CS 7A - Fall 2016 - Escape from Zorkon. Due 12/1

An astronaut's rocket has landed on the middle of a circular lake on the planet Zorkon. Unfortunately, there is an angry Zorkoid at the shore who is eager to capture and eat the astronaut. The Zorkoid can't swim, but it can run along the shore at a speed faster than the astronaut can propel the rocket through the water. The astronaut needs to get to the shore to launch the rocket and escape for the planet Zorkon. To do this, the astronaut adopts a strategy that says "swim directly away from wherever the Zorkoid is." The zorkoid's strategy is to run along the shore towards the side of the lake where the astronaut is. If the line containing the astronaut and the zorkoid is a diameter of the lake's circle, then check whether the zorkoid is less than a radius of the circle away, or not. If it is less than a radius away, the zorkoid sits still, if it is further than a radius away, the zorkoid runs in one direction or the other along the shore.
Your assignment is to write simulations of this situation where you iterate the positions of the Zordoid and your rocket with a step size in time, .

Basic Pseudocode:

```
initialize positions for rocket and Zorkoid
repeat until the astronaut reaches the shore:
update rocket position
update Zorkoid position
```

Use structures and files like those suggested here:
Traveller.h:

```
1  /// Traveller.h header file
2
3
4  ///include files:
5  //...
6  //...
7  const double step = 0.01;
8  const double lakeRadius{1};
9
10 class Point {
11 public:
12     double x;
13     double y;
14     Point(double x, double y) : x(x), y(y) {}
15 };
16 class Traveller {
17 public:
18     Point currentLoc;
19     double speed;
20     double bearing{0};
21     vector<Point> path;
22     void updateLoc();
23     void updateBearing(double);
24     void updatePath();
25     void printPath();
26     bool onShore();
27     Traveller(Point p, double s, double b);
28 };
```

Traveller.cpp:

```
#include "Traveller.h"

Traveller::Traveller(Point p, double s, double b) :  currentLoc(p), speed(s), bearin
void Traveller::updateLoc(Point p) {
    currentLoc = p;
}
bool Traveller::onShore() {
    return currentLoc.x*currentLoc.x+currentLoc.y*currentLoc.y >= lakeRadius*lakeRad
}
//...more definitions, as needed
```

escape.cpp:

```
#include "Traveller.h"

const int maxSteps = 1000;
int main() {
    int steps{0};
    const double Pi{3.14159265358979};
    Point east(1,0);
    Point origin(0,0);
    Traveller spaceman(origin,step,0);
    Traveller zorkoid(east,4*step,Pi);
    while(!spaceman.onshore()) {
        //update zorkoid's position (note that it must be on shore) and path
        //update spaceman's bearing
        //update spaceman's position and path
        //...
    }
}
```

Hint: For the "swim directly away from wherever the Zorkoid is" strategy, find the vector (physics style vector) from the zorkoid to the astronaut

(`zorkoid.currentLoc.x` − `astronaut.currentLoc.x`, `zorkoid.currentLoc.y` − `astronaut.currentLoc.y`

Then divide this vector by its length to get a vector, $\vec{v}$ of length 1. Increment the x-coordinate of the astronaut's position by `step*v.x` and increment the y-coordinate by `step*v.y`.

1. Write code to meet these specifications:
   Precondition: The positions of the astronaut and the zorkoid, the step size, `step`, of the astronaut and the step size of the zorkoid, some multiple of `step`, say, `4*step`, the radius of the lake.

   Postcondition: The paths of astronaut and the zorkoid and zorkoid and whether or not the astronaut has reached the shore in a prescribed time limit.

   Vary the step size, speed multiplier of the zorkoid to experiment with the results.
2. Write the paths of some interesting simulation(s) to a .txt file and import them into Excel where you can graph them.
3. What happens when you change the shape of the lake to an ellipse with major axis 4 and minor axis 2?
4. Write a short paper describing your results.

   Submit the code using your initials in the usual format: say XX_escape.cpp