LAPORAN PRAKTIKUM ALGORITMA STRUKTUR DATA

MODUL 04

"PENCARIAN"

NAMA : AIZA FRAVY QANZA
NIM : L200170144
KELAS : D

Soal-Soal untuk Mahasiswa

1. Buatlah suatu fungsi pencarian yang, alih-alih mengembalikan True/False, mengembalikan semua index lokasi elemen yang dicari. Kalau yang dicari tidak ditemukan, fungsi ini akan mengembalikan list kosong.

```
>>> class mhsTIF():
    def __init__(self, nama, asal, saku):
        self.nama = nama
        self.asal = asal
        self.saku = saku


>>> def cari(n):
        baru = []
        for i in range(len(n)):
                if(n[i].asal.lower() == 'samarinda'):
                        baru.append(i)
        return baru

>>> c0 = mhsTIF('Aiza','Samarinda',240000)
>>> c1 = mhsTIF('Bella','Jakarta',290000)
>>> c2 = mhsTIF('Chiara','Balikpapan',250000)
>>> c3 = mhsTIF('Devia','Bandung',230000)
>>> c4 = mhsTIF('Elvira','Surabaya',235000)
>>> c5 = mhsTIF('Farah','Bandung',220000)
>>> c6 = mhsTIF('Gege','Tarakan',260000)
>>> daftar = [c0,c1,c2,c3,c4,c5,c6]
>>> cari(daftar)
[0]
```

2. Dari list daftar mahasiswa, buatlah fungsi untuk menemukan uang saku yang terkecil.

```
>>> c0 = mhsTIF('Aiza','Samarinda',240000)
>>> c1 = mhsTIF('Bella','Jakarta',290000)
>>> c2 = mhsTIF('Chiara','Balikpapan',250000)
>>> c3 = mhsTIF('Devia','Bandung',230000)
>>> c4 = mhsTIF('Elvira','Surabaya',235000)
>>> c5 = mhsTIF('Farah','Bandung',220000)
>>> c6 = mhsTIF('Gege','Tarakan',260000)
>>> daftar = [c0,c1,c2,c3,c4,c5,c6]
```

```
>>> def sakuTerkecil(n):
        baru = n[0].saku
        for i in range(len(n)):
                if(n[i].saku<baru):
                        baru = n[i].saku
        return baru

>>> sakuTerkecil(daftar)
220000
```

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

```
>>> c0 = mhsTIF('Aiza','Samarinda',240000)
>>> c1 = mhsTIF('Bella','Jakarta',290000)
>>> c2 = mhsTIF('Chiara','Balikpapan',250000)
>>> c3 = mhsTIF('Devia','Bandung',230000)
>>> c4 = mhsTIF('Elvira','Surabaya',235000)
>>> c5 = mhsTIF('Farah','Bandung',220000)
>>> c6 = mhsTIF('Gege','Tarakan',260000)
>>> daftar = [c0,c1,c2,c3,c4,c5,c6]
```

```
>>>def sakuTerkecil2(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

>>> sakuTerkecil2(daftar)
['Farah']
```

4. Buatlah suatu program yang mengembalikan semua objek mahasiswa yang uang sakunya kurang dari 250000.

```
>>> c0 = mhsTIF('Aiza','Samarinda',240000)
>>> c1 = mhsTIF('Bella','Jakarta',290000)
>>> c2 = mhsTIF('Chiara','Balikpapan',250000)
>>> c3 = mhsTIF('Devia','Bandung',230000)
>>> c4 = mhsTIF('Elvira','Surabaya',235000)
>>> c5 = mhsTIF('Farah','Bandung',220000)
>>> c6 = mhsTIF('Gege','Tarakan',260000)
>>> daftar = [c0,c1,c2,c3,c4,c5,c6]
```

```
>>> def sakuKurang(n):
        batas = 250000
        list = []
        for i in range(len(n)):
                if(n[i].saku < batas):
                        list.append(n[i].nama)
        return list

>>> sakuKurang(daftar)
['Aiza', 'Devia', 'Elvira', 'Farah']
```

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

```
>>> class LinkedList:
        def __init__(self):
                self.head = None
        def pushAw(self, new_data):
                new_node = Node(new_data)
                new_node.next = self.head
                self.head = new_node
                return self.head
        def search(self, x):
                current = self.head
                while current != None:
                        if current.data == x:
                                return "True"
                        current = current.next
                return "False"
        def display(self):
                current = self.head
                while current is not None:
                        print(current.data, end = ' ')
                        current = current.next


>>> k=LinkedList()
>>> k.pushAw(12)
<__main__.Node object at 0x00000176071279B0>
>>> k.pushAw(23)
<__main__.Node object at 0x000001760729BA90>
>>> k.pushAw(10)
<__main__.Node object at 0x000001760729BCF8>
>>> k.pushAw(19)
<__main__.Node object at 0x00000176072ABBE0>
>>> k.pushAw(12)
<__main__.Node object at 0x00000176072ABC18>
>>> k.pushAw(34)
<__main__.Node object at 0x00000176072ABC50>
>>> k.search(23)
'True'
>>> k.search(8)
'False'
```

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

```
>>> def binSe(list, target):
        low = 0
        high = len(list) - 1
        while(low<=high):
                mid = (low+high)//2
                if(list[mid] == target):
                        return "Target di index "+str(mid)
                elif(target<list[mid]):
                        high = mid - 1
                else:
                        low = mid +1
        return "Target tidak ditemukan"

>>> l=[1,2,3,4,5,6,7,8,9]
>>> target=10
>>> binSe(l, target)
'Target tidak ditemukan'
>>> target=22
>>> binSe(l, target)
'Target tidak ditemukan'
>>> target=1
>>> binSe(l, target)
'Target di index 0'
```

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

```
>>> def binSe(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

>>> kumpulan=[1,3,5,7,9,11,13,15]
>>> target = 1
>>> binSe(kumpulan,target)
[0]
>>> target = 5
>>> binSe(kumpulan,target)
[2]
>>> target = 10
>>> binSe(kumpulan,target)
False
```

8. Pada permainan tebak angka yang sudah kamu buat di Modul 1 (soal nomer 12, halaman 16), kalau angka yang harus ditebak berada diantara 1 dan 100, seharusnya maksimal jumlah tebakan adalah 7. Kalau antara 1 dan 1000, maksimal jumlah tebakan adalah 10. Mengapa seperti itu? Bagaimanakah polanya?

   Ada 2 kemungkinan pola yang bisa digunakan.
   Pola pertama :
   a = nilai tebakan pertama // 2
   tebakan selanjutnya = nilai tebakan "Lebih dari" + a dan jika hasil tebakan selanjutnya "Kurang dari", maka nilai yang dipakai tetap nilai sebelumnya
   a = a // 2
   Misalkan, angka yang akan ditebak adalah 60.
   tebakan 1 : 50 (mengambil nilai tengah) jawaban "Lebih dari itu"
   tebakan 2 : 72 (lebih dari 50) jawaban "Kurang dari itu"
   tebakan 3 : 62 (kurang dari 72) jawaban "Kurang dari itu"
   tebakan 4 : 55 (kurang dari 62) jawaban "Lebih dari itu"
   tebakan 5 : 59 (lebih dari 55) jawaban "Lebih dari itu"
   tebakan 6 : 61 (lebih dari 59) jawaban "Kurang dari itu"
   tebakan 7 : antara 61 dan 59, jadi jawabannya 60

   Pola kedua :
   Menggunakan barisan geometri Sn = 2^n
   Barisan yang terjadi 2, 4, 8, 16, 32
   Misal angka yang akan ditebak adalah 62
   tebakan 1 : 60 jawaban "Lebih dari itu"
   tebakan 2 : 92 (60 + 32) jawaban "Kurang dari itu"
   tebakan 3 : 76 (60 + 16) jawaban "Kurang dari itu"
   tebakan 4 : 68 (60 + 8) jawaban "Kurang dari itu"
   tebakan 5 : 64 (60 + 4) jawaban "Lebih dari itu"
   tebakan 6 : 62 (60 + 2) jawaban "Tepat sekali"