Nama : Nur Taufiq Hidayat
NIM : L200180069
Kelas : C

MODUL 4 Algostruk Tugas

PENCARIAN

4.3 Soal-soal untuk Mahasiswa

1. Membuat fungsi yang mengembalikan list berisi semua index lokasi elemen yang dicari, jika yang dicari tidak ditemukan, maka akan mengembalikan list kosong.

```
def cariKotaTinggal(list, target):
    a = []
    for i in list :
        if i.kotaTinggal == target:
            a.append(list.index(i))
    return a
```

Outputnya :

```
>>> a = cariKotaTinggal(Daftar, "Klaten")
>>> print(a)
[6, 8]
>>> b = cariKotaTinggal(Daftar, 'Ngawi')
>>> print(b)
[]
```

2. Dari list daftar mahasiswa, buat fungsi untuk menemukan uang saku terkecil diantaranya.

```
def cariUangSakuTerkecil(list):
    terkecil = Daftar[0].uangSaku
    for i in range (len(Daftar)):
        if terkecil > Daftar[i].uangSaku:
            terkecil = Daftar[i].uangSaku
    return terkecil
```

Outputnya :

```
>>> cariUangSakuTerkecil()
230000
```

3. Ubah program agar mengembalikan objek mahasiswa yang mempunyai uang saku terkecil. Jika ada lebih dari satu mahasiswa yang uang sakunya terkecil, semua objek mahasiswa itu dikembalikan.

```
def cariUangSakuTerkecilObject(list):
    terkecil = Daftar[0].uangSaku
    x = []
    a = cariUangSakuTerkecil
    for i in range (len(Daftar)):
        if terkecil > Daftar[i].uangSaku:
            terkecil = Daftar[i].uangSaku

    for i in range (len(Daftar)):
        if Daftar[i].uangSaku == terkecil:
            x.append(Daftar[i].nama)
    return x
```

Outputnya :

```
>>> cariUangSakuTerkecilObject(Daftar)
['Budi']
```

Jika data uang saku Chandra diganti, outputnya :

```
>>> d = cariUangSakuTerkecilObject(Daftar)
>>> print(d)
['Budi', 'Chandra']
```

4. Membuat fungsi yang mengembalikan semua objek mahasiswa yang uang sakunya kurang dari 250000.

```
def cariUSKurang250k(list):
    x = []
    for i in range(len(list)):
        if list[i].uangSaku < 250000:
            x.append(list[i].nama)
    return x
```

Outputnya :

```
>>> cariUSKurang250k(Daftar)
['Ika', 'Budi', 'Chandra', 'Eka', 'Deni', 'Galuh', 'Janto']
```

5. Membuat suatu program untuk mencari suatu item di sebuah linked list.

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

    def cariLinkedList(self, dicari):
        curNode = self
        while curNode is not None:
            if curNode.next != None:
                if curNode.data != dicari:
                    curNode = curNode.next
                else:
                    print ("Data", dicari, "ada dalam Linked List")
                    break
            elif curNode.next == None:
                print ("Data", dicari, "tidak ada dalam Linked List")
                break
```

Outputnya :

```
>>> a = node(10)
>>> x = a
>>> a.next = node(15)
>>> a = a.next
>>> a.next = node(9)
>>> a = a.next
>>> a.next = node(18)
>>> x.cariLinkedList(9)
Data 9 ada dalam Linked List
>>> x.cariLinkedList(13)
Data 13 tidak ada dalam Linked List
>>> x.cariLinkedList(15)
Data 15 ada dalam Linked List
```

6. Binary search mengubah fungsi binSe di halaman 43 agar mengembalikan index lokasi elemen yang ditemukan. Kalu tidak ketemu, akan mengembalikan False.

```
def binSe(kumpulan, target):
    low = 0
    high = len(kumpulan)-1
    while low <= high:
        mid = (high+low)//2
        if kumpulan[mid] == target:
            return "Target pada indeks " + str(mid)
        elif target < kumpulan[mid]:
            high = mid-1
        else:
            low = mid+1
    return False
```

Outputnya :

```
>>> A = [2, 4, 5, 10, 13, 18, 23, 29, 31, 51, 64]
>>> binSe(kumpulan, 10)
'Target pada indeks 3'
>>> binSe(kumpulan, 6)
False
```

7. Binary search. Mengubah fungsi binSe agar mengembalikan semua index lokasi elemen yang ditemukan.

```
def binSe2(kumpulan, target):
    x = []
    low = 0
    high = len(kumpulan)-1
    while low <= high :
        mid = (high+low)//2
        if kumpulan[mid] == target:
            midKiri = mid-1
            while kumpulan[midKiri] == target:
                x.append(midKiri)
                midKiri = midKiri-1
            x.append(mid)
            midKanan = mid+1
            while kumpulan[midKanan] == target:
                x.append(midKanan)
                midKanan = midKanan+1
            return x
        elif target < kumpulan[mid]:
            high = mid-1
        else:
            low = mid+1
    return False
```

Outputnya :

```
>>> G = [2, 4, 5, 6, 6, 6, 8, 9, 9, 10, 11, 12, 13, 13, 14]
>>> binSe2(G, 6)
[3, 4, 5]
```

8. Pada permainan tebak angka yang sudah pernah dibuat di Modul 1, kalu angka yang harus ditebak antara 1 dan 100, seharusnya maksimal jumlah tebakan adalah 7. Kalau antara 1 dan 1000, maksimal jumlah tebakan adalah 10. Mengapa seperti itu? Bagaimanakah polanya?
   Jawab :
   Karena menggunakan konsep Big-O, yang dipakai adalah rumus O(log n) dengan rincian 1 = 1, 2 = 2, 4 = 3, 10 = 4, 100 = 7, 1000 = 10. Dimana log berasal dari pangkat log berbasis 2. Jadi dapat mengetahui jumlah maksimal tebakan.

   Untuk pola nya :
   Apabila ingin menebak angka 70

   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

Simulasinya :

- tebakan ke 1: 50 (mengambil nilai tengah) jawaban= "lebih dari itu"
- tebakan ke 2: 75 (dari 50 + 25) jawaban = "kurang dari itu"
- tebakan ke 3: 62 (dari 50 + 12) jawaban = "lebih dari itu"
- tebakan ke 4: 68 (dari 62 + 6) jawaban = "lebih dari itu"
- tebakan ke 5: 71 (dari 68 + 3) jawaban = "kurang dari itu"
- tebakan ke 6: 69 (dari 68 + 1) jawaban = "lebih dari itu"
- tebakan ke 7: antara 71 dan 69 hanya ada 1 angka = 70!!!