

return 14

1-right-right-right:

def mergesort(5):

if len(arr)<=1

return 5

By putting the values in the 1-right-right:

1-right-right:

Def mergesort[14,5]:

If len(arr)<=1:

X

Mid=len(14,5)//2

Mid=2//2

Mid=1

Lefthalf=14

Righthalf=5

1-right-right-left: lefthalf=14

1-right-right-right: righthalf=5

IV: return merge[lefthalf,righthalf]

IV:

def merge(14,5):

merged=[]

leftindex=rightindex=0

while leftindex<len(left) and rightindex<len(right):

if left[0]<right[0]:

if 14<5

X

Else:

merged[5]

Rightindex=1

While 0<1 and 1<1:

X

merged.extend(left[0])

merged[5,14]

merged.extend(right[1]) X

return merged

return [5,14]

[5,14] → 1-right-right in 1-right

1-right :

```
Def  merge([12,14,5])
                If len(arr)<=1:
                        X
                mid=len(arr)//2
                mid=3//2
                 mid=1
                lefthalf=arr[:1]
                righthalf=arr[1:]
```

1-right-left: lefthalf=[12]

1-right-right: righthalf=[5,14]

III: return merge[lefthalf,righthalf]

Now:

III:

```
def merge([12],[5,14]):
    merged=[]
    leftindex=rightindex=0
```

while leftindex<len(12) and rightindex<len([5,14]):

if left[0]<right[0]:

12<5

X

Else:

Merged[5]

Rightindex=1

While 0<1 and 1<2:

If left[0]<right[1]:

12<14

merged[5,12]

leftindex=1

while 1<1:

X

Merged extend(left[1:]) X

Merged.extend(right[1:])

merged extend(14)

merged [5,12,14]

[5,12,14] →1-right →1

1:

Def merge_sort([23,56,12,14,5])

If len(arr)<=1

return arr

X

mid=len(arr)//2

mid=5//2

mid=2

lefthalf=arr[:2]

righthalf=arr[2:]

1-left lefthalf=[23,56]

1-right righthalf=[5,12,14]

I return merge[left,right]

I:

```
def merge([23,56],[5,12,14]):
   merged=[]
   leftindex=rightindex=0
   while   0<2 and 0<3:
         if  left[0]<right[0]:
               23<12
         Else:
               Merged[5]
               Rightindex=1

while   0<2 and 1<3:
         if  left[0]<right[1]:
               23<12
         Else:
               Merged[5,12]
               Rightindex=2

While  0<2 and 2<3:
      If left[0]<right[2]:
         23<14
```

X

Else:

Merged[5,12,14]

Rightindex=3

While 0<2 and 3<3:

X

Merged.extend(left[0:])

Merged.extend[23,56]

Merged[5,12,14,23,56]

merged.extend(right,3) X

return [5,12,14,23,56]

sorted_a1=[5,12,14,23,56]

for x in [5,12,14,23,56]:

print(x,end= ‘ ’)

Class Assignment

Q: Write a program to apply the merge sort algorithm on an unsorted dictionary to sort its values.

Consider the following dictionary:

```
d = {'ali':14 ,'muhammad':1,'abbas':12,'saqlain':10, 'imran':11}
```