In this assignment, you will get used to working with the C language as well as working within the Linux/VirtualBox environment. The program will require you to use constants, variables, math expressions, random numbers, user I/O, IF statements, FOR loops and arrays.

(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          $0.87 each
2. Coffee Crisp      $0.88 each
3. Aero              $0.50 each
4. Smarties          $1.29 each
5. Crunchie          $0.98 each
```

Your program should then ask the user for the # of boxes of each type of chocolate bar to be ordered (assume less than 1000 each time … you do not need to check for valid numbers ranges nor valid characters):

```
How many boxes of Oh Henry bars would you like (24 bars per box)?  5
How many boxes of Coffee Crisp bars would you like (36 bars per box)? 10
How many boxes of Aero bars would you like (48 bars per box)? 50
How many boxes of Smarties would you like (24 bars per box)? 100
How many boxes of Crunchie bars would you like (36 bars per box)? 500
```

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:

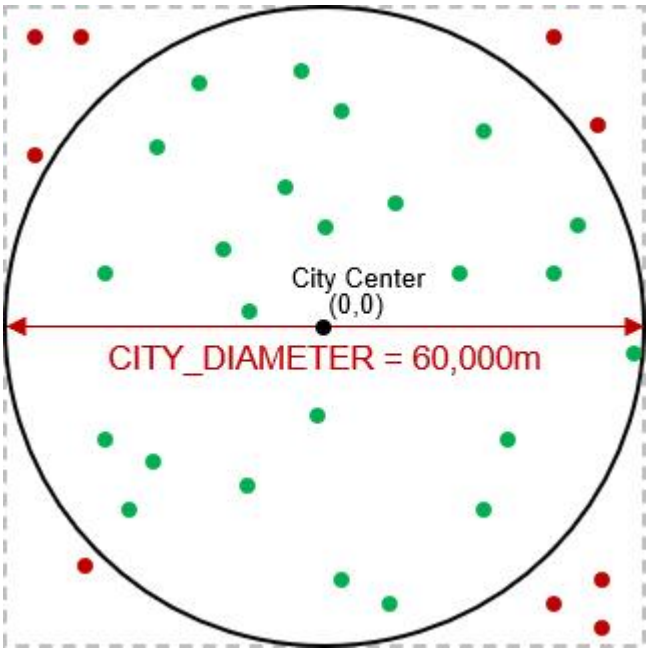
```
  5 boxes of Oh Henry      (24 x $0.87 per box) = $   104.40
 10 boxes of Coffee Crisp  (36 x $0.88 per box) = $   316.80
 50 boxes of Aero          (48 x $0.50 per box) = $  1200.00
100 boxes of Smarties      (24 x $1.29 per box) = $  3096.00
500 boxes of Crunchie      (36 x $0.98 per box) = $ 17640.00
-------------------------------------------------------------
Sub Total                                       = $ 22357.20
HST                                             = $  2906.44
=============================================================
Amount Due                                      = $ 25263.63
```

Notice how things are nicely aligned and formatted. Ensure that your code is also properly lined up. YOU MUST define the HST as well as all individual bar prices separately and bars per box separately as constants at the top of your code and use those throughout the program. You should not have any of these hard-coded in your calculations nor in your **printf()** statements. Test your program with each of the following test cases as well as the above test case:

Oh Henry	Coffee Crisp	Aero	Smarties	Crunchie	SubTotal	HST	Amount Due
1	1	1	1	1	$142.80	$ 18.56	$161.36
1	2	3	4	5	$456.48	$ 59.34	$515.82
0	0	0	0	0	$ 0.00	$ 0.00	$ 0.00

(2) Amazon Delivery

Amazon hires many delivery drivers to deliver their thousands of packages each day within a city. You will write a program to assign various package-delivery-locations to various drivers within the city's boundaries. The delivery area for the city will be defined by a circle with diameter of 60,000 meters centered at the city's center. Locations within the city's delivery area (i.e., green circles shown) will be allowed, but any locations outside of the city's delivery area (i.e., red circles shown) will not be considered.



The city center has location (**0**,**0**). Therefore, all valid delivery locations will have **x** and **y** coordinates each within the range of (**-30.0** km to **+30.0** km) … within the square bounding box. In addition, the delivery locations MUST also be within the city's radius. In the image here, the red circles have coordinates within the (**-30.0** km to **+30.0** km) but they are not within the city's radius, and are therefore invalid.

Write a program called **amazon1.c** that does the following:

a) The program should ask the user for the number of drivers, which must be a number from **2** to **200**, inclusively. If an invalid number is entered, it should display an error message and then ask the user again. This should continue until a valid number is entered. You may assume that only digit characters are entered by the user. Then, in the same manner, the program should ask the user for the number of packages to be delivered, which must be a number from **10** to **50000**, inclusively. Once both these numbers are valid, the program will continue.

b) The program should then compute a randomly-chosen (x, y) coordinate for each **driver** and store them in either one or two arrays (your choice). Each driver must have a coordinate that is within the city's circular area (e.g., like the green circles shown). The coordinates MUST be stored in Kilometers, so you will need them to be **floats**.

c) The program should then compute a randomly chosen (x, y) coordinate for each **package** and store them in either one or two arrays (your choice). Each package must have a coordinate that is within the city's circular area (e.g., like the green circles shown). The coordinates MUST be stored in Kilometers, so you will need them to be **floats**.

d) The program should then go through all the packages and determine the driver that is closest to it. You will need to use this formula to compute the distance between two points:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For each package, you will need to remember which driver was closest by storing the index of the driver (you can decide how to do that). If, for example, you have **5** drivers, then the indices are **0**, **1**, **2**, **3** and **4**.

e) The program should then display the number of drivers, the coordinates of each driver, the number of packages and the coordinates of each package along with the driver index for that package. The coordinates must all be displayed aligned nicely with the decimals aligned. Here is an example of the output for the program (the red indicates what the user entered):

```
Enter number of delivery drivers: 1
 *** Number of delivery drivers must be between 2 and 200 ***
Enter number of delivery drivers: 6734
 *** Number of delivery drivers must be between 2 and 200 ***
Enter number of delivery drivers: 3
Enter number of packages to be delivered: 4
 *** Number of packages must be between 10 and 50000 ***
Enter number of packages to be delivered: 789234
 *** Number of packages must be between 10 and 50000 ***
Enter number of packages to be delivered: 10

Drivers:  3
 15.76   0.87
 18.97  16.14
-13.68  19.94

Packages: 10
 21.37  -18.43  0
  9.94  -14.75  0
 -5.67    4.20  2
-10.13   12.12  2
  6.60   13.68  1
 12.90    4.32  0
  9.83  -15.38  0
 24.23   -1.21  0
 15.70    4.24  0
 28.06    3.36  0
```

Make sure to test your program a few times. The values should be as random as possible, so make sure that you do not use the same random number sequence each time.

(3) A Visual Display

Now copy your code into a file called **amazon2.c**. Adjust the code so that it just waits for the values from the user without display the prompts with **printf()**. Also, instead of printing error messages when invalid numbers are entered, the program should simply exit. Now adjust the program so that it prints out exactly: the number of drivers (without a newline character), followed by a space character, followed by the number of packages and then followed by a newline character. It should then print out all the packages one at a time as follows: print the **x** value, print a **space** character, print the **y** value, print a **space** character, print the **driver index** then print a **space** character. You MUST NOT print any newline characters at all when you print out the package information.

Run the program. Assuming that you enter **3** drivers and **10** packages, the output would look like this (the red was typed in by the user … and t he numbers will differ):

```
3
10
3 10
12.65 -19.03 2 -11.82 22.99 0 -11.72 -15.07 2 -15.28 14.30 0 -22.73 -5.30 2 -13.80 -
18.51 2 -12.80 11.97 0 16.44 -23.42 2 16.00 -2.92 2 14.71 14.86 1
```

Notice how it does not ask the user for the number of drivers and packages … it simply allows the user to enter them one at a time. It then outputs these two values and a newline character and then all the coordinates and driver indices for the packages.

The reason for these changes is that you will now run your data through a program that will display the data. The program is called **display.c**. Download it and compile it as follows:

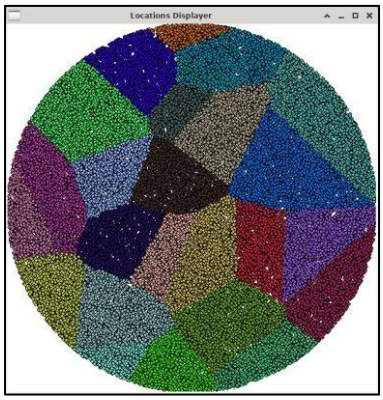
```
gcc -o display display.c -lX11.
```

The last part of that line is a "minus elle ex eleven" which is the standard C library that is used for windows and graphics. Do NOT alter the code in **display.c**.

To test your program, you will "pipe" the output (i.e., send the output) from the **amazon2** program into the **display** program. Just do the following in the shell window:

```
./amazon2 | ./display
```

The character in-between is a vertical bar character. On my keyboard it is above the ENTER key … but the location on your keyboard will likely differ.



When you run it, a blank window will appear. Click back onto the shell window and type in **25** followed by **50000**. You will need to grab the window and drag it to update it.

After a bit of a delay … you should see something that looks like this … although the colors and shapes will be different each time →

Try your program with different a different number of drivers and packages.

IMPORTANT SUBMISSION INSTRUCTIONS:

Submit all of your **.c source** files and all other files needed for testing/running your programs. Make sure that your name and student number is in each source file at the top as a comment.

The code **MUST** compile and run on the course VM.

- 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 and containing a reasonable number of comments.** See course notes for examples of what is proper indentation, writing style and reasonable commenting).
