Type	Supply Temperature (°C)	Target Temperature (°C)	Flow Rate (kg/s)	Specific Heat (kJ/kg.°C)	Heat Load (kW)	Heat capacity flow rate (kW/°C)
H	240	90	3	2		
H	130	100	6		540	
H	170	40		4	1040	
C	40		5	3	1800	
C		120		2	1120	16

En este ejercicio utilizaremos la notación compacta general para cualquier número de intervalo y corrientes calientes y frías. Para ello, tendremos que leer de dos archivos Excel los aportes de las corrientes calientes y la eliminación de las frías. Estos datos, en forma de tabla, están en el archivo "Ejercicio3.xlsx", en las hojas "Hot" y "Cold", respectivamente.

En este caso, tendremos que usar la librería Pandas, con la que importaremos los datos para, posteriormente usarlos dentro del modelo de optimización.

In [ ]:
```
from pyomo.environ import *
import pandas as pd

model = ConcreteModel()
```

Leemos los datos de las corrientes calientes

In [ ]:
```
Hot_df = pd.read_excel('Datos/Ejercicio3.xlsx', 'Hot',skiprows = 1,index_col=0)
Hot_df
```

Out[ ]:

ID	0	1	2	3	4	5	6	7	8	9
H1	0	360	60	180	60	180	60	0	0	0
H2	0	0	0	0	0	540	0	0	0	0
H3	0	0	0	240	80	240	80	160	80	160

E igualmente de las frías.

In [ ]:
```
Cold_df = pd.read_excel('Datos/Ejercicio3.xlsx', 'NewCold3',skiprows = 1, index_col=
Cold_df
```

Out[ ]:

ID	0	1	2	3	4	5	6	7	8	9
C1	0	0	150	450	150	450	150	300	150	0
C2	0	0	0	0	160	480	160	320	0	0
C3	0	0	0	0	0	0	0	400	200	400

Las columnas de las tablas anteriores son los diferentes intervalos de temperatura (9+1 ya que incluimos un intervalo 0 que necesitamos porque en el balance aparece el término R(i-1)), mientras que las filas son las corrientes calientes y frías respectivamente. Con estos datos

crearemos tres índices para el modelo: uno (h) contendrá los identificadores de las corrientes calientes (H1, H2 y H3), otro (c) de las frías (C1 y C2) y un último (i) con los diez intervalos (0-9)

In [ ]:
```
hotst = Hot_df.index.values.tolist()
coldst = Cold_df.index.values.tolist()
model.h=Set(initialize=hotst)
model.c=Set(initialize=coldst)
model.i=Set(initialize=Hot_df.columns)
```

Podemos ver lo que hemos creado hasta ahora haciendo un model.pprint()

In [ ]:
```
model.pprint()
```

```
3 Set Declarations
    c : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    3 : {'C1', 'C2', 'C3'}
    h : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    3 : {'H1', 'H2', 'H3'}
    i : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :   10 : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

3 Declarations: h c i
```

Creamos ahora las variables del modelo. Hay i potenciales aportes de calor por utilities calientes (Qs), uno en cada intervalo, al igual que de frías (Qw). De esta forma damos la oportunidad de que entren las utilities en cualquier nivel de temperatura. Creamos también i residuos (R), aunque sabemos que no habrá residuo del intervalo 0 ni tampoco del intervalo 9, lo cual implementaremos como dos restricciones diferentes.

In [ ]:
```
Qs = model.Qs = Var(model.i,within = NonNegativeReals)
Qw = model.Qc = Var(model.i,within = NonNegativeReals)
R = model.R = Var(model.i,within = NonNegativeReals)
```

Buscamos la minimización del vapor del intervalo 1 más la utility fría del intervalo 9.

In [ ]:
```
model.util = Objective(expr = model.Qs[1] + model.Qc[9])
```

Constraints

In [ ]:
```
ni = list(model.i)[1:] #Creamos una lista que contenga los índices sobre los que har
model.int = ConstraintList()
for i in ni:
    model.int.add(
        R[i-1]+Qs[i]+sum(Hot_df[i]) == R[i]+sum(Cold_df[i])+Qw[i]
    )

model.R0 = Constraint(expr = R[0] == 0) #No hay residuo ni desde intervalo 0 ni desd
model.R9 = Constraint(expr = R[9] == 0)

nii = list(model.i)[0:1]+list(model.i)[2:] #Con esta restricción buscamos que solo s
model.steam = ConstraintList()
for i in nii:
    model.steam.add(
        Qs[i]==0
```

```
    )

niii = list(model.i)[0:9] #Lo mismo deantes para la utility fría. Es decir de 0 a 8
model.cw = ConstraintList()
for i in niii:
    model.cw.add(
        Qw[i]==0
    )
```

Resolvemos el modelo

In [ ]:

```
results = SolverFactory('glpk').solve(model)
model.pprint()
results.write()
```

```
6 Set Declarations
    c : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    3 : {'C1', 'C2', 'C3'}
    cw_index : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    9 : {1, 2, 3, 4, 5, 6, 7, 8, 9}
    h : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    3 : {'H1', 'H2', 'H3'}
    i : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :   10 : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
    int_index : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    9 : {1, 2, 3, 4, 5, 6, 7, 8, 9}
    steam_index : Size=1, Index=None, Ordered=Insertion
        Key  : Dimen : Domain : Size : Members
        None :     1 :    Any :    9 : {1, 2, 3, 4, 5, 6, 7, 8, 9}

3 Var Declarations
    Qc : Size=10, Index=i
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          0 :     0 :   0.0 :  None : False : False : NonNegativeReals
          1 :     0 :   0.0 :  None : False : False : NonNegativeReals
          2 :     0 :   0.0 :  None : False : False : NonNegativeReals
          3 :     0 :   0.0 :  None : False : False : NonNegativeReals
          4 :     0 :   0.0 :  None : False : False : NonNegativeReals
          5 :     0 :   0.0 :  None : False : False : NonNegativeReals
          6 :     0 :   0.0 :  None : False : False : NonNegativeReals
          7 :     0 :   0.0 :  None : False : False : NonNegativeReals
          8 :     0 :   0.0 :  None : False : False : NonNegativeReals
          9 :     0 :   0.0 :  None : False : False : NonNegativeReals
    Qs : Size=10, Index=i
        Key : Lower : Value  : Upper : Fixed : Stale : Domain
          0 :     0 :    0.0 :  None : False : False : NonNegativeReals
          1 :     0 : 1440.0 :  None : False : False : NonNegativeReals
          2 :     0 :    0.0 :  None : False : False : NonNegativeReals
          3 :     0 :    0.0 :  None : False : False : NonNegativeReals
          4 :     0 :    0.0 :  None : False : False : NonNegativeReals
          5 :     0 :    0.0 :  None : False : False : NonNegativeReals
          6 :     0 :    0.0 :  None : False : False : NonNegativeReals
          7 :     0 :    0.0 :  None : False : False : NonNegativeReals
          8 :     0 :    0.0 :  None : False : False : NonNegativeReals
          9 :     0 :    0.0 :  None : False : False : NonNegativeReals
    R : Size=10, Index=i
        Key : Lower : Value  : Upper : Fixed : Stale : Domain
          0 :     0 :    0.0 :  None : False : False : NonNegativeReals
```

```
          1 :     0 : 1800.0 :  None : False : False : NonNegativeReals
          2 :     0 : 1710.0 :  None : False : False : NonNegativeReals
          3 :     0 : 1680.0 :  None : False : False : NonNegativeReals
          4 :     0 : 1510.0 :  None : False : False : NonNegativeReals
          5 :     0 : 1540.0 :  None : False : False : NonNegativeReals
          6 :     0 : 1370.0 :  None : False : False : NonNegativeReals
          7 :     0 :  510.0 :  None : False : False : NonNegativeReals
          8 :     0 :  240.0 :  None : False : False : NonNegativeReals
          9 :     0 :    0.0 :  None : False : False : NonNegativeReals

1 Objective Declarations
    util : Size=1, Index=None, Active=True
        Key  : Active : Sense    : Expression
        None :   True : minimize : Qs[1] + Qc[9]

5 Constraint Declarations
    R0 : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper : Active
        None :   0.0 : R[0] :   0.0 :   True
    R9 : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper : Active
        None :   0.0 : R[9] :   0.0 :   True
    cw : Size=9, Index=cw_index, Active=True
        Key : Lower : Body  : Upper : Active
          1 :   0.0 : Qc[0] :   0.0 :   True
          2 :   0.0 : Qc[1] :   0.0 :   True
          3 :   0.0 : Qc[2] :   0.0 :   True
          4 :   0.0 : Qc[3] :   0.0 :   True
          5 :   0.0 : Qc[4] :   0.0 :   True
          6 :   0.0 : Qc[5] :   0.0 :   True
          7 :   0.0 : Qc[6] :   0.0 :   True
          8 :   0.0 : Qc[7] :   0.0 :   True
          9 :   0.0 : Qc[8] :   0.0 :   True
    int : Size=9, Index=int_index, Active=True
        Key : Lower : Body                                          : Upper : Active
          1 :   0.0 :        R[0] + Qs[1] + 360 - (R[1] + Qc[1]) :   0.0 :   True
          2 :   0.0 :   R[1] + Qs[2] + 60 - (R[2] + 150 + Qc[2]) :   0.0 :   True
          3 :   0.0 :  R[2] + Qs[3] + 420 - (R[3] + 450 + Qc[3]) :   0.0 :   True
          4 :   0.0 :  R[3] + Qs[4] + 140 - (R[4] + 310 + Qc[4]) :   0.0 :   True
          5 :   0.0 :  R[4] + Qs[5] + 960 - (R[5] + 930 + Qc[5]) :   0.0 :   True
          6 :   0.0 :  R[5] + Qs[6] + 140 - (R[6] + 310 + Qc[6]) :   0.0 :   True
          7 :   0.0 : R[6] + Qs[7] + 160 - (R[7] + 1020 + Qc[7]) :   0.0 :   True
          8 :   0.0 :   R[7] + Qs[8] + 80 - (R[8] + 350 + Qc[8]) :   0.0 :   True
          9 :   0.0 :  R[8] + Qs[9] + 160 - (R[9] + 400 + Qc[9]) :   0.0 :   True
    steam : Size=9, Index=steam_index, Active=True
        Key : Lower : Body  : Upper : Active
          1 :   0.0 : Qs[0] :   0.0 :   True
          2 :   0.0 : Qs[2] :   0.0 :   True
          3 :   0.0 : Qs[3] :   0.0 :   True
          4 :   0.0 : Qs[4] :   0.0 :   True
          5 :   0.0 : Qs[5] :   0.0 :   True
          6 :   0.0 : Qs[6] :   0.0 :   True
          7 :   0.0 : Qs[7] :   0.0 :   True
          8 :   0.0 : Qs[8] :   0.0 :   True
          9 :   0.0 : Qs[9] :   0.0 :   True

15 Declarations: h c i Qs Qc R util int_index int R0 R9 steam_index steam cw_index c
w
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
```

```
Problem:
- Name: unknown
  Lower bound: 1440.0
  Upper bound: 1440.0
  Number of objectives: 1
  Number of constraints: 30
  Number of variables: 31
  Number of nonzeros: 57
  Sense: minimize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.08428764343261719
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

In [ ]:

```
Qh = value(model.Qs[1])
Qc = value(model.Qc[9])
print('Cold utility = {0:2.2f}, Hot utility = {1:2.2f}'.format(Qc, Qh))
```

```
Cold utility = 0.00, Hot utility = 1440.00
```