Assignment 2

Name:____

Choose one (1) of the three (3) programs. Make sure to write the flowchart.

Distance and Midpoint in One-Dimension

Define the functions

Function Name:	`maximum()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`minimum()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`midpoint()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`distance()`
Parameter(s):	$x1$: float $x2$: float
Return:	float

where `maximum()` returns the maximum of x and y, `minimum()` returns the minimum of x and y, `midpoint()` returns the midpoint of x and y, and `distance()` returns the distance between x and y. In the main function,

1. prompt the user to enter two (2) numbers on a number line separately.
2. print the midpoint and distance of the input values.

Note: Do not use the cmath library. Likewise, round solutions to two decimal places. A possible output of the program is:

```
Enter the first number: 6.5
Enter the second number: -8

The midpoint is -0.75 and the distance is 14.50
```

Green text are inputs.

Distance and Midpoint in Three-Dimensions

Define the functions

Function Name:	`maximum()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`minimum()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`midpoint()`
Parameter(s):	x: float y: float
Return:	float

Function Name:	`square()`
Parameter(s):	x: float
Return:	float

Function Name:	`midpoint()`
Parameter(s):	*x1*: float *y1*: float *z1*: float *x2*: float *y2*: float *z2*: float *midx*: float reference *midy*: float reference *midz*: float reference
Return:	nothing

Function Name:	`distance()`
Parameter(s):	*x1*: float *y1*: float *z1*: float *x2*: float *y2*: float *z2*: float
Return:	float

where `maximum()` returns the maximum of x and y, `minimum()` returns the minimum of x and y, `square()` returns the square of x, `midpoint()` returns the midpoint of x and y, the second `midpoint` stores the midpoint of the x, y and z coordinates in *midx*, *midy* and *midz* respectively; and `distance()` returns the distance of the two three-dimensional points (namely, $(x1, y1, z1)$ and $(x2, y2, z2)$. In the main function,

1. prompt the user to enter two (2) three-dimensional points separately.
2. print the midpoint and distance of the input values.

Note: Only use the `sqrt()` function from cmath library. Likewise, round solutions to two decimal places. A possible output of the program is:

```
Enter the first number: 6 8 2.5
Enter the second number: -8 4 -4.5

The midpoint is (-1.00, 6.00, -1.00) and the distance is 16.16
```

Green text are inputs.

Vector Arithmetic

Define the functions

Function Name:	`toDegrees()`
Parameter(s):	*rad*: float
Return:	float

Function Name:	`toRadian()`
Parameter(s):	*deg*: float
Return:	float

Function Name:	`toRectangular()`
Parameter(s):	*magnitude*: float *angle*: float x: float reference y: float reference
Return:	nothing

Function Name:	`toPolar()`
Parameter(s):	x: float y: float *magnitude*: float reference *angle*: float reference
Return:	nothing

Function Name:	`subtract()`
Parameter(s):	*mag1*: float *ang1*: float *mag2*: float *ang2*: float *mag3*: float reference *ang3*: float reference
Return:	float

Function Name:	`add()`
Parameter(s):	*mag1*: float *ang1*: float *mag2*: float *ang2*: float *mag3*: float reference *ang3*: float reference
Return:	float

Function Name:	`PI()`
Parameter(s):	none
Return:	float

where `PI()` returns pi to at least 8 digits, `toRadian()` returns *deg* converted to radian, `toDegrees()` returns *rad* converted to degrees, `toRectangular()` stores the conversion of polar coordinates (*magnitude*, *angle*) into rectangular coordinates (x, y), `toPolar` stores the conversion of rectangular coordinates (x, y) into polar coordinates (*magnitude*, *angle*), `add` stores the sum of vector 1 (*mag1*, *ang1*) and vector 2 (*mag2*, *ang2*) into vector 3 (*mag3*, *ang3*), and `subtract` stores the difference of vector 2 (*mag2*, *ang2*) from vector 1 (*mag1*, *ang1*) into vector 3 (*mag3*, *ang3*) . In the main function,

1. prompt the user to enter two (2) vectors in polar coordinates with the angle in degrees separately.
2. print the vectors, their sum and differences in both polar and rectangular coordinates.

Note: Only use the `sqrt()`, `cos()`, `sin()`, `atan()`, `atan2()` functions from cmath library. Likewise, round solutions to one decimal places. A possible output of the program is:

```
Enter the first vector: 5 180
Enter the second vector: 4 90

v1:
Polar Coordinates: (4.0,180.0)
Rectangular Coordinates: (-4.0,0.0)
v2:
Polar Coordinates: (4.0, 90.0)
Rectangular Coordinates: (0.0,4.0)
v1 + v2:
Polar Coordinates: (5.7, -45.0)
Rectangular Coordinates: (-4.0,4.0)
v1 - v2:
Polar Coordinates: (5.7, -135.0)
Rectangular Coordinates: (-4.0,-4.0)
v2 - v1:
Polar Coordinates: (5.7, 45.0)
Rectangular Coordinates: (4.0,4.0)
```

Green text are inputs.