Nama : Reza Aristo Rifandi

NIM : L200180206

Kelas : H

MODUL 4 Praktikum Algoritma dan Sturktur Data

Nomor 1

```
class Mahasiswa(object):
    def __init__(self, nama, usia, kota, us):
        self.nama = nama
        self.usia = usia
        self.kotaTinggal = kota
        self.uangSaku = us
    def __str__(self):
        s = self.nama + ", usia " + str(self.usia) + "tahun" \
            + ". Tinggal di " + self.kotaTinggal \
            + ". Uang saku " + self.uangsaku + " tiap bulannya."
        return s
    def ambilNama(self):
        return self.nama
    def ambilNIM(self):
        return self.NIM
    def ambilUangSaku(self):
        return self.uangSaku

a1=Mahasiswa("Andi",19,"Salatiga",250000)
a2=Mahasiswa("Riska",27,"Surakarta",550000)
a3=Mahasiswa("Kiana",6,"Ngawi",50000)
a4=Mahasiswa("Harun",10,"Jakarta",100000)
a5=Mahasiswa("Tyas",32,"Surabaya",750000)
a6=Mahasiswa("Joko",29,"Malang",650000)
a7=Mahasiswa("Fahrur",34,"Ngawi",8250000)
a8=Mahasiswa("Junet",20,"Salatiga",400000)
a9=Mahasiswa("Putri",23,"Ngawi",480000)
a10=Mahasiswa("Fara",60,"Surabaya",950000)
a11=Mahasiswa("Erwan",21,"Salatiga",365000)
Daftar = [a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11]

def cariKota(target):
    p=[]
    for i in Daftar :
        if i.kotaTinggal==target:
            p.append(Daftar.index(i))
    print (p)
```

Nomor 2

```
def cariUsKcl():
    n = len(Daftar)
    p=[]
    minimum=Daftar[0].uangSaku
    for i in Daftar:
        if i.uangSaku == minimum :
            p.append(i.nama)
            minimum=i.uangSaku
        elif i.uangSaku < minimum :
            p=[]
            minimum=i.uangSaku
    print (p, ' adalah mahasiswa dengan uang saku terkecil dengan nominal ', minimum)
```

Nomor 3

```
def cariUsKcl2():
    n = len(Daftar)
    p=[]
    minimum=Daftar[0].uangSaku
    for i in Daftar:
        if i.uangSaku == minimum :
            p.append(i)
            minimum=i.uangSaku
        elif i.uangSaku < minimum :
            p=[]
            p.append(i)
            minimum=i.uangSaku
    for i in range(len(p)):
        print(p[i])
```

Nomor 4

```
def kurang():
    p=[]
    maksimum=Daftar[0].uangSaku
    for i in Daftar:
        if i.uangSaku < 250000 :
            p.append(i)
    for i in range(len(p)):
        print(p[i])
```

Nomor 5

```
class node(object):
    def __init__ (self, data, next = None):
        self.data = data
        self.next = next

    def cariLinkedList(self, dicari):
        cNode = self
        while cNode is not None:
            if cNode.next != None:
                if cNode.data != dicari:
                    cNode = cNode.next
                else:
                    print ("Data", dicari, "ada dalam Linked List")
                    break
            elif cNode.next == None:
                print ("Data", dicari, "tidak ada dalam Linked List")
                break
u = node(97)
menu = u
u.next = node (42)
u = u.next
u.next = node (63)
u = u.next
u.next = node (59)
```

Nomor 6

```
def binSe(kumpulan, target):
    low = 0
    high = len(kumpulan) - 1
    while low<=high:
        mid = (high+low) // 2
        if kumpulan[mid] == target:
            return mid

        elif target < kumpulan[mid]:
            high = mid - 1
        else:
            low = mid + 1
    return False
```

Nomor 7

```
def binSe_7(kumpulan, target):
    temp = []
    low = 0
    high = len(kumpulan)-1
    while low <= high :
        mid = (high+low)//2
        if kumpulan[mid] == target:
            midKiri = mid-1
            while kumpulan[midKiri] == target:
                temp.append(midKiri)
                midKiri = midKiri-1
            temp.append(mid)
            midKanan = mid+1
            while kumpulan[midKanan] == target:
                temp.append(midKanan)
                midKanan = midKanan+1
            return temp
        elif target < kumpulan[mid]:
            high = mid-1
        else:
            low = mid+1
    return False
```

Nomor 8

```
"""Dalam kasus ini menggunakan konsep Big-O. Yang mana rumus yang dipakai adalah rumus O(log n)
dengan rincian 1 = 1, 2 = 2, 4 = 3, 10 = 4, 100 = 7, 1000=10.
Yang nantinya log itu berasal dari pangkat log berbasis 2. Sehingga dapat mengetahui jumlah
maksimal tebakan.
Untuk pola adalah sebagai berikut:
        apabila ingin menebak angka 80

        a = nilai tebakan pertama // 2
        tebakan selanjutnya = nilai tebakan "lebih dari" + a
        *jika hasil tebakan selanjutnya "kurang dari", maka nilai yang dipakai
        tetap nilai lebih dari sebelumnya*
        a = a // 2

    Untuk simulasinya adalah sebagai berikut
        tebakan ke 1: 50 (mengambil nilai tengah) jawaban= "itu terlalu kecil"
        tebakan ke 2: 90 (dari 50 + 25) jawaban = "itu terlalu besar"
        tebakan ke 3: 70 (dari 50 + 12) jawaban = "itu terlalu kecil"
        tebakan ke 4: 78 (dari 72 + 6) jawaban = "itu terlalu besar"
        tebakan ke 5: 81 (dari 78 + 3) jawaban = "itu terlalu besar"
        tebakan ke 6: 79 (dari 78 + 1) jawaban = "itu terlalu kecil"
        tebakan ke 7: antara 81 dan 79 hanya ada 1 angka yaitu 80
        jadi angka yang harus ditebak pada tebakan ketujuh adalah angka 80"""
```