PRAKTIKUM DASAR PEMROGRAMAN
PERTEMUAN KE-10

TREE

TUJUAN PRAKTIKUM

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu merealisasikan Tree ke dalam bahasa pemrograman Python serta mengaplikasikan masalah/kasus yang perlu diselesaikan dengan menggunakan set.

TOOLS

Tools yang diperlukan untuk melakukan praktikum ini adalah interpreter Python yang telah terinstall di komputer.

MATERI PRAKTIKUM:

Sebuah POHON adalah himpunan terbatas tidak kosong, dengan elemen yang dibedakan sebagai berikut

1. Sebuah elemen dibedakan dari yang lain, yang disebut sebagai AKAR dari pohon
2. Elemen yang lain (jika masih ada) dibagi-bagi menjadi beberapa sub himpunan yang disjoint, dan masing-masing sub himpunan tersebut adalah POHON yang disebut sebagai SUB POHON dari POHON yang dimaksud.

Berikut adalah langkah-langkah pada praktikum ini:

1. Buatlah sebuah file python bernama `treeNaire_<nim>.py`, contohnya: `treeNaire_24060119110023.py`.
2. Tuliskan nama file, deskripsi, pembuat, dan tanggal pada bagian awal file Anda sebagai komentar.
3. Agar dapat menggunakan konstruktor, selektor, dan fungsi yang telah dibuat pada list, lakukan import dengan menambahkan baris kode seperti berikut:

```
from List import *
```

File list dan lol berada dalam folder yang sama.

4. Buatlah fungsi kontsruktor dan selector untuk Tree.

5. Buatlah fungsi predikat khusus dalam list of list seperti berikut:

```
8   #DEFINISI DAN SPESIFIKASI KONSTRUKTOR
9   def makePN(A,PN):
10      return [A,PN]
11
12  #DEFINISI DAN SPESIFIKASI SELEKTOR
13  #Akar : PohonN-ner tidak kosong → Elemen
14  # { Akar(P) adalah Akar dari P. Jika P adalah (A,PN) = Akar(P) adalah A }
15  def akar(PN):
16      return PN[0]
17
18  #Anak : PohonN-ner tidak kosong → list of PohonN-ner
19  #{ Anak(P) adalah list of pohon N-ner yang merupakan anak-anak (sub phon)
20  # dari P. Jika P adalah (A, PN) = Anak (P) adalah PN }
21  def anak(PN):
22      return PN[1]
```

6. Buatlah fungsi predikat tree

```python
#DEFINISI DAN SPESIFIKASI PREDIKAT
#IsTreeNEmpty : PohonN-ner → boolean
# {IsTreeNEmpty(PN) true jika PN kosong : () }
def isTreeNEmpty(PN):
    return PN == []

#IsOneElmt : PohonN-ner → boolean
#{IsOneElmt(PN) true jika PN hanya terdiri dari Akar }
def isOneElmt(PN):
    return (isTreeNEmpty(PN) == False) and (isTreeNEmpty(anak(PN)) == True)
```

7. Fungsi-fungsi tambahan.

```python
#NbNElmt : PohonN-ner → integer ≥ 0
# {NbNElmt(P) memberikan banyaknya node dari pohon P :
# Basis 1: NbNElmt ((A)\) = 1
# Rekurens : NbNElmt ((A,PN)) = 1 + NbElmt(PN) }
def NbNElmt(PN):
    # Basis: Jika pohon kosong
    if isTreeNEmpty(PN):
        return 0

    # Jika hanya ada satu elemen (akar saja)
    if isOneElmt(PN):
        return 1

    # Hitung 1 untuk akar, dan rekursif pada setiap anak pohon
    # Tanpa menggunakan loop, kita memanggil fungsi untuk setiap anak secara rekursif
    # Pertama pada anak pertama
    return 1 + NbNElmt(anak(PN)[0]) + NbNElmtChild(anak(PN)[1:])

# Fungsi tambahan untuk menghitung jumlah elemen pada sisa anak-anak
def NbNElmtChild(PN):
    # Basis: Jika tidak ada anak
    if isTreeNEmpty(PN):
        return 0

    # Jika ada anak, rekursif pada anak pertama dan sisa anak-anak
    return NbNElmt(PN[0]) + NbNElmtChild(PN[1:])
```

```python
# Fungsi tambahan untuk menghitung jumlah elemen pada sisa anak-anak
def NbNElmtChild(PN):
    # Basis: Jika tidak ada anak
    if isTreeNEmpty(PN):
        return 0

    # Jika ada anak, rekursif pada anak pertama dan sisa anak-anak
    return NbNElmt(PN[0]) + NbNElmtChild(PN[1:])

def NbNDaun(PN):
    # Basis: Jika pohon kosong
    if isTreeNEmpty(PN):
        return 0

    # Jika pohon adalah daun (anak kosong)
    if isOneElmt(PN) and isTreeNEmpty(anak(PN)):
        return 1

    # Rekursi pada akar dan anak-anak
    return NbNDaunChild(anak(PN))

# Fungsi tambahan untuk menghitung jumlah daun pada sisa anak-anak
def NbNDaunChild(PN):
    # Basis: Jika tidak ada anak
    if isTreeNEmpty(PN):
        return 0

    # Jika ada anak, rekursif pada anak pertama dan sisa anak-anak
    return NbNDaun(PN[0]) + NbNDaunChild(PN[1:])
```

8. Setiap kali merealisasikan sebuah fungsi, aplikasikan fungsi tersebut, sehingga dapat diketahui apakah realisasi sudah sesuai dengan spesifikasi yang diharapkan.
   Contoh aplikasi fungsi berikut dipanggil di dalam fungsi `print()` agar hasil aplikasi atau output dapat langsung ditampilkan pada output program.
   APLIKASI

```
#APLIKASI
T = makePN(2,[])
print(makePN(2,[]))
print(isTreeNEmpty(T))
print(isOneElmt(T))
T2 = makePN('A',[makePN('B',[makePN('D',[]),makePN('E',[]),makePN('F',[])]),makePN('C',[makePN('G',[]),makePN('H',[makePN('I',[])])])])
print(T2)
print(NbNElmt(T2))
print(NbNDaun(T2))
```

   OUTPUT

```
[2, []]
False
True
['A', [['B', [['D', []], ['E', []], ['F', []]]], ['C', [['G', []], ['H', [['I', []]]]]]]]
9
5
```

**********Selamat Mengerjakan dan Berlatih **********