Actual LOC to-date: 0

Estimated LOC at completion: ???

total effort to-date

William Mullen: 3 hours

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?

Most of the information I've found on VOR navigation state that the pilot is attempting to follow an intended radial. Are we to assume the pilot's planned course is either directly away from or directly towards the VOR station in question? E.g. pass over VOR station until signal becomes good, set heading to a specific radial away from station, correct course until plane is in line with that radial, and further correct course so that VOR signal is behind the aircraft, so the aircraft is on course moving away from station? Or, alternately, should our system allow an aircraft to maintain course while passing a VOR on their left or right (which seems much harder)?

In the instuctions for the fake radio signal, you've left out the morse code identifying the station. Are we to assume (for now) the system is receiving a signal from the correct station, so we don't need to write a function to handle the station identifier signal?

The instructions do not mention a DME signal. Am I right in assuming our system does not receive one / does not know / does not need to know distance from VOR station? (except in the isntances when too close or two far away, which are indicated by the "good" or "bad" portion of the signal).

**********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:

```
//DESIGN A - for complex navigation


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
    }
}
```

**********Note**********

Upon further evaluation of the assignment, a different intepretation of the requirements led to a possible changes in our

methods and parameters. Feedback from our questions in the first part of the assignment will help us finalize our design

the following is another draft of our methods and parameters

****

```
//DESIGN B - for navigation to/from VOR station


/* main method
 * Parameters: int OBS, int signal
```

```
 * Returns: String adjustment (possible outputs "BAD" "AHEAD" "TL1" "TL2" "TL3" "TL4"
"TLM" "TR1" "TR2" "TR3" "TR4" "TRM" "FL1" "FL2" "FL3" "FL4" "FLM" "FR1" "FR2" "FR3" "FR4"
"FRM")

 * In return string, T/F is To/From, L/R is left/right, and 1-4 or M is 1/2 number for
degrees or Maximum correction

 *

 * First calls signalTest, returns "BAD" if signalTest is 1, proceeds otherwise

 * Calls signalParse, stores result

 * Calls heading, stores result

 * Returns value from heading

*/


/* signalTest

 * Parameters: int signal

 * Returns: 0 or 1 (0 for any signal with a "GOOD" indicator, 1 for a "BAD" indicator)

*/


/* signalParse

 * Parameters: int signal

 * Returns: int radial (0 to 359)

*/


/* heading

 * Parameters: int radial, int OBS

 * Returns: String adjustment

 *

 * Will return "BAD" if OBS - radial equals 89, 90, 91, 269, 270, 271, -89, -90, -91, -
269, -270, or -271

 * Will return "FWD" if OBS == radial

 * Will return a simplified version of the comparison otherwise, with To/From indicated
(e.g. "TL1", "FRM")

*/


/* radio

 * Parameters: int radial, boolean status (0 for "Good", 1 for "BAD")

 * Returns: int signal

 *

 * This is essentially a driver method used to produce a fake radio signal
```

*/

**********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.