In [1]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

In [2]:
```
def Gauss_Seidel(A, B, threshold_error=0.1, float_decimals=2, verbose=True):
    '''
    This function solves A*x = B as A is a n*n matrix and B is n*1 matrix and find the x values with Gauss Seidel Algorithm.

    threshold_error:
        This parameter is the threshold of error and the algorithm iterates until the error rate gets less than threshold.

    float_decimal:
        Rounds the output to the value of this parameter.

    verbose:
        True or False
        if True prints the process log of the algorithm!

        Prepared by Dara Samii
        Data: April 2020
    '''
    #check input parameters be corect!
    if (A.shape[0] != A.shape[1]):
        raise "Matrix A must be a square Matrix!"

    elif (A.shape[0] != B.shape[0]):
        raise "A Matrix dimention must be same to B matrix dimention "

    elif (B.shape[1] != 1):
        raise "B matix shape must be like (n * 1)"

    # Matrix dimention
    n = A.shape[0]

    # X: output matrix
    X = [0.0 for i in range(n)]
    X = np.array(X,dtype=np.float64)

    #this functions calculate the error of the results
    def error_calc(current,previous):
        return (abs(current - previous) / current) * 100

    Iter = 0
    # iterates until the Error gets less than threshold
    Error = threshold_error
    while Error >= threshold_error:
        Iter += 1
        previous_X = X.copy()

        for i in range(n):
            SUMX = 0.0
            for j in range(n):
                if j != i:
                    SUMX += A[i][j] * X[j]
                else:
                    pass

            X[i] = (float(B[i][0]) - float(SUMX))/float(A[i][i])

        Error = round(error_calc(X, previous_X).max(), ndigits = float_decimals)

        if verbose == True:
            print("iteration number: ", Iter)
            print("X : ", X.round(decimals = float_decimals))
            print("error: ",Error)
    return X.round(decimals = float_decimals)
```

Question

In [3]:
```
im = Image.open('question.png')
display(im)
```

Consider the square of Figure Example 3-5. The left face is maintained at 100°C and the top face at 500°C, while the other two faces are exposed to an environment at 100°C:

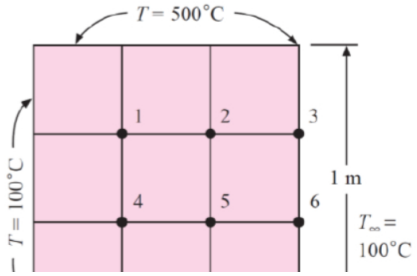
$$h = 10 \text{ W/m}^2 \cdot ^\circ\text{C} \quad \text{and} \quad k = 10 \text{ W/m} \cdot ^\circ\text{C}$$

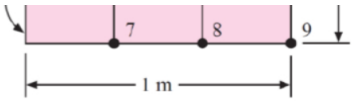
The block is 1 m square. Compute the temperature of the various nodes as indicated in Figure Example 3-5 and the heat flows at the boundaries.

Solution

In [4]:
```
im = Image.open('shape.png')
display(im)
```

Figure Example 3-5 | Nomenclature for Example 3-5.





$T_2 + T_4 - 4T_1 = -600$

$T_1 + T_3 + T_5 - 4T_2 = -500$

$T_1 + T_5 + T_7 - 4T_4 = -100$

$T_1 + T_2 + T_6 + T_8 - 4T_5 = 0$

$2T_2 + T_6 - 4.67T_3 = -567$

$2T_5 + T_3 + T_9 - 4.67T_6 = 0$

$2T_4 + T_8 - 4.67T_7 = -167$

$2T_5 + T_7 + T_9 - 4.67T_8 = -67$

$T_6 + T_8 - 2.67T_9 = -67$

In [5]:
```
A = [[-4, 1, 0,  1, 0, 0, 0, 0, 0],
     [ 1,-4, 1,  0, 1, 0, 0, 0, 0],
     [ 0, 2, -4.67,  0, 0, 1, 0, 0, 0],
     [ 1, 0, 0, -4, 1, 0, 1, 0, 0],
     [ 0, 1, 0,  1, -4, 1, 0, 1, 0],
     [ 0, 0, 1,  0, 2, -4.67, 0, 0, 1],
     [ 0, 0, 0,  2, 0, 0, -4.67, 1, 0],
     [ 0, 0, 0,  0, 2, 0, 1, -4.67, 1],
     [ 0, 0, 0,  0, 0, 1, 0, 1, -2.67]]

for i in range(9):
    for j in range(9):
        A[i][j] = float(A[i][j])

A = np.array(A,dtype=np.float64)
print("A:\n",A)
```

```
A:
 [[-4.    1.    0.    1.    0.    0.    0.    0.    0.  ]
 [ 1.   -4.    1.    0.    1.    0.    0.    0.    0.  ]
 [ 0.    2.   -4.67  0.    0.    1.    0.    0.    0.  ]
 [ 1.    0.    0.   -4.    1.    0.    1.    0.    0.  ]
 [ 0.    1.    0.    1.   -4.    1.    0.    1.    0.  ]
 [ 0.    0.    1.    0.    2.   -4.67  0.    0.    1.  ]
 [ 0.    0.    0.    2.    0.    0.   -4.67  1.    0.  ]
 [ 0.    0.    0.    0.    2.    0.    1.   -4.67  1.  ]
 [ 0.    0.    0.    0.    0.    1.    0.    1.   -2.67]]
```

In [6]:
```
B = [-600.0, -500.0, -567.0, -100.0, 0.0, -67.0, -167.0, -67.0, -67.0]
B = np.array(B,dtype=np.float64)
B = B.reshape(9,1)
print("B:\n",B)
```

```
B:
 [[-600.]
 [-500.]
 [-567.]
 [-100.]
 [   0.]
 [ -67.]
 [-167.]
 [ -67.]
 [ -67.]]
```

In [7]:
```
T = Gauss_Seidel(A, B, threshold_error=0.0001,float_decimals=2)
```

```
iteration number:  1
X :  [150.   162.5  191.01  62.5   56.25  79.34  62.53  51.83  74.22]
error:  100.0
iteration number:  2
X :  [206.25 238.38 240.49 106.26 118.95 132.68  92.36 100.96 112.6 ]
error:  52.71
iteration number:  3
X :  [236.16 273.9  267.13 136.87 161.1  164.65 115.99 132.29 136.31]
error:  26.16
iteration number:  4
X :  [252.69 295.23 283.11 157.45 187.4  184.42 131.52 151.96 151.08]
error:  14.04
iteration number:  5
X :  [263.17 308.42 292.99 170.52 203.83 196.73 141.33 164.25 160.29]
error:  8.06
iteration number:  6
X :  [269.74 316.64 299.14 178.72 214.09 204.41 147.47 171.94 166.05]
error:  4.79
iteration number:  7
X :  [273.84 321.77 302.99 183.85 220.49 209.21 151.31 176.73 169.64]
error:  2.91
iteration number:  8
X :  [276.4  324.97 305.39 187.05 224.49 212.21 153.71 179.73 171.89]
error:  1.78
iteration number:  9
X :  [278.01 326.97 306.88 189.05 226.99 214.08 155.21 181.6  173.29]
error:  1.1
iteration number:  10
X :  [279.01 328.22 307.82 190.3  228.55 215.25 156.15 182.77 174.16]
error:  0.68
iteration number:  11
X :  [279.63 329.   308.4  191.08 229.53 215.98 156.73 183.5  174.71]
error:  0.42
iteration number:  12
X :  [280.02 329.49 308.77 191.57 230.13 216.43 157.1  183.96 175.05]
error:  0.26
iteration number:  13
X :  [280.26 329.79 309.   191.87 230.51 216.72 157.32 184.24 175.27]
error:  0.16
```

```
iteration number:  14
X :  [280.42 329.98 309.14 192.06 230.75 216.9  157.47 184.42 175.4 ]
error:  0.1
iteration number:  15
X :  [280.51 330.1  309.23 192.18 230.9  217.01 157.56 184.53 175.48]
error:  0.06
iteration number:  16
X :  [280.57 330.17 309.28 192.26 230.99 217.08 157.61 184.6  175.53]
error:  0.04
iteration number:  17
X :  [280.61 330.22 309.32 192.3  231.05 217.12 157.65 184.64 175.57]
error:  0.03
iteration number:  18
X :  [280.63 330.25 309.34 192.33 231.09 217.15 157.67 184.67 175.59]
error:  0.02
iteration number:  19
X :  [280.65 330.27 309.35 192.35 231.11 217.16 157.68 184.69 175.6 ]
error:  0.01
iteration number:  20
X :  [280.65 330.28 309.36 192.36 231.12 217.18 157.69 184.7  175.61]
error:  0.01
iteration number:  21
X :  [280.66 330.29 309.37 192.37 231.13 217.18 157.69 184.7  175.61]
error:  0.0
```

In [8]:
```
print('\n')
for i in range(9):
    print(f"T_{i+1} : {T[i]} C")
```

```
T_1 : 280.66 C
T_2 : 330.29 C
T_3 : 309.37 C
T_4 : 192.37 C
T_5 : 231.13 C
T_6 : 217.18 C
T_7 : 157.69 C
T_8 : 184.7 C
T_9 : 175.61 C
```

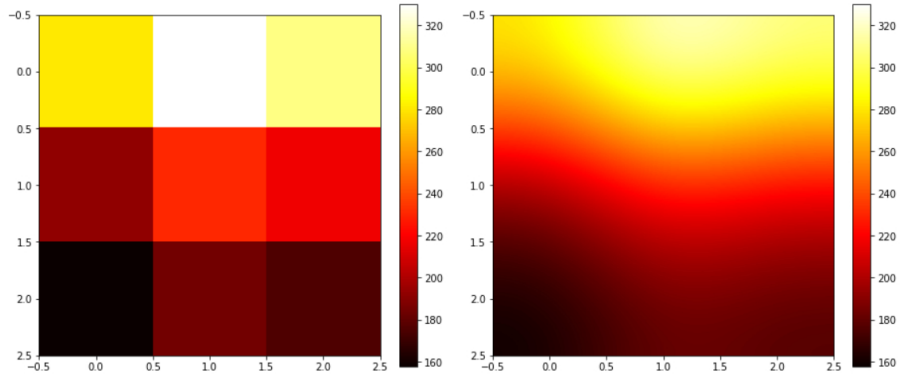
Heatmap of 2D heat transfer

In [9]:
```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (15, 8))

plt.subplot(1, 2, 1)
plt.imshow(T.reshape(3, 3), cmap = 'hot')
plt.colorbar(fraction=0.048)

plt.subplot(1,2,2)
plt.imshow(T.reshape(3,3),cmap='hot',interpolation='bicubic')
plt.colorbar(fraction=0.048)
```

Out[9]: <matplotlib.colorbar.Colorbar at 0xfa70d90>



In [ ]: