```
# Date: 28 October 2021
#
```

Program design:

1. Ask user for temperature input
2. Find where the user's temperature value is in relation to the temperatures on the excel sheet
    a. If the user's temperature is one of the given temperature from the excel sheet, return those four values
    b. Ex: if the temperature given is 20, return the other values on the table for 20 3. Determine the next highest and lowest temperature on the list of values provided a. Ex: if the temperature given is 25, the low temperature would be 20 and the high would be 40
4. Use linear interpolation to find the values for that temperature
    a. Linear interpolation: (x0 - x1)(slope) + y1
    b. Slope = (y2 - y1) / (x2 - x1)
5. Return the values of P, 5, MpA, and (263.94C) for that temperature

It would be best to use a dictionary for the values, because 1 temperature can be used for the index of the 4 different values. It would also make it easier to calculate the linear interpolation of the values because the same index can be used to find all 4 values by just changing which part of the dictionary it uses rather than getting all the values to perform linear interpolation 4 times.

Variables:

- tempProperties {}
    - Ex: tempProperties = {}
    - tempProperties [0] = [.0009977, .04, 5.03, 2.9207]
    - tempProperties [20] = [... etc]
- userTemp = input("Enter temperature value")
- tempMin
- tempMax
    - Ex: if userTemp = 25, tempMin = 20, tempMax = 40
- slope = tempProperties[tempMax][index] - tempProperties[tempMin][index] / (tempMax-tempMin)
- interpolate = (linear interpolation formula)
- userProperties = []
    - Add the properties found by linear interpolation to this list to save them for the final output
    -

Dictionary:

```
tempProperties = {
                0 : [0.0009977, 0.04, 5.03, 0.0001],
      range(1, 20) : [0.0009996, 83.61, 88.61, 0.2954],
        range(21, 40) : [0.0010057, 166.92, 171.95, 0.5705],
        range(41, 60) : [0.0010149, 250.29, 255.36, 0.8287],
        range(61, 80) : [0.0010267, 333.82, 338.96, 1.0723],
        range(81, 100) : [0.0010410, 417.65, 422.85, 1.3034],
        range(101, 120) : [0.0010576, 501.91, 507.19, 1.5236],
        range(121, 140) : [0.0010769, 586.8, 592.18, 1.7344],
        range(141, 160) : [0.0010988, 672.55, 678.04, 1.9374],
        range(161, 180) : [0.0011240, 759.47, 765.09, 2.1338],
        range(181, 200) : [0.0011531, 847.92, 853.68, 2.3251],
        range(201, 220) : [0.0011868, 938.39, 944.32, 2.5127],
        range(221, 240) : [0.0012268, 1031.6, 1037.7, 2.6983],
        range(241, 260) : [0.0012755, 1128.5, 1134.9, 2.8841]
        }


#gathers values from user and initializes variables
userTemp = input("Enter a temperature value: ")
tempMin = 0
tempMax = 0

#goes through the dictionary
#if userTemp is greater than the index before and less than the index after
#those values will be assigned
for i in range(tempProperties - 1):
    if tempProperties[i] > userTemp and tempProperties[i] < userTemp:
        tempMin = tempProperties[i]
        tempMax = tempProperties[i + 1]
#if userTemp is equal to one of the temperature indexes, the min and max values
#don't matter

def equation(A, B):
    volume = tempProperties[userTemp][0] + (userTemp - tempProperties[userTemp][4]) *
((tempProperties[userTemp][0] - tempProperties[userTemp][0]) / (tempProperties[userTemp][4] -
tempProperties[userTemp][4]))
    IntTemp = tempProperties[userTemp][1] + (userTemp - tempProperties[userTemp][4]) *
((tempProperties[userTemp][1] - tempProperties[B][1]) / (tempProperties[userTemp][4] -
tempProperties[userTemp][4]))
    Enthalpy = tempProperties[userTemp][2] + (userTemp - tempProperties[userTemp][4]) *
((tempProperties[userTemp][2] - tempProperties[userTemp][2]) / (tempProperties[userTemp][4] -
tempProperties[userTemp][4]))
    Entropy = tempProperties[userTemp][3] + (userTemp - tempProperties[userTemp][4]) *
((tempProperties[userTemp][3] - tempProperties[userTemp][3]) / (tempProperties[userTemp][4] -
tempProperties[userTemp][4]))
```

```
    print("Properties at {} deg C are:".format(userTemp))
    print("Specific volume (m^3/kg): {:.7f}".format(volume))
    print("Specific internal energy (kJ/kg): {:.2f}".format(IntTemp))
    print("Specific enthalpy (kJ/kg): {:.2f}".format(Enthalpy))
    print("Specific entropy (kJ/kgK): {:.4f}".format(Entropy))
```

• Describing the difficulty with which your team was able to combine the code at the end. Did this provide your team any insight into how the design itself might have been specified more clearly?

There was some difficulty when combining and debugging the code at the end. Small differences like having a variable be type integer instead of type flAs well, when debugging the code, if errors came up at the beginning of the code it would affect the rest of the code. In the future, things like specifying the type of the variable and how the different sections will be organized should be taken into consideration.

• Describing any benefits and drawbacks you saw into dividing the coding like this. Can you see reasons why this might be a good idea? Can you see rea bad idea?

Some benefits of dividing the coding like this is that it takes less time to put the full code together and get it running. It’s also less work for each person to do. Some drawbacks with dividing the coding like this is that some small mistakes in one section can lead the entire code to fall apart or not being able to divide the work equally. It’s also harder to see if your particular section will work without being able to run it with the rest of the code.

Test cases:
Temp = 0
Temp = 35
Temp = 25
Temp = 120
Temp = 124.5
Temp = 22
Temp = 260
Temp = 255
Temp = 88
Temp = 140