

Nama : Noor Aniq W.

NIM : L200180191

Kelas : G

## Praktikum Algoritma dan Struktur Data

1.

```
Modul 6_1.py - F:\Garapan\Modul 6_1.py (3.7.3)
File Edit Format Run Options Window Help

from LatOOP4 import Mahasiswa
class MhsTIF(Mahasiswa):
    def katakanPy(self):
        print("Python is cool")

m1=MhsTIF("Budi",131,"Sieman",300000)
m2=MhsTIF("Ahmad",213,"Jambi",2000000)
m3=MhsTIF("Erik",125,"Jakarta",1750000)
m4=MhsTIF("Zainuddin",141,"Depok",1000000)
m5=MhsTIF("Andre",172,"Tangerang",800000)
m6=MhsTIF("Putri",152,"Klaten",300000)
m7=MhsTIF("Fatimah",181,"Depok",900000)
m8=MhsTIF("Nurul",113,"Garangan",200000)
m9=MhsTIF("Jason",138,"Jakarta",1500000)
m10=MhsTIF("Bagas",159,"Semarang",500000)

urut = [m1.NIM, m2.NIM, m3.NIM, m4.NIM, m5.NIM,
        m6.NIM, m7.NIM, m8.NIM, m9.NIM, m10.NIM]

def mergeSort(nlist):
    print("Memelah ", nlist)
    if len(nlist)>1:
        mid = len(nlist)//2
        lefthalf = nlist[:mid]
        righthalf = nlist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=j=k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                nlist[k]=lefthalf[i]
                i=i+1
            else:
                nlist[k]=righthalf[j]
                j=j+1
            k=k+1
        while i < len(lefthalf):
            nlist[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j < len(righthalf):
            nlist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Menggabungkan ", nlist)
    nlist = urut
    print("Hasil MergeSort")
    mergeSort(nlist)
    print(nlist)

def quickSort(data_list):
    quickSortHlp(data_list,0,len(data_list)-1)

def quickSortHlp(data_list,first,last):
    if first < last:
        splitpoint = partition(data_list,first,last)

        quickSortHlp(data_list,first,splitpoint-1)
        quickSortHlp(data_list,splitpoint+1,last)

def partition(data_list,first,last):
    pivot = data_list[first]
    left = first+1
    right = last
    while True:
        while left <= right:
            if data_list[left] <= pivot:
                left = left + 1
            else:
                data_list[left] = data_list[right]
                left = left + 1
        while right >= left:
            if data_list[right] >= pivot:
                right = right - 1
            else:
                data_list[right] = data_list[left]
                right = right - 1
        data_list[left] = data_list[first]
        data_list[right] = pivot
    return right
```

```
Modul 6_1.py - F:\Garapan\Modul 6_1.py (3.7.3)
File Edit Format Run Options Window Help

mergeSort(lefthalf)
mergeSort(righthalf)
i=j=k=0
while i < len(lefthalf) and j < len(righthalf):
    if lefthalf[i] < righthalf[j]:
        nlist[k]=lefthalf[i]
        i=i+1
    else:
        nlist[k]=righthalf[j]
        j=j+1
    k=k+1

while i < len(lefthalf):
    nlist[k]=lefthalf[i]
    i=i+1
    k=k+1

while j < len(righthalf):
    nlist[k]=righthalf[j]
    j=j+1
    k=k+1

print("Menggabungkan ",nlist)
nlist = urut
print("Hasil MergeSort")
mergeSort(nlist)
print(nlist)

def quickSort(data_list):
    quickSortHlp(data_list,0,len(data_list)-1)

def quickSortHlp(data_list,first,last):
    if first < last:
        splitpoint = partition(data_list,first,last)

        quickSortHlp(data_list,first,splitpoint-1)
        quickSortHlp(data_list,splitpoint+1,last)

def partition(data_list,first,last):
    pivot = data_list[first]
    left = first+1
    right = last
    while True:
        while left <= right:
            if data_list[left] <= pivot:
                left = left + 1
            else:
                data_list[left] = data_list[right]
                left = left + 1
        while right >= left:
            if data_list[right] >= pivot:
                right = right - 1
            else:
                data_list[right] = data_list[left]
                right = right - 1
        data_list[left] = data_list[first]
        data_list[right] = pivot
    return right
```

```
Modul 6_1.py - F:/Garapan/Modul 6_1.py (3.7.3)
File Edit Format Run Options Window Help

def partition(data_list, first, last):
    pivotvalue = data_list[first]

    leftmark = first+1
    rightmark = last

    done = False
    while not done:
        while leftmark <= rightmark and data_list[leftmark] <= pivotvalue:
            leftmark = leftmark + 1

        while data_list[rightmark] >= pivotvalue and rightmark >= leftmark:
            rightmark = rightmark - 1

        if rightmark < leftmark:
            done = True
        else:
            temp = data_list[leftmark]
            data_list[leftmark] = data_list[rightmark]
            data_list[rightmark] = temp

        temp = data_list[first]
        data_list[first] = data_list[rightmark]
        data_list[rightmark] = temp

    return rightmark

data_list = urut
quickSort(data_list)
print("\n"+"Hasil QuickSort")
print(data_list)
```

Ln: 17 Col: 7

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_1.py =====
Hasil MergeSort
Membelah [131, 213, 125, 141, 172, 192, 181, 113, 138, 159]
Membelah [131, 213, 125, 141, 172]
Membelah [131, 213]
Membelah [131]
Menggabungkan [131]
Membelah [213]
Menggabungkan [213]
Menggabungkan [131, 213]
Membelah [125, 141, 172]
Membelah [125]
Menggabungkan [125]
Membelah [141, 172]
Membelah [141]
Menggabungkan [141]
Membelah [172]
Menggabungkan [172]
Menggabungkan [141, 172]
Menggabungkan [125, 141, 172]
Menggabungkan [125, 131, 141, 172, 213]
Membelah [192, 181, 113, 138, 159]
Membelah [192, 181]
Membelah [192]
Menggabungkan [192]
Membelah [181]
Menggabungkan [181]
Menggabungkan [181, 192]
Membelah [113, 138, 159]
Membelah [113]
Menggabungkan [113]
Membelah [138, 159]
Membelah [138]
Menggabungkan [138]
Membelah [159]
Menggabungkan [159]
Menggabungkan [138, 159]
Menggabungkan [113, 138, 159]
Menggabungkan [113, 138, 159, 181, 192]
```

Ln: 48 Col: 4

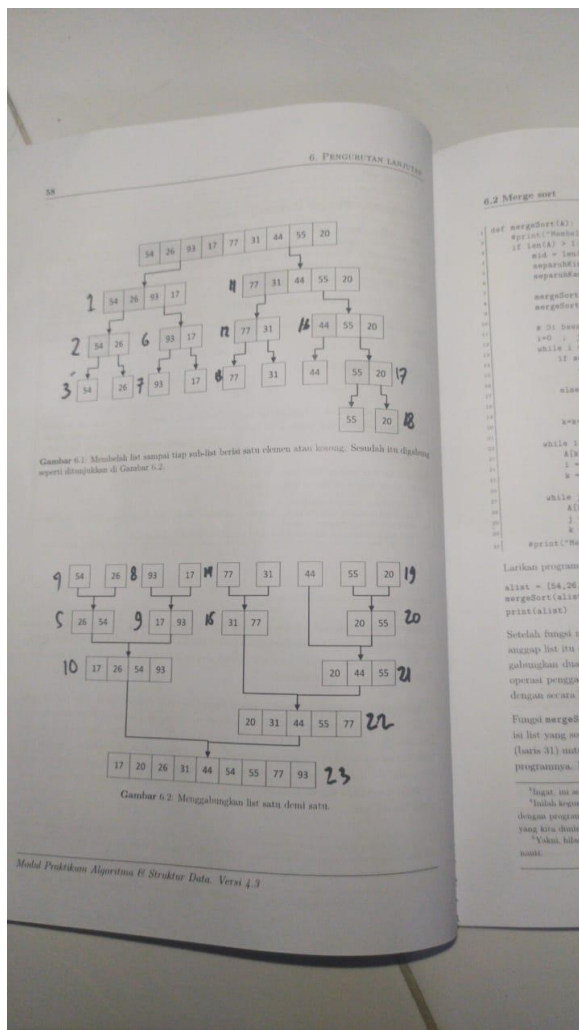
```

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Membelah [131, 213]
Membelah [131]
Menggabungkan [131]
Membelah [213]
Menggabungkan [213]
Menggabungkan [131, 213]
Membelah [125, 141, 172]
Membelah [125]
Menggabungkan [125]
Membelah [141, 172]
Membelah [141]
Menggabungkan [141]
Membelah [172]
Menggabungkan [172]
Menggabungkan [141, 172]
Menggabungkan [125, 141, 172]
Menggabungkan [125, 131, 141, 172, 213]
Membelah [192, 181, 113, 138, 159]
Membelah [192, 181]
Membelah [192]
Menggabungkan [192]
Membelah [181]
Menggabungkan [181]
Menggabungkan [181, 192]
Membelah [113, 138, 159]
Membelah [113]
Menggabungkan [113]
Membelah [138, 159]
Membelah [138]
Menggabungkan [138]
Membelah [159]
Menggabungkan [159]
Menggabungkan [138, 159]
Menggabungkan [113, 138, 159]
Menggabungkan [113, 138, 159, 181, 192]
Menggabungkan [113, 125, 131, 138, 141, 159, 172, 181, 192, 213]
[113, 125, 131, 138, 141, 159, 172, 181, 192, 213]

Hasil QuickSort
[113, 125, 131, 138, 141, 159, 172, 181, 192, 213]
>>>

```

2.



3.

```
Modul 6.3.py - F:/Garapan/Modul 6.3.py (3.7.3)
File Edit Format Run Options Window Help

from time import time as detik
from random import shuffle as kocok
import time
k = [1 for i in range(1,6001)]
kocok(k)

def bubbleSort(X) :
    n = len(X)
    for i in range(n):
        for j in range(0, n-i-1):
            if X[j] > X[j+1] :
                X[j], X[j+1] = X[j+1], X[j]]

def selectionSort(X) :
    for i in range(len(X)):
        min_idk = i
        for j in range(i+1, len(X)):
            if X[min_idk] > X[j]:
                min_idk = j
        X[i], X[min_idk] = X[min_idk], X[i]

def insertSort(X) :
    n = len(X)
    for i in range(1, n) :
        nilai = X[i]
        abc = i-1
        while abc >= 0 and nilai < X[abc-1] :
            X[abc] = X[abc+1]
            abc -=1
        X[abc+1] = nilai

def mergeSort(X):
    if len(X) >1:
        mid = len(X)//2
        L = X[:mid]
        R = X[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                X[k] = L[i]
                i+=1
            else:
                X[k] = R[j]
                j+=1
                k+=1
        while i < len(L):
            X[k] = L[i]
            i+=1
            k+=1
        while j < len(R):
            X[k] = R[j]
            j+=1
            k+=1

def partition(X,low,high):
    i = ( low-1 )
    pivot = X[high]
    for j in range( low , high):
        if X[j] <= pivot:
            i = i+1
            X[i],X[j] = X[j],X[i]
    X[i+1],X[high] = X[high],X[i+1]
    return ( i+1 )

def quickSort(X,low,high):
    if low < high:
        pi = partition(X,low,high)
        quickSort(X, low, pi-1)
        quickSort(X, pi+1, high)

u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mer = k[:]
u_qck = k[:]

aw = detik () ; bubbleSort (u_bub) ; ak = detik() ; print('bubble : % g detik' % (ak - aw)) ;
aw = detik () ; selectionSort (u_sel) ; ak = detik() ; print('selection : % g detik' % (ak - aw)) ;
aw = detik () ; insertSort (u_ins) ; ak = detik() ; print('insert : % g detik' % (ak - aw)) ;
aw = detik () ; mergeSort (u_mer) ; ak = detik() ; print('merge : % g detik' % (ak - aw)) ;
aw = detik () ; quickSort (u_qck, 0, len(u_qck)-1) ; ak = detik() ; print('quick : % g detik' % (ak - aw)) ;
```

```
Modul 6.3.py - F:/Garapan/Modul 6.3.py (3.7.3)
File Edit Format Run Options Window Help

        i+=1
        k+=1
    while i < len(L):
        X[k] = L[i]
        i+=1
        k+=1
    while j < len(R):
        X[k] = R[j]
        j+=1
        k+=1
def partition(X,low,high):
    i = ( low-1 )
    pivot = X[high]
    for j in range( low , high):
        if X[j] <= pivot:
            i = i+1
            X[i],X[j] = X[j],X[i]
    X[i+1],X[high] = X[high],X[i+1]
    return ( i+1 )

def quickSort(X,low,high):
    if low < high:
        pi = partition(X,low,high)
        quickSort(X, low, pi-1)
        quickSort(X, pi+1, high)

u_bub = k[:]
u_sel = k[:]
u_ins = k[:]
u_mer = k[:]
u_qck = k[:]

aw = detik () ; bubbleSort (u_bub) ; ak = detik() ; print('bubble : % g detik' % (ak - aw)) ;
aw = detik () ; selectionSort (u_sel) ; ak = detik() ; print('selection : % g detik' % (ak - aw)) ;
aw = detik () ; insertSort (u_ins) ; ak = detik() ; print('insert : % g detik' % (ak - aw)) ;
aw = detik () ; mergeSort (u_mer) ; ak = detik() ; print('merge : % g detik' % (ak - aw)) ;
aw = detik () ; quickSort (u_qck, 0, len(u_qck)-1) ; ak = detik() ; print('quick : % g detik' % (ak - aw)) ;
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_3.py =====
bubble : 3.46763 detik
selection : 1.65586 detik
insert : 0 detik
merge : 0.0155864 detik
quick : 0.0156145 detik
>>> |
```

4.

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

a. Merge sort

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

Langkah 1

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

Langkah 2

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

Langkah 3

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

Langkah 4

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

b. Quick sort

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

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

Pivot

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

Low

High

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

5.

```
Modul 6.5.py - F:/Garapan/Modul 6.5.py (3.7.3)
File Edit Format Run Options Window Help

import random
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)

print(merge_sort([13,45,12,3,10,2]))
```

```
Modul 6.5.py - F:/Garapan/Modul 6.5.py (3.7.3)
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)

print(merge_sort([13,45,12,3,10,2]))
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_5.py =====
[2, 3, 10, 12, 13, 45]
>>> |
```



6.

The screenshot shows a Python IDE window titled 'Modul 6.5.py - F:/Garapan/Modul 6.5.py (3.7.3)'. The code defines three functions: `quickSort`, `quicksorthelp`, and `Partition`. `quickSort` is a recursive function that sorts a list `L` in ascending order. `quicksorthelp` is a helper function that performs the partitioning step. `Partition` is a function that selects a pivot (the median of the first, middle, and last elements) and partitions the list around it. The IDE also shows a Python 3.7.3 Shell window with the following code and output:

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_6.py =====
>>> l = list([14,4,2,104,23,50])
>>> quickSort(l, False)
>>> print(l)
[104, 50, 23, 14, 4, 2]
>>>
```

7.

The screenshot shows a Python IDE window titled 'Modul 6.7.py - F:/Garapan/Modul 6.7.py (3.7.3)'. The code defines three functions: `mergeSort`, `partition`, and `quickSort`. `mergeSort` is a recursive function that sorts a list `arr` using the merge sort algorithm. `partition` is a function that selects a pivot (the first element) and partitions the list around it. `quickSort` is a function that sorts a list `arr` in ascending order using the partition function. The IDE also shows a Python 3.7.3 Shell window with the following code and output:

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_6.py =====
>>> l = list([14,4,2,104,23,50])
>>> quickSort(l, False)
>>> print(l)
[104, 50, 23, 14, 4, 2]
>>>
```

```
Modul 6,7.py - F:/Garapan/Modul 6,7.py (3.7.3)
File Edit Format Run Options Window Help

def quickSort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

import random
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])

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
    return new_list

Ln: 1 Col: 0
```

```
Modul 6,7.py - F:/Garapan/Modul 6,7.py (3.7.3)
File Edit Format Run Options Window Help

    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

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

def quickSortMOD(L, ascending = True):
    quicksorthelp(L, 0, len(L), ascending)

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

def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low+1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i-1] = L[i-1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    return (low + high) // 2

Ln: 1 Col: 0
```

```
Modul 6,7.py - F:/Garapan/Modul 6,7.py (3.7.3)
File Edit Format Run Options Window Help

    return result

def Partition(L, low, high, ascending = True):
    result = 0
    pivot, pidx = median_of_three(L, low, high)
    L[low], L[pidx] = L[pidx], L[low]
    i = low + 1
    for j in range(low+1, high, 1):
        result += 1
        if (ascending and L[j] < pivot) or (not ascending and L[j] > pivot):
            L[i], L[j] = L[j], L[i]
            i += 1
    L[low], L[i-1] = L[i-1], L[low]
    return i - 1, result

def median_of_three(L, low, high):
    mid = (low+high-1)//2
    a = L[low]
    b = L[mid]
    c = L[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

mer = k[:]
qui = k[:]
mer2 = k[:]
qui2 = k[:]

aw=detak();mergeSort(mer);ak=detak();print('merge : %g detik' %(ak-aw));
aw=detak();quickSort(qui,0,len(qui)-1);ak=detak();print('quick : %g detik' %(ak-aw));
aw=detak();merge_sort(mer2);print('merge mod : %g detik' %(ak-aw));
aw=detak();quickSortMOD(qui2, False);print('quick mod : %g detik' %(ak-aw));

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_7.py =====
merge : 0.0309527 detik
quick : 0.0156238 detik
merge mod : -0.0156226 detik
quick mod : -0.0468276 detik
>>>
```

8.

```
Modul 6,8.py - F:/Garapan/Modul 6,8.py (3.7.3)
File Edit Format Run Options Window Help

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def appendList(self, data):
        node = Node(data)
        if self.head == None:
            self.head = node
        else:
            curr = self.head
            while curr.next != None:
                curr = curr.next
            curr.next = node

    def appendSorted(self, data):
        node = Node(data)
        curr = self.head
        prev = None

        while curr is not None and curr.data < data:
            prev = curr
            curr = curr.next

        if prev == None:
            self.head = node
        else:
            prev.next = node
        node.next = curr

    def printList(self):
        curr = self.head
        while curr != None:
            print ("%d"%curr.data),
            curr = curr.next

    def mergeSorted(self, list1, list2):
        # to be implemented
```

```
Modul 6_8.py - F:/Garapan/Modul 6_8.py (3.7.3)
File Edit Format Run Options Window Help
#next curr = None:
    print ("%d"%curr.data),
    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(13)
list1.appendSorted(12)
list1.appendSorted(3)
list1.appendSorted(16)
list1.appendSorted(7)

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

list2 = LinkedList()
list2.appendSorted(9)
list2.appendSorted(10)
list2.appendSorted(1)

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

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

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

Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4e6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/Garapan/Modul 6_8.py =====
List 1 :
3
7
12
13
16
List 2 :
1
9
10
Merged List :
1
3
7
9
10
12
13
16
>>> |
Ln: 24 Col: 4
```