

## Tugas Type Bentukan.py

```
1 # Nama: Muhammad Farhan Abdul Azis
2 # Kelas: D
3 # NIM: 24060124140166
4
5 # NO 1 TIPE BENTUKAN PECAHAN CAMPURAN
6 # =====
7 # =====
8
9 # DEFINISI DAN SPESIFIKASI TYPE
10 # tipe pCamp : <bil: integer, n:integer >= 0, d:integer > 0 >
11 # {<bil,n,d> adalah sebuah pecahan campuran dengan bil dapat bernilai positif, negatif,
    maupun nol,
12 #      n selalu bernilai lebih kecil dari d, dengan d lebih dari nol.}
13 # type pBiasa : <n:integer, d: integer > 0>
14 # {<n,d> adalah pecahan biasa dengan n bisa bernilai negatif, nol, maupun positif, dan d
    lebih besar dari nol.}
15 #
16 # DEFINISI DAN SPESIFIKASI SELEKTOR
17 # B : pCamp --> integer
18 # {B(F) memberikan bilangan bil dari pecahan campuran F, <bil,n,d>.}
19 # Pc : pCamp --> integer >= 0
20 # {Pc(F) memberikan pembilang n dari pecahan campuran F, <bil,n,d>.}
21 # Py : pCamp --> integer > 0
22 # {Py(F) memberikan penyebut d dari pecahan campuran F, <bil,n,d>.}
23 # Pb : pBiasa --> integer
24 # {Pb(FR) memberikan pembilang n dari pecahan FR, <n,d>.}
25 # PyB : pBiasa --> integer > 0
26 # {PyB(FR) memberikan penyebut d dari pecahan FR, <n,d>.}
27 #
28 # DEFINISI DAN SPESIFIKASI KONSTRUKTOR
29 # MkPC : integer, integer >= 0, integer > 0 --> pCamp
30 # {MkPC(bil,n,d) membentuk pecahan campuran, <bil,n,d>.}
31 # MkP : integer, integer > 0 --> pBiasa
32 # {MkP(n,d) membentuk pecahan, <n,d>.}
33 #
34 # DEFINISI DAN SPESIFIKASI FUNGSI/OPERATOR TERHADAP PECAHAN CAMPURAN
35 # ToP : pCamp --> pBiasa
36 # {ToP(F) mengonversi sebuah pecahan campuran menjadi pecahan.}
37 # ToR : pCamp --> real
38 # {ToR(F) mengonversi sebuah pecahan campuran menjadi bilangan real.}
39 # AddP : 2 pCamp --> pCamp
40 # {AddP(F1,F2) menambahkan pecahan campuran F1 dengan pecahan campuran F2.}
41 # SubP : 2 pCamp --> pCamp
42 # {SubP(F1,F2) mengurangi pecahan campuran F1 dengan pecahan campuran F2.}
43 # DivP : 2 pCamp --> pCamp
44 # {DivP(F1,F2) membagi pecahan campuran F1 dengan pecahan campuran F2.}
45 # MulP : 2 pCamp --> pCamp
```

```
46 # {MulP(F1,F2) mengalikan pecahan campuran F1 dengan pecahan campuran F2}
47 #
48 # DEFINISI DAN SPESIFIKASI PREDIKAT
49 # EqP? : 2 pCamp --> boolean
50 # {EqP?(F1,F2) bernilai benar jika pecahan campuran F1 sama dengan pecahan campuran F2.}
51 # LtP? : 2 pCamp --> boolean
52 # {LtP?(F1,F2) bernilai benar jika pecahan campuran F1 kurang dari pecahan campuran F2.}
53 # GtP? : 2 pCamp --> boolean
54 # {GtP?(F1,F2) bernilai benar jika pecahan campuran F1 lebih dari pecahan campuran F2.}
55 #
56 # REALISASI DALAM PYTHON
57
58 def B(F):
59     return F[0]
60
61 def Pc(F):
62     return F[1]
63
64 def Py(F):
65     return F[2]
66
67 def Pb(FR):
68     return FR[0]
69
70 def PyB(FR):
71     return FR[1]
72
73 def MkPC(bil, n, d):
74     return [bil, n, d]
75
76 def MkP(n, d):
77     return [n, d]
78
79 def ToP(F):
80     if B(F) < 0:
81         return MkP(
82             B(F) * Py(F) - Pc(F),
83             Py(F)
84         )
85     else:
86         return MkP(
87             B(F) * Py(F) + Pc(F),
88             Py(F)
89         )
90
91 def ToR(F):
92     if B(F) < 0:
93         return B(F) - Pc(F) / Py(F)
94     else:
```

```
95         return B(F) + Pc(F) / Py(F)
96
97 def AddP(F1, F2):
98     return MkPC(
99         B(F1) + B(F2),
100         Pc(F1) * Py(F2) + Pc(F2) * Py(F1),
101         Py(F1) * Py(F2)
102     )
103
104 def SubP(F1, F2):
105     return MkPC(
106         B(F1) - B(F2),
107         Pc(F1) * Py(F2) - Pc(F2) * Py(F1),
108         Py(F1) * Py(F2)
109     )
110
111 def DivP(F1, F2):
112     return MkPC(
113         Pb(ToP(F1)) * PyB(ToP(F2)) //
114         (PyB(ToP(F1)) * Pb(ToP(F2))),
115         Pb(ToP(F1)) * PyB(ToP(F2)) %
116         (PyB(ToP(F1)) * Pb(ToP(F2))),
117         PyB(ToP(F1)) * Pb(ToP(F2))
118     )
119
120 def MulP(F1, F2):
121     return MkPC(
122         Pb(ToP(F1)) * Pb(ToP(F2)) //
123         (PyB(ToP(F1)) * PyB(ToP(F2))),
124         Pb(ToP(F1)) * Pb(ToP(F2)) %
125         (PyB(ToP(F1)) * PyB(ToP(F2))),
126         PyB(ToP(F1)) * PyB(ToP(F2))
127     )
128
129 def EqP(F1, F2):
130     return (Pb(ToP(F1)) * PyB(ToP(F2))
131            == Pb(ToP(F2)) * PyB(ToP(F1)))
132
133 def LtP(F1, F2):
134     return (Pb(ToP(F1)) * PyB(ToP(F2))
135            < Pb(ToP(F2)) * PyB(ToP(F1)))
136
137 def GtP(F1, F2):
138     return (Pb(ToP(F1)) * PyB(ToP(F2))
139            > Pb(ToP(F2)) * PyB(ToP(F1)))
140
141 # APLIKASI DALAM PYTHON
142 print(SubP(MkPC(1, 1, 2), MkPC(2, 1, 2)))
143 print(EqP(MkPC(2, 3, 4), MkPC(2, 3, 4)))
```

```
144 print(DivP(MkPC(2, 1, 2), MkPC(2, 1, 2)))
145 print(MulP(MkPC(2, 1, 2), MkPC(2, 1, 2)))
146
147 # NO 2 TIPE BENTUKAN GARIS
148 # =====
149 # =====
150
151 # DEFINISI DAN SPESIFIKASI TYPE
152 # tipe point : <x:real, y:real>
153 #   {<x,y> adalah sebuah point dengan x sebagai absis dan y sebagai ordinat.}
154 # tipe line : <P1:point, P2:point>
155 #   {<P1,P2> adalah sebuah garis dengan 2 point.}
156 #
157 # DEFINISI DAN SPESIFIKASI SELEKTOR
158 # Abs : point --> real
159 #   {Abs(P) memberikan nilai absis x dari point P, <x,y>}.}
160 # Ord : point --> real
161 #   {Ord(P) memberikan nilai ordinat y dari point P, <x,y>}.}
162 # Pt1 : line --> point
163 #   {Pt1(L) memberikan point P1 dari garis L, <P1,P2>}.}
164 # Pt2 : line --> point
165 #   {Pt2(L) memberikan point P2 dari garis L, <P1,P2>}.}
166 #
167 # DEFINISI DAN SPESIFIKASI KONSTRUKTOR
168 # MkPnt : 2 real --> point
169 #   {MkPnt(x,y) membentuk sebuah point.}
170 # MkLine : 2 point --> line
171 #   {MkLine(P1,P2) membentuk sebuah garis.}
172 #
173 # DEFINISI DAN SPESIFIKASI FUNGSI/OPERATOR TERHADAP GARIS
174 # Grad : line --> real
175 #   {Grad(L) menghitung gradien dari sebuah garis.}
176 # LenLine : line --> real
177 #   {LenLine(L) menghitung panjang sebuah garis.}
178 #
179 # DEFINISI DAN SPESIFIKASI PREDIKAT
180 # IsPar? : 2 line --> boolean
181 #   {IsPar?(L1,L2) bernilai benar jika kedua gradien bernilai sama.}
182 # IsPerp? : 2 line --> boolean
183 #   {IsPerp?(L1,L2) bernilai benar jika perkalian kedua gradien sama dengan -1.}
184 #
185 # REALISASI DALAM PYTHON
186
187 def MkPnt(x, y):
188     return [x, y]
189
190 def Abs(P):
191     return P[0]
192
```

```
193 def Ord(P):
194     return P[1]
195
196 def MkLine(P1, P2):
197     return [P1, P2]
198
199 def Pt1(L):
200     return L[0]
201
202 def Pt2(L):
203     return L[1]
204
205 def Grad(L):
206     return (Ord(Pt1(L)) - Ord(Pt2(L))) / (Abs(Pt1(L)) - Abs(Pt2(L)))
207
208 def LenLine(P1, P2):
209     return ((Abs(P1) - Abs(P2)) ** 2 + (Ord(P1) - Ord(P2)) ** 2) ** 0.5
210
211 def IsPar(L1, L2):
212     return Grad(L1) == Grad(L2)
213
214 def IsPerp(L1, L2):
215     return Grad(L1) * Grad(L2) == -1
216
217 # APLIKASI DALAM PYTHON
218
219 print(Grad(MkLine(MkPnt(1, 2), MkPnt(2, 1))))
220
221 print(IsPar(MkLine(MkPnt(1, 3), MkPnt(3, 2)),
222             MkLine(MkPnt(1, 2), MkPnt(2, 3))))
223
224 print(IsPerp(MkLine(MkPnt(4, 2), MkPnt(5, 3)),
225              MkLine(MkPnt(2, 5), MkPnt(1, -10))))
```