Tutorial 4 Total 10 marks

Question 1 (10 marks)

Consider a one dimensional, heat conducting bar with fixed temperatures at the left and right hand sides.

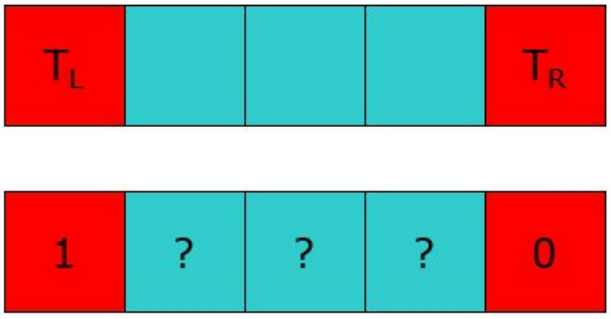
1. Prove that the steady one dimensional heat transfer equation, written as:

$$\frac{d}{dx}\left(\alpha\frac{dT}{dx}\right)=0$$

   can be written in the discretized form:

$$\alpha\left(\frac{T_{i+1}+T_{i-1}-2T_i}{\Delta x^2}\right)=0$$

2. Show that the equations above can be written in a matrix form Ax = B where x is a vector containing the solutions to the steady temperatures T.
3. For the simple problem with 3 DOF as shown below (figure 1), show that the matrix A and vector B is equal to those shown. Find the missing three temperatures through calculation by hand (or using MATLAB).
4. Write a C/C++ code on the remote server to use the **CG method** to compute the 3 temperatures for the problem shown in Figure 1. Show the result converges onto the result obtained via hand calculation. Save your code as Tutorial_4_cg.c in the directory created.



$$A=\begin{bmatrix}2&-1&0\\-1&2&-1\\0&-1&2\end{bmatrix}\qquad B=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

Figure 1 - Simple Problem and Matrix for Question 1. TL = 1, TR = 0.

HINTS AND TIPS

1) To write a CG solver, you'll need functions which can:
   - Multiply a matrix and a vector to produce a vector,
   - Multiply a vector by a constant to produce a vector,
   - Subtract and add vectors together to produce a vector, and
   - Compute the sum of a vector.

Create different functions for these things, and test each one while you are writing them.

2) You should avoid using 2D arrays! Instead of this....

```
// Our 2D matrix might look like this (BUT DON'T DO THIS)
// float A[5][5] = { {-2, 1, 0, 0, 0}, {1, -2, 1, 0, 0}, {0, 1, -2, 1, 0}, {0, 0, 1, -2, 1},  {0, 0, 0, 1, -2}};
```

..do this

```
// What I am going to do is unroll it like this - what is below is a 1D array holding data for a 2D array
float A[25] =  { -2, 1, 0, 0, 0,       1, -2, 1, 0, 0,       0, 1, -2, 1, 0,      0, 0, 1, -2, 1,     0, 0, 0, 1, -2};
```

3) Don't write all of the CG code at once! Write the functions in (1) (above) and test each one carefully.

4) Remember, we only create OpenMP threads once. For the CG method, we MUST create threads before we start iterating to get an answer.

When writing the functions above (in part (1)), don't forget – each function you write will be called by multiple OpenMP threads at the same time.

This should be taken into account NOW, even **before you use OpenMP**.

4) If you want to test your CG solver, you can use the Ax = B system shown in Figure 1, and you can use this:

```
float A[25] =  { -2, 1, 0, 0, 0,       1, -2, 1, 0, 0,       0, 1, -2, 1, 0,      0, 0, 1, -2, 1,     0, 0, 0, 1, -2};
float B[5] = {-1, 0, 0, 0, 0};
```

This is (basically) identical to the smaller problem shown in Figure 1.