Actual LOC to-date: 0

Estimated LOC at completion: ???

total effort to-date

  William Mullen: ???

  Lambert Leong: 3 hours

**********Questions**********

Are we creating the component of the VOR instrument that handles the calculations?

Since we are not implementing an interface, are we to print the output to the console? (i.e., "GOOD, To, 45 degrees Left")

We are receiving input from the OBS. Are we to implement a fake OBS, for testing, in which the user inputs their location and heading via console?

We are also receiving input from a fake radio for testing purposes. Does/can the radio take user input to determine its location in relation to the aircraft?

Is the defection of the needle output to indicate the number of dots off center for <10 degrees and a far left/right output? Or, should the output just include a number and direction, left or right?

**********Code Location**********

We will be using git as our version control and all code, documentation, and assignment related files will be hosted on github.

An organization called "TeamFisher-ICS414" was created with a repo titled "VOR". A link to our teams organization is on the next line below.

https://github.com/TeamFisher-ICS414

**********Initial Thoughts and Design**********

The following design is based off of our interpretation until our questions are answered and further instruction is given.

Pseudo Code and Mock Design including classes and methods below:

```
class OBS{

    float x_coord, y_coord, z_coord, log_dest, lat_dest;

    double speed;

    void setPosition(float x_coord, float y_coord, float z_coord){

      //user input sets aircrafts location in the sky

    }
```

```
    void setDestination(float log_dest, float lat_dest){
        //user input sets destination via longitude and latitiude
    }
    void setSpeed(){
        //sets speed, not sure if we need this yet
    }
    float getXPosition(){
        //returns X postion
    }
    float getYPosition(){
        //returns Y postion
    }
    float getZPosition(){
        //returns Z postion
    }
    float getLong(){
        //returns longitude
    }
    float getLat(){
        // returns latitiude
    }
    double getSpeed(){
        //returns speed
    }
}
class radio{
    float x_coord, y_coord;
    double signalRange, intercept;
    void setPosition(){
        //sets position of radio
    }
    void setRange(){
        //sets the range of radio signal
    }
    void calcRadial(){
```

```
        //determines radial
    }
    float getXPosition(){
        //returns x postition
    }
    float getYPosition(){
        //returns y postition
    }
    float getRadial{
        //returns intecept
    }
}
class VOR{
    float craftXpos, craftYpos, craftZpos, log_dest, lat_dest, radioXpos, radioYpos, distance;
    double speed, signalRange, defelection;
    int signal; //0=bad, 1=good
    double heading; //0=left, 1=right
    OBS obs = new OBS();
    radio fake_radio = new radio();

    setCraft(){
        //calls OBS getters to orient craft
        //set speed
    }
    setRadio(){
        //sets radio with radio getters
    }
    float calcDistance(){
        //uses craft position and radio location to calc distance
    }
    int signal(){
        //uses distance and signalRange to determine if signal is GOOD or BAD
    }
    deflection(){
        //calcuated deflection based off OBS and radio
```

```
    }

    int getHeading(){

        //returns heading left/right via binary

    }

    double getDeflection(){

        //returns double between 0 and 359

    }

}
```

**********Testing**********

Using the OBS and radio classes, we can have the user set the location and destination of the craft as well as the location of the radio.

Knowing those components will allow us to calculate, by hand, what the deflection and VOR output should be.

Program correctness is evaluated on the program's console outputs closeness to hand calculated values.

Multiple locations and orientations will be assessed to catch corner cases such as if the plane is in the "cone of confusion" or too far away.Methods:

Using the OBS and radio classes, we can have the user set the location and destination of the craft as well as the location of the radio.

Knowing those components will allow us to calculate, by hand, what the deflection and VOR output should be.

Program correctness is evaluated on the program's console outputs closeness to hand calculated values.

Multiple locations and orientations will be assessed to catch corner cases such as if the plane is in the "cone of confusion" or too far away.