#El metodo SAREM es un ajuste de curva a la correlacion de Standing para determinar el factor de compresibilidad "z"

print("CORRELACION NUMERICA - METODO DE SAREM A.M.")

print("Ingrese los datos de presion en [lpca] y temperatura [R]")

Presion = float(input("Presion:"))

Temperatura = float(input("Temperatura:"))

#Se crea la superclase "Componentes" los datos se ingresan segun los componentes con los que se cuenta

class Componentes:

    print("Ingrese la fraccion molar de los componentes")

    #Se crea atributos de clase

    H2S = float(input("H2S: "))

    CO2 = float(input("CO2: "))

    N2 = float(input("N2: "))

    C1 = float(input("C1: "))

    C2 = float(input("C2: "))

    C3 = float(input("C3: "))

    i\_C4 = float(input("i-C4: "))

    n\_C4 = float(input("n-C4: "))

    i\_C5 = float(input("i-C5: "))

    n\_C5 = float(input("n-C5: "))

    C6 = float(input("C6: "))

    C7plus = float(input("C7+: "))

    # Se crea el inicializador

    def \_\_init\_\_(self,pH2S,pCO2,pN2,pC1,pC2,pC3,pi\_C4,pn\_C4,pi\_C5,pn\_C5,pC6,pC7plus,tH2S,tCO2,tN2,tC1,tC2,tC3,ti\_C4,tn\_C4,ti\_C5,tn\_C5,tC6,tC7plus):

        self.pH2S = pH2S            #Se crean los atributos de instancia

        self.pCO2 = pCO2

        self.pN2 = pN2

        self.pC1 = pC1

        self.pC2 = pC2

        self.pC3 = pC3

        self.pi\_C4 = pi\_C4

        self.pn\_C4 = pn\_C4

        self.pi\_C5 = pi\_C5

        self.pn\_C5 = pn\_C5

        self.pC6 = pC6

        self.pC7plus = pC7plus

        self.tH2S = tH2S

        self.tCO2 = tCO2

        self.tN2 = tN2

        self.tC1 = tC1

        self.tC2 = tC2

        self.tC3 = tC3

        self.ti\_C4 = ti\_C4

        self.tn\_C4 = tn\_C4

        self.ti\_C5 = ti\_C5

        self.tn\_C5 = tn\_C5

        self.tC6 = tC6

        self.tC7plus = tC7plus

    #Se definen los metodos a usar para la suma de presiones y temperaturas segun los componentes ingresados

    def suma\_presion(self):

        return self.pH2S\*Componentes.H2S + self.pCO2\*Componentes.CO2 + self.pN2\*Componentes.N2 + self.pC1\*Componentes.C1 + self.pC2\*Componentes.C2 + self.pC3\*Componentes.C3 + self.pi\_C4\*Componentes.i\_C4 + self.pn\_C4\*Componentes.n\_C4 + self.pi\_C5\*Componentes.i\_C5 + self.pn\_C5\*Componentes.n\_C5 + self.pC6\*Componentes.C6 + self.pC7plus\*Componentes.C7plus

    def suma\_temperatura(self):

        return self.tH2S\*Componentes.H2S + self.tCO2\*Componentes.CO2 + self.tN2\*Componentes.N2 + self.tC1\*Componentes.C1 + self.tC2\*Componentes.C2 + self.tC3\*Componentes.C3 + self.ti\_C4\*Componentes.i\_C4 + self.tn\_C4\*Componentes.n\_C4 + self.ti\_C5\*Componentes.i\_C5 + self.tn\_C5\*Componentes.n\_C5 + self.tC6\*Componentes.C6 + self.tC7plus\*Componentes.C7plus

    print("Fraccion Molar Promedio:", H2S + CO2 + N2 + C1 + C2 + C3 + i\_C4 + n\_C4 + i\_C5 + n\_C5 + C6 + C7plus)

#Se crean los objetos con argumentos segun el numero de atributos de instancia

suma\_pres = Componentes(1306,1071,493,667.8,707.8,616.3,529.1,550.7,490.4,488.6,436.9,370.3,0,0,0,0,0,0,0,0,0,0,0,0)

suma\_temp = Componentes(0,0,0,0,0,0,0,0,0,0,0,0,672.7,547.9,227.6,343.37,550.09,666.01,734.98,765.65,829.1,845.7,913.7,1144.23)

#Se crea una operacion que esta en base a los atributos de clase en caso de que se ingresen los datos de H2S y CO2

epsilon = 120\*(((Componentes.H2S + Componentes.CO2)\*\*0.9) - ((Componentes.H2S + Componentes.CO2)\*\*1.6)) + 15\*(((Componentes.H2S)\*\*0.5) + ((Componentes.H2S)\*\*4))

#La condicional if se usara "or"segun la logica booleana

#Si existe fraccion molar de los contaminantes H2S y CO2 se imprimira if, de lo contrario else

#La existencia de H2S y CO2 se la debe corregir en los calculos por Wichert y Aziz

if Componentes.H2S != 0 or Componentes.CO2 != 0:

    temp\_corr = suma\_temp.suma\_temperatura()-epsilon

    presion\_corr = (suma\_pres.suma\_presion()\*(suma\_temp.suma\_temperatura()-epsilon ))/(suma\_temp.suma\_temperatura()+(Componentes.H2S\*(1-Componentes.H2S)\*epsilon))

    print("Temperatura pseudo critica corregida:", temp\_corr )

    print("Presion pseudo critica corregida:", presion\_corr )

else:

    temp\_corr = suma\_temp.suma\_temperatura()

    presion\_corr = suma\_pres.suma\_presion()

    print("Temperatura pseudo critica:", temp\_corr )

    print("Presion pseudo critica:", presion\_corr )

#Se usaran los datos impresos de if para determinar la presion reducida y temperatura reducida

Psr = Presion/presion\_corr

print("Presion reducida:",Psr)

Tsr = Temperatura/temp\_corr

print("Temperatura reducida:",Tsr)

#Se determina los coeficientes "x" y "y" en base a la presion reducida y temperatura reducida

x=(2\*Psr-15)/14.8

print("Coeficiente X:",x)

y = (2\*Tsr-4)/1.9

print("Coeficiente Y:",y)

#Los coeficientes de "x" y "y" se usaran para determinar el factor de compresibilidad "z"

#Que se basa en los polinomios de Legendre de grado 0 a 5

#Se crea una nueva clase para el desarrollo de este calculo final

class Operacion:

    def \_\_init\_\_(self,x,y):

        self.x = x

        self.y = y

    def P\_0X(self):

        return 0.7071068\*(self.x/self.x)

    def P\_1X(self):

        return 1.224745\*self.x

    def P\_2X(self):

        return 0.7905695\*((3\*(self.x)\*\*2)-1)

    def P\_3X(self):

        return 0.9354145\*((5\*(self.x)\*\*3)-(3\*self.x))

    def P\_4X(self):

        return 0.265165\*((35\*(self.x)\*\*4)-(30\*(self.x)\*\*2)+3)

    def P\_5X(self):

        return 0.293151\*((63\*(self.x)\*\*5)-(70\*(self.x)\*\*3)+(15\*self.x))

    def P\_0Y(self):

        return 0.7071068\*(self.y/self.y)

    def P\_1Y(self):

        return 1.224745\*self.y

    def P\_2Y(self):

        return 0.7905695\*((3\*(self.y)\*\*2)-1)

    def P\_3Y(self):

        return 0.9354145\*((5\*(self.y)\*\*3)-(3\*self.y))

    def P\_4Y(self):

        return 0.265165\*((35\*(self.y)\*\*4)-(30\*(self.y)\*\*2)+3)

    def P\_5Y(self):

        return 0.293151\*((63\*(self.y)\*\*5)-(70\*(self.y)\*\*3)+(15\*self.y))

A = Operacion(x,0)

B = Operacion(0,y)

#Constantes definidads por el metodo SAREM

A00,A01,A02,A03,A04,A05 = 2.1433504,0.0831762,-0.0214670,-0.0008714,0.0042846,-0.0016595

A10,A11,A12,A13,A14,A15 = 0.3312352,-0.1340361,0.0668810,-0.0271743,0.0088512,-0.0021521

A20,A21,A22,A23,A24,A25 = 0.1057287,-0.0503937,0.0050925,0.0105513,-0.0073182,0.0026960

A30,A31,A32,A33,A34,A35 = -0.0521840,0.0443121,-0.0193294,0.0058973,0.0015367,-0.0028327

A40,A41,A42,A43,A44,A45 = 0.0197040,-0.0263834,0.0192621,-0.0115354,0.0042910,-0.0081303

A50,A51,A52,A53,A54,A55 = 0.0053096,0.0089178,-0.0108948,0.0095594,-0.0060114,0.0031175

#Se multiplica los valores obtenidos de las ecuaciones de Legendre

Z0 = A.P\_0X()\*(A00\*B.P\_0Y() + A01\*B.P\_1Y() + A02\*B.P\_2Y() + A03\*B.P\_3Y() + A04\*B.P\_4Y() + A05\*B.P\_5Y())

Z1 = A.P\_1X()\*(A10\*B.P\_0Y() + A11\*B.P\_1Y() + A12\*B.P\_2Y() + A13\*B.P\_3Y() + A14\*B.P\_4Y() + A15\*B.P\_5Y())

Z2 = A.P\_2X()\*(A20\*B.P\_0Y() + A21\*B.P\_1Y() + A22\*B.P\_2Y() + A23\*B.P\_3Y() + A24\*B.P\_4Y() + A25\*B.P\_5Y())

Z3 = A.P\_3X()\*(A30\*B.P\_0Y() + A31\*B.P\_1Y() + A32\*B.P\_2Y() + A33\*B.P\_3Y() + A34\*B.P\_4Y() + A35\*B.P\_5Y())

Z4 = A.P\_4X()\*(A40\*B.P\_0Y() + A41\*B.P\_1Y() + A42\*B.P\_2Y() + A43\*B.P\_3Y() + A44\*B.P\_4Y() + A45\*B.P\_5Y())

Z5 = A.P\_5X()\*(A50\*B.P\_0Y() + A51\*B.P\_1Y() + A52\*B.P\_2Y() + A53\*B.P\_3Y() + A54\*B.P\_4Y() + A55\*B.P\_5Y())

#Se las divide en 6 filas para evitar confusion

Z = Z0 + Z1 + Z2 + Z3 + Z4 + Z5

#Finalmente la suma seria el factor de compresinilidad

print("FACTOR DE COMPRESIBILIDAD =",Z)