Nama : Zulfa Fajrul Falah
Nim : L200170149 / D

LAPORAN PRAKTIKUM ALGORITMA STRUKTUR DATA
MODUL 4

Nomor 1

```
################################No.1################################
class Mahasiswa(object):
    """Class Mahasiswa yang dibangun dari class Manusia."""
    def __init__(self, nama, NIM, kota, us):
        """Metode inisiasi ini menutupi metode inisiasi di class Manusia"""
        self.nama = nama
        self.NIM = NIM
        self.kotaTinggal = kota
        self.uangSaku = us

c0 = Mahasiswa('Ika',10,'Sukoharjo',240000)
c1 = Mahasiswa('Budi',51,'Sragen',230000)
c2 = Mahasiswa('Ahmad',2,'Surakarta',250000)
c3 = Mahasiswa('Chandra',18,'Surakarta',235000)
c4 = Mahasiswa('Eka',4,'Boyolali',240000)
c5 = Mahasiswa('Fandi',31,'Salatiga',250000)
c6 = Mahasiswa('Deni',13,'Klaten',245000)
c7 = Mahasiswa('Galuh',5,'Wonogiri',245000)
c8 = Mahasiswa('Janto',23,'Klaten',245000)
c9 = Mahasiswa('Hasan',64,'Karanganyar',270000)
c10 = Mahasiswa('Khalid',29,'Purwodadi',230000)

Daftar = [c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10]

def mencari(koleksi,target):
    output = []
    index = 0
    for i in koleksi:
        if i.kotaTinggal == target:
            output.append(index)
            index += 1
        else:
            index += 1
    return output

mencari(Daftar,)
```

Nomor 2

```
################################No.2################################
def cariUangSakuTerkecil(kumpulan):
    terkecil = kumpulan[0].uangSaku
    for i in kumpulan:
        if i.uangSaku < terkecil:
            terkecil = i.uangSaku
    return terkecil #kembalikan yang terkecil
```

Nomor 3

```
#######################NOMOR 3#################
def sakuKcl2(n):
    baru = n[0].saku
    list = []
    for i in range(len(n)):
        if(n[i].saku==baru):
            list.append(n[i].nama)
        elif(n[i].saku<baru):
            baru = n[i].saku
            list = []
            list.append(n[i].nama)
    return list
```

Nomor 4

```
    return list
##############################No.4#############################
def cariDaftarUangSakuKurang(kumpulan):
    b = []
    for i in kumpulan:
        if i.uangSaku < 250000:
            terkecil = i.uangSaku
            b.append(kumpulan.index(i))
    return b
```

Nomor 5

```
################################No.5#################################
class Node:
    def __init__(self,data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    def pushAw(self, data_baru):
        node_baru = Node(data_baru)
        node_baru.next = self.head
        self.head = node_baru
    def pushAk(self, data):
        if(self.head == None):
            self.head = Node(data)
        else:
            current = self.head
            while (current.next != None):
                current = current.next
            current.next = Node(data)
        return self.head
    def insert(self, data, pos):
        node = Node(data)
        if not self.head:
            self.head = node
        elif posisi == 0:
            node.next = self.head
            self.head = node
        else:
            prev = None
            current = self.head
            current_pos = 0
            while(current_pos < pos) and current.next:
                prev = current
                current = current.next
```

```
                current_pos +=1
            prev.next = node
            node.next = current
        return self.head
    def search(self, v):
        current = self.head
        while current != None:
            if current.data == v:
                return "True"
            current = current.next
        return "False"
    def display(self):
        current = self.head
        while current != None:
            print(current.data)
            current = current.next
```

Nomor 6

```
#################################No.6#################################
def binSe(kumpulan, target):
    # Mulai dari seluruh runtutan elemen
    low = 0
    high = len(kumpulan) - 1
    data = []

    # Secara berulang belah runtutan itu menjadi separuhnya
    # sampai targetnya ditemukan
    while low <= high:
        # Temukan pertengahan runtut itu
        mid = (high + low) // 2
        # Apakah pertengahannya memuat target?
        if kumpulan[mid] == target:
            data.append(kumpulan.index(target))
            return True
        # ataukah targetnya di sebelah kirinya?
        elif target < kumpulan[mid]:
            high = mid - 1
        # ataukah targetnya di sebelah kanannya?
        else:
            low = mid + 1
        # Jika runtutnya tidak bisa dibelah lagi, berarti targetnya tidak ada
    return False
```

Nomor 7

```
##################################No.7#################################
def binSearch(kumpulan, target):
    # Mulai dari seluruh runtutan elemen
    low = 0
    high = len(kumpulan) - 1
    data = []

    # Secara berulang belah runtutan itu menjadi separuhnya
    # sampai targetnya ditemukan
    while low != high:
        # Temukan pertengahan runtut itu
        mid = (high + low) // 2
        # Apakah pertengahannya memuat target?
        if kumpulan[mid] == target:
            break
        # ataukah targetnya di sebelah kirinya?
        elif target < kumpulan[mid]:
            high = mid - 1
        # ataukah targetnya di sebelah kanannya?
        else:
            low = mid + 1
    for i in range(low, high):
        if target == kumpulan[i]:
            data.append(i)
    return data


a = [2, 3, 5, 6, 6, 6, 8, 9, 9, 10, 11, 12, 13, 13, 14]
```

Nomor 8

```
##############################No.8##############################
def binSearching(kumpulan, target):
    # Mulai dari seluruh runtutan elemen
    low = 0
    high = len(kumpulan) - 1

    # Secara berulang belah runtutan itu menjadi separuhnya
    # sampai targetnya ditemukan
    while low <= high:
        # Temukan pertengahan runtut itu
        mid = (high + low) // 2
        # Apakah pertengahannya memuat target?
        if kumpulan[mid] == target:
            return mid
        elif kumpulan[mid] < target:
            high = mid + 1
        else:
            low = mid - 1

    return -1


b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

"""
untuk mencari berapa jumlah tebakan yang digunakan oleh Binary Search
yaitu dengan menggunakan Logaritma basis 2 (log2(n))
misalkan :
        // apabila terdapat elemen array berjumlah 100 maka memiliki maksimal 7 kali tebakan
           itu dikarenakan log2(100) = 6.643856189774725 sehingga diperoleh angka 7
           dapat juga diperoleh dari log2(128) = 7 karena yang mendekati dari 100 adalah 128
       //  apabila terdapat elemen array berjumlah 1000 maka memiliki maksimal 10 kali tebakan
           itu dikarenakan log2(1000) = 9.965784284662087 sehingga diperoleh angka 10
           dapat juga diperoleh dari log2(1024) = 10 karena yang mendekati dari 1000 adalah 128
"""
```