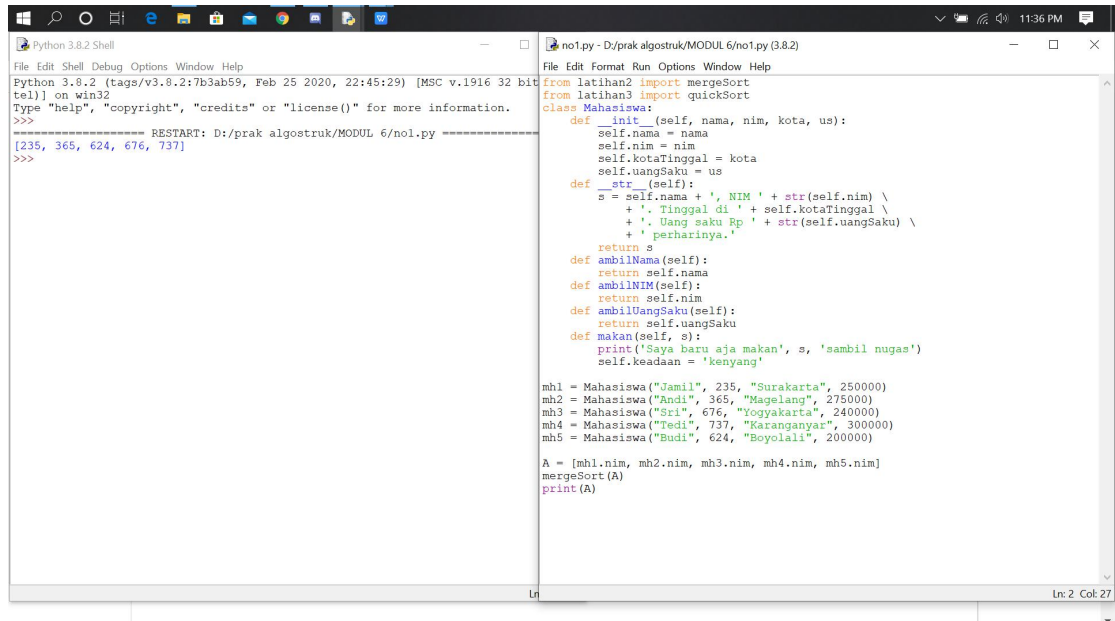


Nama : Auzan Danar K
NIM : L200180005
Kelas : A

TUGAS

1.



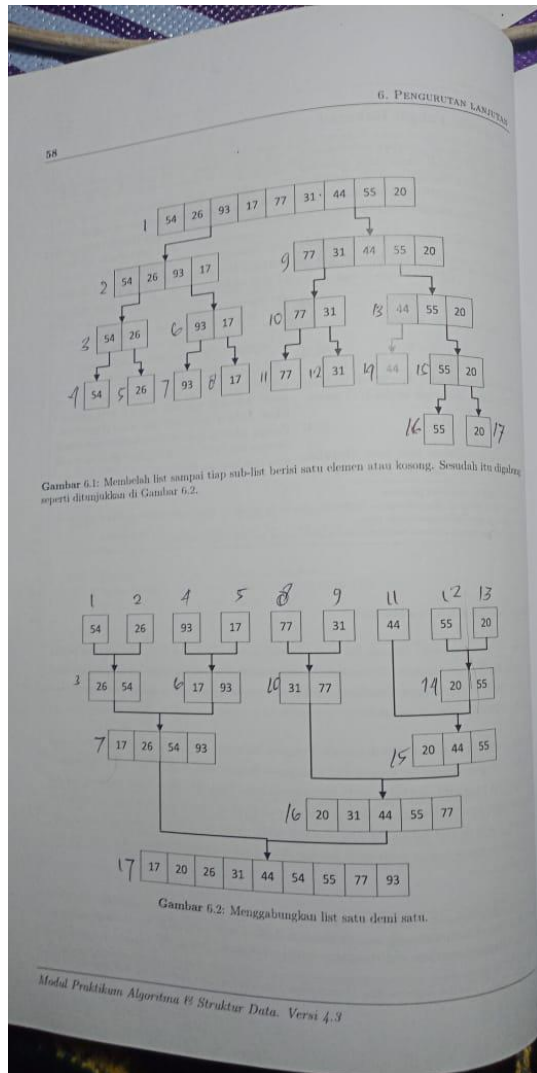
```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no1.py =====
[235, 365, 624, 676, 737]
>>>
```

```
no1.py - D:/prak algostruk/MODUL 6/no1.py (3.8.2)
File Edit Format Run Options Window Help
from latihan2 import mergeSort
from latihan3 import quickSort
class Mahasiswa:
    def __init__(self, nama, nim, kota, us):
        self.nama = nama
        self.nim = nim
        self.kotaTinggal = kota
        self.uangSaku = us
    def __str__(self):
        s = self.nama + ', NIM ' + str(self.nim) \
            + '. Tinggal di ' + self.kotaTinggal \
            + '. Uang saku Rp ' + str(self.uangSaku) \
            + ' perharinya.'
        return s
    def ambilNama(self):
        return self.nama
    def ambilNIM(self):
        return self.nim
    def ambilUangSaku(self):
        return self.uangSaku
    def makan(self, s):
        print('Saya baru aja makan', s, 'sambil nugas')
        self.keadaan = 'kenyang'

mh1 = Mahasiswa("Jamil", 235, "Surakarta", 250000)
mh2 = Mahasiswa("Andi", 365, "Magelang", 275000)
mh3 = Mahasiswa("Sri", 676, "Yogyakarta", 240000)
mh4 = Mahasiswa("Tedi", 737, "Karanganyar", 300000)
mh5 = Mahasiswa("Budi", 624, "Boyolali", 200000)

A = [mh1.nim, mh2.nim, mh3.nim, mh4.nim, mh5.nim]
mergeSort(A)
print(A)
```

2.



3.

The screenshot shows a Python 3.8.2 Shell window on the left and a code editor on the right. The shell displays the execution times for different sorting algorithms on a list of 6000 random integers.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more informati
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no3.py =====
bubble : 5.93134 detik
selection : 2.28618 detik
insertion : 2.71732 detik
merge : 0.0349083 detik
quick : 0.0229733 detik
>>>
```

The code editor shows the implementation of these algorithms:

```
no3.py - D:/prak algostruk/MODUL 6/no3.py (3.8.2)
File Edit Format Run Options Window Help
from time import time as detik
from random import shuffle as kocok
from latihan2 import mergeSort
from latihan3 import *

k = [i for i in range(1, 6000)]
kocok(k)

def swap(A, p, q):
    temp = A[p]
    A[p] = A[q]
    A[q] = temp

def cariposisiterkecil(A, darisini, sampaisini):
    posisiterkecil = darisini
    for i in range(darisini + 1, sampaisini):
        if A[i] < A[posisiterkecil]:
            posisiterkecil = i
    return posisiterkecil

def bubbleSort(A):
    n = len(A)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if A[j] > A[j + 1]:
                swap(A, j, j + 1)

def selectionSort(A):
    n = len(A)
    for i in range(n - 1):
        indexkecil = cariposisiterkecil(A, i, n)
        if indexkecil != i:
            swap(A, i, indexkecil)

def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:
            A[pos] = A[pos - 1]
            pos = pos - 1
        A[pos] = nilai
```

Page Num: 1 Page: 1/1 Section: 1/1 SetValue: 1in Row: 1 Column: 2 Words: 0 Spell Check Lnc: 2 Col

4. a.

L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72]

80	7	24	16	43	91	35	2	19	72
----	---	----	----	----	----	----	---	----	----

Proses 1

7	80	26	24	43	91	2	35	19	72
---	----	----	----	----	----	---	----	----	----

Proses 2

7	16	24	80	2	35	43	91	19	72
---	----	----	----	---	----	----	----	----	----

Proses 3

2	7	16	24	35	43	80	91	19	72
---	---	----	----	----	----	----	----	----	----

Proses 4

2	7	16	19	24	35	43	72	80	91
---	---	----	----	----	----	----	----	----	----

b.

L = L = [80, 7, 24, 16, 43, 91, 35, 2, 19, 72]

80	7	24	16	43	91	35	2	19	72
----	---	----	----	----	----	----	---	----	----

pivot

80	7	24	16	43	91	35	2	19	72
----	---	----	----	----	----	----	---	----	----

low

high

pivot

72	7	24	16	43	91	35	2	19	80
----	---	----	----	----	----	----	---	----	----

low

high

pivot

72	7	24	16	43	91	35	2	19	80
----	---	----	----	----	----	----	---	----	----

low

high

pivot

72	7	24	16	43	80	35	2	19	91
----	---	----	----	----	----	----	---	----	----

low

high

pivot

72	7	24	16	43	19	35	2	80	91
----	---	----	----	----	----	----	---	----	----

low

high

5.

The screenshot shows a Python 3.8.2 Shell on the left and a Python 3.8.2 IDE on the right. The shell displays the execution of a merge sort function on a list of numbers. The IDE shows the implementation of the merge sort algorithm, including recursive functions for splitting and merging the list.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit
tel) on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no5.py =====
>>> a = merge_sort([32,13,53,2,54,21,7,12,43,9])
>>> print(a)
[2, 7, 9, 12, 13, 21, 32, 43, 53, 54]
>>>
```

```
no5.py - D:/prak algostruk/MODUL 6/no5.py (3.8.2)
File Edit Format Run Options Window Help
def merge_sort(indices, the_list):
    start = indices[0]
    end = indices[1]
    half_way = (end - start) // 2 + start
    if start < half_way:
        merge_sort((start, half_way), the_list)
    if half_way + 1 <= end and end - start != 1:
        merge_sort((half_way + 1, end), the_list)
    sort_sub_list(the_list, indices[0], indices[1])
    return the_list

def sort_sub_list(the_list, start, end):
    orig_start = start
    initial_start_second_list = (end - start) // 2 + start + 1
    list2_first_index = initial_start_second_list
    new_list = []
    while start < initial_start_second_list and list2_first_index <= end:
        first1 = the_list[start]
        first2 = the_list[list2_first_index]
        if first1 > first2:
            new_list.append(first2)
            list2_first_index += 1
        else:
            new_list.append(first1)
            start += 1
    while start < initial_start_second_list:
        new_list.append(the_list[start])
        start += 1
    while list2_first_index <= end:
        new_list.append(the_list[list2_first_index])
        list2_first_index += 1
    for i in new_list:
        the_list[orig_start] = i
        orig_start += 1
    return the_list

def merge_sort(the_list):
    return merge_sort((0, len(the_list) - 1), the_list)
```

6.

The screenshot shows a Python 3.8.2 Shell on the left and a Python 3.8.2 IDE on the right. The shell displays the execution of a quick sort function on a list of numbers. The IDE shows the implementation of the quick sort algorithm, including recursive functions for partitioning and sorting the sub-arrays.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit
tel) on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no6.py =====
>>> a = [65,72,83,12,34,65,34,7,43]
>>> quickSort(a)
>>> print(a)
[7, 12, 34, 34, 43, 65, 65, 72, 83]
>>>
```

```
no6.py - D:/prak algostruk/MODUL 6/no6.py (3.8.2)
File Edit Format Run Options Window Help
def quickSort(A):
    quicksorthelp(A, 0, len(A))

def quicksorthelp(A, low, high):
    result = 0
    if low < high:
        pivot_location, result = Partition(A, low, high)
        result += quicksorthelp(A, low, pivot_location)
        result += quicksorthelp(A, pivot_location + 1, high)
    return result

def Partition(A, low, high):
    result = 0
    pivot, pidx = median_of_three(A, low, high)
    A[low], A[pidx] = A[pidx], A[low]
    i = low + 1
    for j in range(low + 1, high, 1):
        result += 1
        if A[j] < pivot:
            A[i], A[j] = A[j], A[i]
            i += 1
    A[low], A[i - 1] = A[i - 1], A[low]
    return i - 1, result

def median_of_three(A, low, high):
    mid = (low + high - 1) // 2
    a = A[low]
    b = A[mid]
    c = A[high - 1]
    if a <= b <= c:
        return b, mid
    if c <= b <= a:
        return b, mid
    if a <= c <= b:
        return c, high - 1
    if b <= c <= a:
        return c, high - 1
    return a, low
```

7.

```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MS
tel]] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no7.py =====
bubble : 6.42455 detik
selection : 2.32183 detik
insertion : 2.94671 detik
merge : 0.0395269 detik
quick : 0.0295527 detik
merge sort baru : 0.0397067 detik
quick sort baru : 0.0319219 detik
merge sort awal : 0.0328009 detik
quick sort awal : 0.0198121 detik
>>>

no7.py - D:/prak algostruk/MODUL 6/no7.py (3.8.2)
File Edit Format Run Options Window Help
from time import time as detak
from random import shuffle as kocok
import no5 # mergeSort baru
import no6 # quickSort baru
import no3 # mergeSort dan quickSort awal
k = [i for i in range(1, 6000)]
kocok(k)

merA = k[:]
merB = k[:]
quiA = k[:]
quiB = k[:]

# merge Sort baru
aw = detak(); no5.merge_sort(merB); ak = detak(); print('merge sort baru : %g detik' % (ak-aw))
# Quick Sort baru
aw = detak(); no6.quickSort(quiB); ak = detak(); print('quick sort baru : %g detik' % (ak-aw))

# Merge Sort dan Quick Sort awal
aw = detak(); no3.mergeSort(merA); ak = detak(); print('merge sort awal : %g detik' % (ak-aw))
aw = detak(); no3.quickSort(quiA); ak = detak(); print('quick sort awal : %g detik' % (ak-aw))

```

8.

```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit
tel]] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/prak algostruk/MODUL 6/no8.py =====
List 1 :
1
4
22
23
32
List 2 :
3
10
26
Merged List :
1
3
4
10
22
23
26
32
>>>

no8.py - D:/prak algostruk/MODUL 6/no8.py (3.8.2)
File Edit Format Run Options Window Help
curr = curr.next

def mergeSorted(self, list1, list2):
    if list1 is None:
        return list2
    if list2 is None:
        return list1

    if list1.data < list2.data:
        temp = list1
        temp.next = self.mergeSorted(list1.next, list2)
    else:
        temp = list2
        temp.next = self.mergeSorted(list1, list2.next)
    return temp

list1 = LinkedList()
list1.appendSorted(23)
list1.appendSorted(22)
list1.appendSorted(1)
list1.appendSorted(32)
list1.appendSorted(4)

print("List 1 :"),
list1.printList()

list2 = LinkedList()
list2.appendSorted(26)
list2.appendSorted(10)
list2.appendSorted(3)

print("List 2 :"),
list2.printList()

list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)

print("Merged List :"),
list3.printList()

```