NAMA : DANANG AJI NUGROHO

NIM : L200180015

KELAS : A

TUGAS

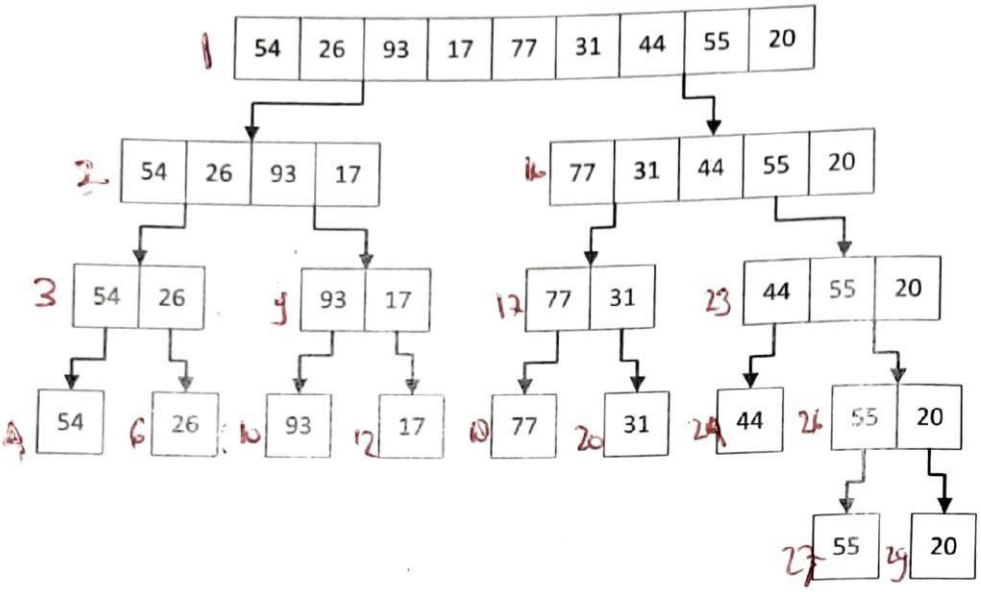
1.

```
#Nomor 1
def mergeSort(A):
    print("Membelah :",A)
    if len(A) > 1:
        mid=len(A)//2
        separuhKiri=A[:mid]
        separuhKanan=A[mid:]

        mergeSort(separuhKiri)
        mergeSort(separuhKanan)

        i=0;j=0;k=0
        while i < len(separuhKiri) and j < len(separuhKanan):
            if separuhKiri[i] < separuhKanan[j]:
                A[k]=separuhKiri[i]
                i=i+1
            else:
                A[k]=separuhKanan[j]
                j=j+1
            k=k+1

        while i < len(separuhKiri):
            A[k]=separuhKiri[i]
            i=i+1
            k=k+1
        while j < len(separuhKanan):
            A[k]=separuhKanan[j]
            j=j+1
            k=k+1
    print("Menggabungkan :",A)
```

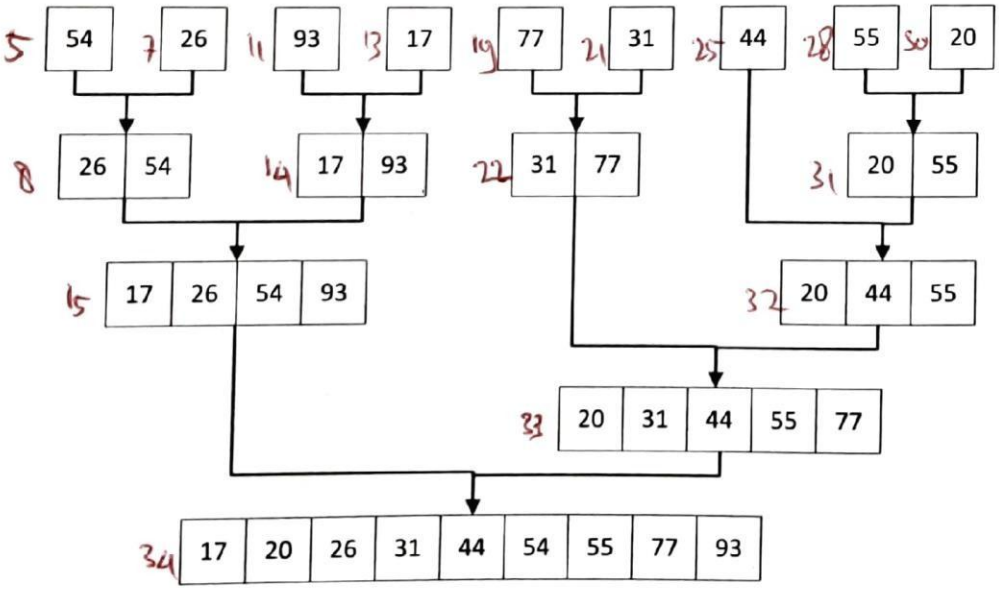
```python
def quickSort(A):
    quickSortBantu(A,0,len(A)-1)
def quickSortBantu(A, awal, akhir):
    if awal < akhir:
        titikBelah=partisi(A,awal,akhir)
        quickSortBantu(A,awal,titikBelah-1)
        quickSortBantu(A,titikBelah+1,akhir)
def partisi(A, awal,akhir):
    nilaiPivot=A[awal]
    penandaKiri=awal+1
    penandaKanan=akhir
    selesai=False
    while not selesai:

        while penandaKiri <= penandaKanan and A[penandaKiri] <= nilaiPivot:
            penandaKiri=penandaKiri+1
        while A[penandaKanan] >= nilaiPivot and penandaKanan >= penandaKiri:
            penandaKanan=penandaKanan-1
        if penandaKanan < penandaKiri:
            selesai=True
        else:
            temp=A[penandaKiri]
            A[penandaKiri]=A[penandaKanan]
            A[penandaKanan]=temp
    temp=A[awal]
    A[awal]=A[penandaKanan]
    A[penandaKanan]=temp

    return penandaKanan

daftar=[c1.NIM,c2.NIM,c3.NIM,c4.NIM,c5.NIM]

print("Hasil MergeSort")
mergeSort(daftar)
print(daftar)
quickSort(daftar)
print("\nHasil QuickSort")
print(daftar)
```

```
======= RESTART: C:/Users/ASUS/Downloads/idlex-1.18/idlex-1.18/tgs6.py =======
Hasil MergeSort
('Membelah :', [51, 2, 18, 4, 31])
('Membelah :', [51, 2])
('Membelah :', [51])
('Menggabungkan :', [51])
('Membelah :', [2])
('Menggabungkan :', [2])
('Menggabungkan :', [2, 51])
('Membelah :', [18, 4, 31])
('Membelah :', [18])
('Menggabungkan :', [18])
('Membelah :', [4, 31])
('Membelah :', [4])
('Menggabungkan :', [4])
('Membelah :', [31])
('Menggabungkan :', [31])
('Menggabungkan :', [4, 31])
('Menggabungkan :', [4, 18, 31])
('Menggabungkan :', [2, 4, 18, 31, 51])
[2, 4, 18, 31, 51]

Hasil QuickSort
[2, 4, 18, 31, 51]
```

2.



Gambar 6.1: Membelah list sampai tiap sub-list berisi satu elemen atau kosong. Sesudah itu digabung seperti ditunjukkan di Gambar 6.2.



Gambar 6.2: Menggabungkan list satu demi satu.

3.

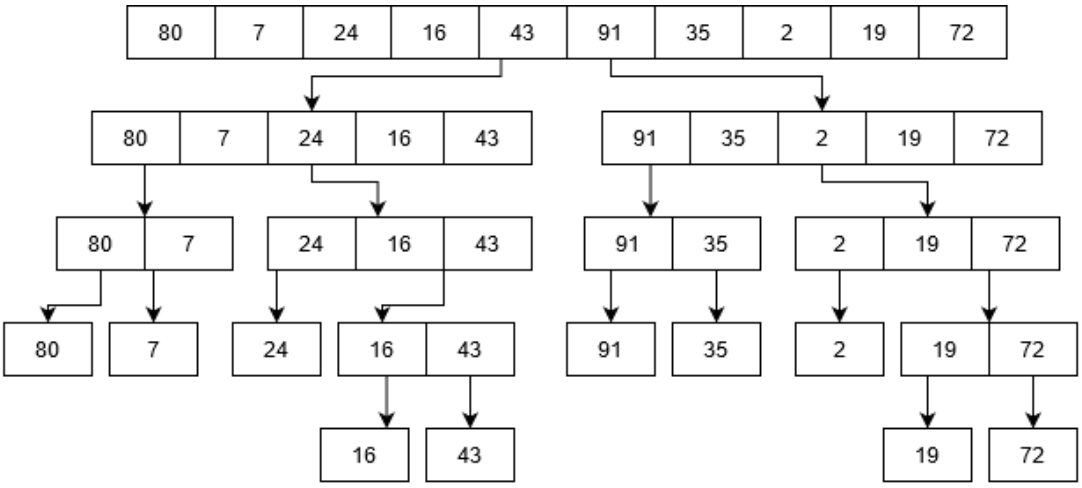
```
#Nomor 3
def swap(A,p,q):
    tmp=A[p]
    A[p]=A[q]
    A[q]=tmp

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
```

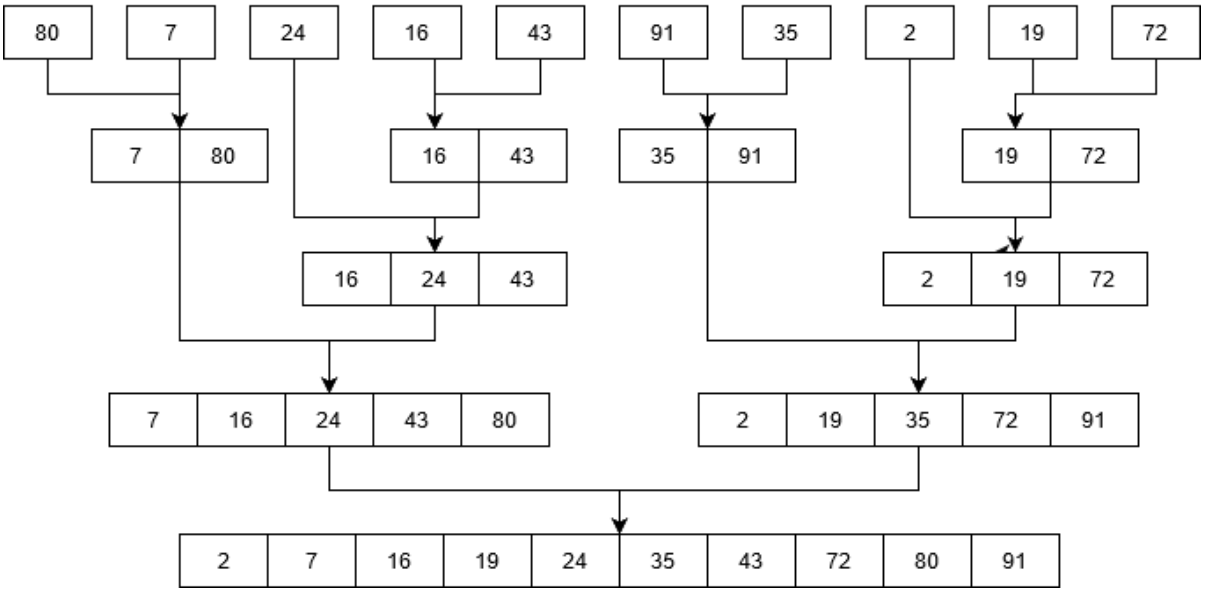
```
from time import time as detak
from random import shuffle as kocok
k=range(6000)
kocok(k)
u_bub=k[:]
u_sel=k[:]
u_ins=k[:]
u_mrg=k[:]
u_qck=k[:]

aw=detak();bubbleSort(u_bub);ak=detak();print('bubble: %g detik' %(ak-aw) );
aw=detak();selectionSort(u_sel);ak=detak();print('selection: %g detik' %(ak-aw)
aw=detak();insertionSort(u_ins);ak=detak();print('insertion: %g detik' %(ak-aw))
aw=detak();mergeSort(u_mrg);ak=detak();print('merge: %g detik' %(ak-aw));
aw=detak();quickSort(u_qck);ak=detak();print('quick: %g detik' %(ak-aw));
```

```
======= RESTART: C:/Users/ASUS/Downloads/idlex-1.18/idlex-1.18/tgs6.py =======
bubble: 5.819 detik
selection: 2.049 detik
insertion: 3.035 detik
merge: 0.126 detik
quick: 0.0550001 detik
>>> |
```

4.

4A


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


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

4B

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	

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	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.

```python
#Nomor 5
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
```

```python
    return the_list


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

print(merge_sort([3,5,2,4,1]))
```

Python 2.7.15 Shell

File Edit Shell Debug Options Window Help

```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:/Users/ASUS/Downloads/idlex-1.18/idlex-1.18/tgs6.py =======
[1, 2, 3, 4, 5]
>>>
```

6.

```
#Nomor 6
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
```

```
m = list([12,5,1,76,3

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

```
Type "copyright", "credits" or "license()" for more infor
>>>
======= RESTART: C:\Users\ASUS\Downloads\idlex-1.18\idlex
sorted:
[76, 32, 22, 12, 5, 1]
>>>
```

7.

```
#Nomor 7
from time import time as detak
from random import shuffle as kocok
k=range(6000)
kocok(k)
u_mrgM=k[:]
u_qckM=k[:]
u_mrgA=k[:]
u_qckA=k[:]

aw=detak();merge_sort(u_mrgM);ak=detak();print('merge modif: %g detik' %(ak-aw)
aw=detak();quicksort(u_qckM);ak=detak();print('quick modif: %g detik' %(ak-aw))
aw=detak();mergeSort(u_mrgA);ak=detak();print('merge asli: %g detik' %(ak-aw));
aw=detak();quickSort(u_qckA);ak=detak();print('quick asli: %g detik' %(ak-aw));
```

Python 2.7.15 Shell

File Edit Shell Debug Options Window Help

```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\ASUS\Downloads\idlex-1.18\idlex-1.18\tgs6.py =======
merge modif: 0.052 detik
quick modif: 0.039 detik
merge asli: 0.046 detik
quick asli: 0.023 detik
>>>
```

8.

```
#Nomor 8
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
```

```python
  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(7)
list1.appendSorted(5)
list1.appendSorted(4)
list1.appendSorted(6)

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

list2 = LinkedList()
list2.appendSorted(2)
list2.appendSorted(3)
list2.appendSorted(1)

print("\nList 2 :"),
list2.printList()

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

print("\nMerged List :"),
list3.printList()
```

Python 2.7.15 Shell

File Edit Shell Debug Options Wir

```
Python 2.7.15 (v2.7.15:ca079a
tel)] on win32
Type "copyright", "credits" c
>>>
======= RESTART: C:\Users\ASU
List 1 : 4 5 6 7
List 2 : 1 2 3
Merged List : 1 2 3 4 5 6 7
>>>
```