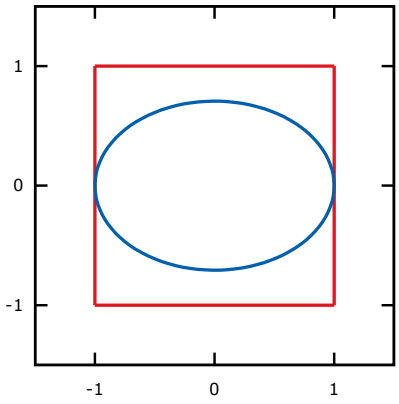
Problem 1: (Monte Carlo Experiments)
The Monte Carlo method is used in modeling a wide-range of physical systems at the forefront of scientific research today. Let's consider the problem of estimating the area of the region $x^2 + 2y^2 \leq 1$ by utilizing the Monte Carlo method. This region is enclosed by a blue ellipse, which is inscribed in a 2×2 red square (shown in the figure).



The experiment simply consists of throwing darts on this figure completely at random (meaning that every point in the red square has an equal chance of being hit by the dart). You keep throwing darts at random. All of the darts fall within the square, but not all of them fall within the ellipse. If you throw darts completely at random, this experiment estimates the ratio of the area of the ellipse to the area of the square, by counting the number of darts within each area.
Program this Monte Carlo method to compute the area of the blue ellipse using different numbers of darts. For each experiment, use N darts where $N = 10, 10^2, \cdots, 10^6$, and output the estimated area and the corresponding numerical error |estimated area − exact area|. The exact area for the ellipse is given by $\frac{\sqrt{2}}{2}\pi \approx 2.22144146907918$.

A successful run of your code may look like:

```
# darts: 10       estimated area = 2.40000   error = 0.17856
# darts: 100      estimated area = 2.16000   error = 0.06144
# darts: 1000     estimated area = 2.18000   error = 0.04144
# darts: 10000    estimated area = 2.23680   error = 0.01536
# darts: 100000   estimated area = 2.21072   error = 0.01072
# darts: 1000000  estimated area = 2.22037   error = 0.00107
```

About: This program is to give you practice using loops, the random number generator, math library, and output formatting. You may use `<iomanip>` to format your output.

Problem 2: (Random Walk Robot)

A robot is initially located at position $(0, 0)$ in a grid $[-5, 5] \times [-5, 5]$. The robot can move randomly in any of the directions: up, down, left, right. The robot can only move one step at a time. For each move, print the direction of the move and the current position of the robot. If the robot makes a circle, which means it moves back to the original place, print "Back to the origin!" to the console and stop the program. If it reaches the boundary of the grid, print "Hit the boundary!" to the console and stop the program.

A successful run of your code may look like:

```
Down      (0,-1)
Down      (0,-2)
Up        (0,-1)
Left      (-1,-1)
Left      (-2,-1)
Up        (-2,0)
Left      (-3,0)
Left      (-4,0)
Left      (-5,0)
Hit the boundary!
```

or

```
Left      (-1,0)
Down      (-1,-1)
Right     (0,-1)
Up        (0,0)
Back to the origin!
```

About: This program is to give you practice using the control flow, the random number generator, and output formatting. You may use `<iomanip>` to format your output.

Instructions for submission:

- Name your files exactly `hw4_P1.cpp` and `hw4_P2.cpp` respectively.
- You may NOT use `#include "stdafx.h"`.
- Add code description in the comment at the beginning of the file. A sample description may look like:

```
/*  PIC 10A 2A, Homework 1
    Purpose: miles to kilometers and feet converter
    Author: Hanqin Cai
    Date: 10/10/2018
*/
```

- Submit only `hw4_P1.cpp` and `hw4_P2.cpp` on BruinLearn.