In this assignment, you will create some simple programs in C to get used to the language (i.e., control structs, variables, rand(), arrays), the Linux environment, and the compiling/running process.

(1) Chocolate Bar Orders

Assume that you have a corner store and need to order some chocolate bars. Write a program called `chocolateBars.c` that displays the following menu:

1. Oh Henry $0.65
2. Coffee Crisp $0.80
3. Aero $0.60
4. Smarties $0.70
5. Crunchie $0.75

Your program should then ask the user for the # of boxes of each type of chocolate bar to be ordered:

How many boxes of Oh Henry bars would you like (48 bars per box) ? 2
How many boxes of Coffee Crisp bars would you like (48 bars per box) ? 15
How many boxes of Aero bars would you like (48 bars per box) ? 10
How many boxes of Smarties would you like (48 bars per box) ? 130
How many boxes of Crunchie bars would you like (48 bars per box) ? 1

Your program should then display the total cost for each type of chocolate bar and the sub total indicating the final cost before taxes. It should also then display the HST (13%) amount followed by the total amount due as follows:

2 boxes of Oh Henry ($0.65 x 48) = $ 62.40
15 boxes of Coffee Crisp ($0.80 x 48) = $ 576.00
10 boxes of Aero ($0.60 x 48) = $ 288.00
130 boxes of Smarties ($0.70 x 48) = $ 4368.00
1 boxes of Crunchie ($0.75 x 48) = $ 36.00

Sub Total = $ 5330.40
HST = $ 692.95

Amount Due = $ 6023.35

Notice how things are nicely aligned and formatted. Ensure that your code is also properly lined up.

Test your program with each of the following test cases as well as the above test case:

<table>
<thead>
<tr>
<th>Oh Henry</th>
<th>Coffee Crisp</th>
<th>Aero</th>
<th>Smarties</th>
<th>Crunchie</th>
<th>SubTotal</th>
<th>HST</th>
<th>Amount Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$168.00</td>
<td>$21.84</td>
<td>$189.84</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>$508.80</td>
<td>$66.14</td>
<td>$574.94</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$ 0.00</td>
<td>$ 0.00</td>
<td>$ 0.00</td>
</tr>
</tbody>
</table>
(2) Intersection Test

In many areas of science (e.g., computer graphics, computational geometry, image analysis, robotics, physics), it is often necessary to determine whether or not two line segments intersect. Assume that we have two 2D line segments one line segment has endpoints \((Ax,Ay)\) and \((Bx,By)\) and the other segment has endpoints \((Cx,Cy)\) and \((Dx,Dy)\). Write a C program called **intersectionTester** that asks the user for the eight values of the integer coordinates for two line segments and then displays a message indicating whether or not the line segments intersect. Assume that all coordinates are entered as integer values. Here are some examples:

In your code, you should write a function called **calculate()** that takes all 8 integer parameters and returns 1 if the line segments intersect and 0 otherwise. To determine intersection, you will need to handle vertical and horizontal lines as special cases. Otherwise, you can use the following equations to determine whether or not the line segments intersect:

\[
t = \frac{(Ax-Cx)(Cy-Dy) - (Ay-Cy)(Cx-Dx)}{(Ax-Bx)(Cy-Dy) - (Ay-By)(Cx-Dx)}
\]

\[
u = \frac{(Bx-Ax)(Ay-Cy) + (Ay-By)(Ax-Cx)}{(Ax-Bx)(Cy-Dy) - (Ay-By)(Cx-Dx)}
\]
When using the above equations, if \(0 \leq t \leq 1\) and \(0 \leq u \leq 1\) then the line segments intersect. This extra check ensures that the intersection of the two lines actually lies on the segments.

In your main function, you may hard code some calls to \textit{calculate()} with the coordinates for the 8 examples in the image above. Make sure that they all give the correct answer. Your output must be in this format (color added for highlighting purposes only):

Line segments (2,17)\(\rightarrow\)(15,10) and (7,19)\(\rightarrow\)(3,6) \textit{intersect.}
Line segments (2,17)\(\rightarrow\)(15,10) and (8,10)\(\rightarrow\)(3,6) \textit{do not intersect.}
Line segments (3,17)\(\rightarrow\)(3,6) and (7,10)\(\rightarrow\)(15,10) \textit{do not intersect.}
Line segments (4,18)\(\rightarrow\)(4,14) and (4,6)\(\rightarrow\)(4,3) \textit{do not intersect.}
Line segments (15,18)\(\rightarrow\)(15,6) and (15,14)\(\rightarrow\)(15,4) \textit{intersect.}
Line segments (3,17)\(\rightarrow\)(3,6) and (1,10)\(\rightarrow\)(15,10) \textit{intersect.}
Line segments (2,17)\(\rightarrow\)(12,17) and (7,17)\(\rightarrow\)(18,17) \textit{intersect.}
Line segments (2,10)\(\rightarrow\)(7,10) and (12,10)\(\rightarrow\)(18,10) \textit{do not intersect.}

(3) Histogram Generation

Write a program called \textit{histogram.c} that creates an array of 100 randomly-generated \texttt{char} values ranging from 0 to 9. The program should then display a histogram showing (as a string of * characters) the number of times that a certain number was generated. The histogram should have a format like the one shown here (although results will vary each time that you run it):

\[ \begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} 14 & * & * \\ 13 & * & * \\ 12 & * & * & * & * \\ 11 & * & * & * & * \\ 10 & * & * & * & * & * & * \\ 9 & * & * & * & * & * & * & * \\ 8 & * & * & * & * & * & * & * & * \\ 7 & * & * & * & * & * & * & * & * & * \\ 6 & * & * & * & * & * & * & * & * & * & * \\ 5 & * & * & * & * & * & * & * & * & * & * & * \\ 4 & * & * & * & * & * & * & * & * & * & * & * & * \\ 3 & * & * & * & * & * & * & * & * & * & * & * & * & * \\ 2 & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\ 1 & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\ \hline \end{array} \]

\[ 0123456789 \]

Note that there will always be 100 \texttt{*} characters shown. Also, the numbers on the left/vertical axis will vary depending on the random distribution. Therefore, the height of the histogram will vary. The width will always be the same, with the same horizontal axis.

In your code, you MUST ensure that the random numbers are different each time that you run your program. Also, your code must have three stages to it: (1) generate the random numbers into one array, (2) create the histogram in a different array, (3) display the histogram.

\textbf{IMPORTANT SUBMISSION INSTRUCTIONS:}

Submit all of your \texttt{c} source code files as well as your executables.

- If your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due!

- You WILL lose marks on this assignment if any of your files are missing. So, make sure that you hand in the correct files and version of your assignment. You will also lose marks if your code is not written neatly with proper indentation. See examples in the notes for proper style.