Tugas Type Bentukan.py

```
# Nama: Muhammad Farhan Abdul Azis
# Kelas: D
# NIM: 24060124140166

# NO 1 TIPE BENTUKAN PECAHAN CAMPURAN
# ================================================
# ================================================

# DEFINISI DAN SPESIFIKASI TYPE
# tipe pCamp : <bil: integer, n:integer >= 0, d:integer > 0 >
#   {<bil,n,d> adalah sebuah pecahan campuran dengan bil dapat bernilai positif, negatif, maupun nol,
#       n selalu bernilai lebih kecil dari d, dengan d lebih dari nol.}
# type pBiasa : <n:integer, d: integer > 0>
#   {<n,d> adalah pecahan biasa dengan n bisa bernilai negatif, nol, maupun positif, dan d lebih besar dari nol.}
#
# DEFINISI DAN SPESIFIKASI SELEKTOR
# B : pCamp --> integer
#   {B(F) memberikan bilangan bil dari pecahan campuran F, <bil,n,d>.}
# Pc : pCamp --> integer >= 0
#   {Pc(F) memberikan pembilang n dari pecahan campuran F, <bil,n,d>.}
# Py : pCamp --> integer > 0
#   {Py(F) memberikan penyebut d dari pecahan campuran F, <bil,n,d>.}
# Pb : pBiasa --> integer
#   {Pb(FR) memberikan pembilang n dari pecahan FR, <n,d>.}
# PyB : pBiasa --> integer > 0
#   {PyB(FR) memberikan penyebut d dari pecahan FR, <n,d,>.}
#
# DEFINISI DAN SPESIFIKASI KONSTRUKTOR
# MkPC : integer, integer >= 0, integer > 0 --> pCamp
#   {MkPC(bil,n,d) membentuk pecahan campuran, <bil,n,d>.}
# MkP : integer, integer > 0 --> pBiasa
#   {MkP(n,d) membentuk pecahan, <n,d>.}
#
# DEFINISI DAN SPESIFIKASI FUNGSI/OPERATOR TERHADAP PECAHAN CAMPURAN
# ToP : pCamp --> pBiasa
#   {ToP(F) mengonversi sebuah pecahan campuran menjadi pecahan.}
# ToR : pCamp --> real
#   {ToR(F) mengonversi sebuah pecahan campuran menjadi bilangan real.}
# AddP : 2 pCamp --> pCamp
#   {AddP(F1,F2) menambahkan pecahan campuran F1 dengan pecahan campuran F2.}
# SubP : 2 pCamp --> pCamp
#   {SubP(F1,F2) mengurangi pecahan campuran F1 dengan pecahan campuran F2.}
# DivP : 2 pCamp --> pCamp
#   {DivP(F1,F2) membagi pecahan campuran F1 dengan pecahan campuran F2.}
# MulP : 2 pCamp --> pCamp
```

```
#   {MulP(F1,F2) mengalikan pecahan campuran F1 dengan pecahan campuran F2}
#
# DEFINISI DAN SPESIFIKASI PREDIKAT
# EqP? : 2 pCamp --> boolean
#   {EqP?(F1,F2) bernilai benar jika pecahan campuran F1 sama dengan pecahan campuran F2.}
# LtP? : 2 pCamp --> boolean
#   {LtP?(F1,F2) bernilai benar jika pecahan campuran F1 kurang dari pecahan campuran F2.}
# GtP? : 2 pCamp --> boolean
#   {GtP?(F1,F2) bernilai benar jika pecahan campuran F1 lebih dari pecahan campuran F2.}
#
# REALISASI DALAM PYTHON

def B(F):
    return F[0]

def Pc(F):
    return F[1]

def Py(F):
    return F[2]

def Pb(FR):
    return FR[0]

def PyB(FR):
    return FR[1]

def MkPC(bil, n, d):
    return [bil, n, d]

def MkP(n, d):
    return [n, d]

def ToP(F):
    if B(F) < 0:
        return MkP(
            B(F) * Py(F) - Pc(F),
            Py(F)
        )
    else:
        return MkP(
            B(F) * Py(F) + Pc(F),
            Py(F)
        )

def ToR(F):
    if B(F) < 0:
        return B(F) - Pc(F) / Py(F)
    else:
```

```
        return B(F) + Pc(F) / Py(F)

def AddP(F1, F2):
    return MkPC(
        B(F1) + B(F2),
        Pc(F1) * Py(F2) + Pc(F2) * Py(F1),
        Py(F1) * Py(F2)
    )

def SubP(F1, F2):
    return MkPC(
        B(F1) - B(F2),
        Pc(F1) * Py(F2) - Pc(F2) * Py(F1),
        Py(F1) * Py(F2)
    )

def DivP(F1, F2):
    return MkPC(
        Pb(ToP(F1)) * PyB(ToP(F2)) //
        (PyB(ToP(F1)) * Pb(ToP(F2))),
        Pb(ToP(F1)) * PyB(ToP(F2)) %
        (PyB(ToP(F1)) * Pb(ToP(F2))),
        PyB(ToP(F1)) * Pb(ToP(F2))
    )

def MulP(F1, F2):
    return MkPC(
        Pb(ToP(F1)) * Pb(ToP(F2)) //
        (PyB(ToP(F1)) * PyB(ToP(F2))),
        Pb(ToP(F1)) * Pb(ToP(F2)) %
        (PyB(ToP(F1)) * PyB(ToP(F2))),
        PyB(ToP(F1)) * PyB(ToP(F2))
    )

def EqP(F1, F2):
    return (Pb(ToP(F1)) * PyB(ToP(F2))
            == Pb(ToP(F2)) * PyB(ToP(F1)))

def LtP(F1, F2):
    return (Pb(ToP(F1)) * PyB(ToP(F2))
            < Pb(ToP(F2)) * PyB(ToP(F1)))

def GtP(F1, F2):
    return (Pb(ToP(F1)) * PyB(ToP(F2))
            > Pb(ToP(F2)) * PyB(ToP(F1)))

# APLIKASI DALAM PYTHON
print(SubP(MkPC(1, 1, 2), MkPC(2, 1, 2)))
print(EqP(MkPC(2, 3, 4), MkPC(2, 3, 4)))
```

```python
print(DivP(MkPC(2, 1, 2), MkPC(2, 1, 2)))
print(MulP(MkPC(2, 1, 2), MkPC(2, 1, 2)))

# NO 2 TIPE BENTUKAN GARIS
# ===============================================
# ===============================================

# DEFINISI DAN SPESIFIKASI TYPE
# tipe point : <x:real, y:real>
#   {<x,y> adalah sebuah point dengan x sebagai absis dan y sebagai ordinat.}
# tipe line : <P1:point, P2:point>
#   {<P1,P2> adalah sebuah garis dengan 2 point.}
#
# DEFINISI DAN SPESIFIKASI SELEKTOR
# Abs : point --> real
#   {Abs(P) memberikan nilai absis x dari point P, <x,y>.}
# Ord : point --> real
#   {Ord(P) memberikan nilai ordinat y dari point P, <x,y>.}
# Pt1 : line --> point
#   {Pt1(L) memberikan point P1 dari garis L, <P1,P2>.}
# Pt2 : line --> point
#   {Pt2(L) memberikan point P2 dari garis L, <P1,P2>.}
#
# DEFINISI DAN SPESIFIKASI KONSTRUKTOR
# MkPnt : 2 real --> point
#   {MkPnt(x,y) membentuk sebuah point.}
# MkLine : 2 point --> line
#   {MkLine(P1,P2) membentuk sebuah garis.}
#
# DEFINISI DAN SPESIFIKASI FUNGSI/OPERATOR TERHADAP GARIS
# Grad : line --> real
#   {Grad(L) menghitung gradien dari sebuah garis.}
# LenLine : line --> real
#   {LenLine(L) menghitung panjang sebuah garis.}
#
# DEFINISI DAN SPESIFIKASI PREDIKAT
# IsPar? : 2 line --> boolean
#   {IsPar?(L1,L2) bernilai benar jika kedua gradien bernilai sama.}
# IsPerp? : 2 line --> boolean
#   {IsPerp?(L1,L2) bernilai benar jika perkalian kedua gradien sama dengan -1.}
#
# REALISASI DALAM PYTHON

def MkPnt(x, y):
    return [x, y]

def Abs(P):
    return P[0]

```

```python
def Ord(P):
    return P[1]

def MkLine(P1, P2):
    return [P1, P2]

def Pt1(L):
    return L[0]

def Pt2(L):
    return L[1]

def Grad(L):
    return (Ord(Pt1(L)) - Ord(Pt2(L))) / (Abs(Pt1(L)) - Abs(Pt2(L)))

def LenLine(P1, P2):
    return ((Abs(P1) - Abs(P2)) ** 2 + (Ord(P1) - Ord(P2)) ** 2) ** 0.5

def IsPar(L1, L2):
    return Grad(L1) == Grad(L2)

def IsPerp(L1, L2):
    return Grad(L1) * Grad(L2) == -1

# APLIKASI DALAM PYTHON

print(Grad(MkLine(MkPnt(1, 2), MkPnt(2, 1))))

print(IsPar(MkLine(MkPnt(1, 3), MkPnt(3, 2)),
            MkLine(MkPnt(1, 2), MkPnt(2, 3))))

print(IsPerp(MkLine(MkPnt(4, 2), MkPnt(5, 3)),
            MkLine(MkPnt(2, 5), MkPnt(1, -10))))
```