NAMA : AHMAD ROZIN

NIM : L200170135

KELAS : D

MODUL 6

Nomor 1

nomor1.py - C:\Users\evolved\Desktop\nomor1.py (3.7.2)

File Edit Format Run Options Window Help

```
from base import urut
class mahasiswa():
    def __init__(self, nama, nim, kota, us):
        self.nama=nama
        self.nim=nim
        self.kota=kota
        self.us=us
mh1 = mahasiswa("Rozin", 135, "Solo", 10000)
mh2 = mahasiswa("Rozin1", 134, "Solo", 13000)
mh3 = mahasiswa("Rozin2", 135, "Solo", 5000)
mh4 = mahasiswa("Rozin3", 133, "Solo", 12000)
mh5 = mahasiswa("Rozin4", 132, "Solo", 2000)

nimMH = [mh1.nim, mh2.nim, mh3.nim, mh4.nim, mh5.nim]
usMH = [mh1.us, mh2.us, mh3.us, mh4.us, mh5.us]

a1 = urut(nimMH)
b2 = urut(usMH)

a1.printMerge(nimMH)
b2.printMerge(usMH)

a1.printQuick(nimMH)
b2.printQuick(usMH)
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.19
16 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informatio
n.
>>>
========== RESTART: C:\Users\evolved\Desktop\nomor1.py ==========
=======
ini merge sort
132 133 134 135 135

ini merge sort
2000 5000 10000 12000 13000

ini quick sort
132 133 134 135 135

ini quick sort
2000 5000 10000 12000 13000

>>>
```

Ln: 17 Col: 4

Nomor 3

nomor3.py - C:\Users\evolved\Desktop\nomor3.py (3.7.2)

File Edit Format Run Options Window Help

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

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

def sele(A):
    for i in range(len(A)):
        min_idx = i
        for j in range(i+1, len(A)):
            if A[min_idx] > A[j]:
                min_idx = j
        A[i], A[min_idx] = A[min_idx], A[i]

def inse(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key
def mergeSort(arr):
    if len(arr) >1:
        mid = len(arr)//2
        L = arr[:mid]
        R = arr[mid:]
        mergeSort(L)
        mergeSort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+=1
            else:
                arr[k] = R[j]
                j+=1
            k+=1
        while i < len(L):
            arr[k] = L[i]
            i+=1
            k+=1
        while j < len(R):
```

Python 3.7.2 Shell

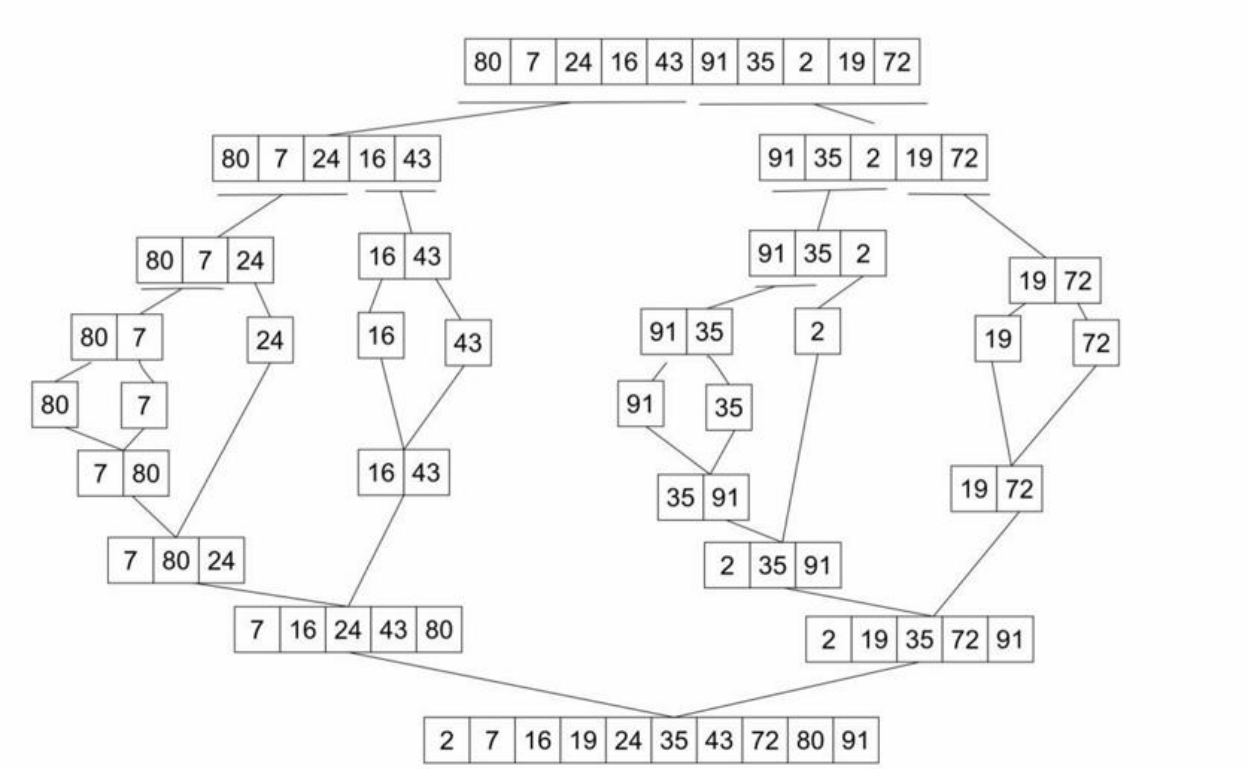
File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\evolved\Desktop\nomor3.py ==========
bubble : 3.24059 detik
selection : 1.2032 detik
insertion : 1.52656 detik
merge : 0.026068 detik
quick : 0.0125327 detik
>>>
```

Ln: 10 Col: 4

Ln: 1

Nomor 4A (Tracing Algoritma Merge Sort)



Nomor 5 (Merge Sort tanpa Slicing, menggunakan recursive)

nomor5.py - C:\Users\evolved\Desktop\nomor5.py (3.7.2)

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([11,40,9,32,44,2,122]))
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\evolved\Desktop\nomor5.py ================
[2, 9, 11, 32, 40, 44, 122]
>>>
```

Ln: 6 Col: 4

Ln: 45 Col: 37

Nomor 6 (Quick Sort dengan Median of Three)

nomor6.py - C:\Users\evolved\Desktop\nomor6.py (3.7.2)

File Edit Format Run Options Window Help

```
def quickSort(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):
    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

liste1 = list([13,5,16,125,124])

quickSort(liste1, False)
print('sorted:')
print(liste1)
```

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\evolved\Desktop\nomor6.py ==========
sorted:
[125, 124, 16, 13, 5]
>>>
```

Ln: 7 Col: 4

Nomor 7

nomor7.py - C:\Users\evolved\Desktop\nomor7.py (3.7.2)

File Edit Format Run Options Window Help

```
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):
    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 3.7.2 Shell

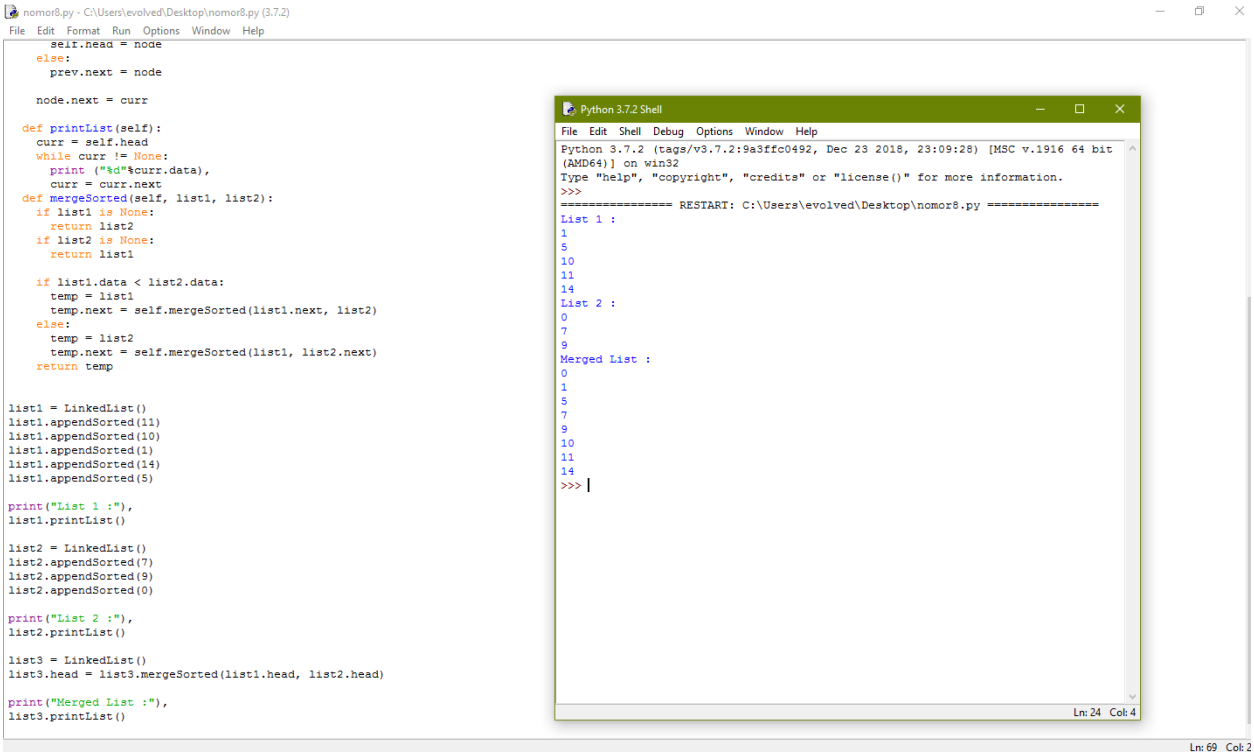
File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\evolved\Desktop\nomor7.py ==========
merge : 0.0270717 detik
quick : 0.0120313 detik
merge mod : -0.00200582 detik
quick mod : -0.0330887 detik
>>>
```

Ln: 9 Col: 4

Ln: 1 Col: 0

Nomor 8


nomor8.py - C:\Users\evolved\Desktop\nomor8.py (3.7.2)
File Edit Format Run Options Window Help
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):
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(11)
list1.appendSorted(10)
list1.appendSorted(1)
list1.appendSorted(14)
list1.appendSorted(5)
print("List 1 :"),
list1.printList()
list2 = LinkedList()
list2.appendSorted(7)
list2.appendSorted(9)
list2.appendSorted(0)
print("List 2 :"),
list2.printList()
list3 = LinkedList()
list3.head = list3.mergeSorted(list1.head, list2.head)
print("Merged List :"),
list3.printList()
Ln: 69 Col: 2
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\evolved\Desktop\nomor8.py ==========
List 1 :
1
5
10
11
14
List 2 :
0
7
9
Merged List :
0
1
5
7
9
10
11
14
>>>
Ln: 24 Col: 4