Problem 5

Bouncing Marbles

9 Points

Program Name: marbles.cpp   Input File: marbles.dat

In this problem, you are to write a program that will determine the path of a marble as it bounces through a maze. The maze consists of a 10 by 10 set of cells and marbles only travel in exactly 1 of 4 directions: toward the top, toward the bottom, toward the left, or toward the right. Cells in the maze can contain the structures in the following table.

Cell Structures		**Where the marble goes if it approaches from the…**			
		Left	**Right**	**Bottom**	**Top**
.	Empty cell. The marble travels through this cell in the same direction.	Right	Left	Top	Bottom
/	Angle #1. The marble bounces at 90 degrees.	Top	Bottom	Right	Left
\	Angle #2. The marble bounces at 90 degrees.	Bottom	Top	Left	Right
O	Blocked cell. The marble bounces back in the direction from which it came.	Left	Right	Bottom	Top

Table 3 : Description of Cell Structures in the Maze

Your program will be given a 10 by 10 maze. Your program is to determine if the maze has the potential to trap a marble (imperfect maze) or if there is **no** position/direction combination on the maze that can keep the marble from bouncing off of the maze (perfect maze). For example, you can see from the maze below that a marble that starts at row 0 column 2 (0,2) and is traveling toward the bottom of the maze will:

- Bounce toward the right at (1,2),
- Bounce toward the bottom at (1,7),
- Bounce toward the left at (4,7),
- Bounce toward the bottom at (4,0), and
- Exit the maze from position (9,0).

```
            Columns
          0123456789
        0 .O....OOO.
        1 ..\....\O.
        2 ...O./....
        3 ..//......
   Rows 4 /....../.O
        5 .........O
        6 ./...O...O
        7 ..O.../..O
        8 ......./..
        9 ..OOOOOO/.
```

Figure 3: Sample Imperfect Maze

Likewise, a marble that enters from (8,2) traveling toward the right will:

- Bounce toward the top at (8,7),
- Bounce toward the right at (4,7),
- Bounce toward the left at (4,9),
- Bounce toward the bottom at (4,7),
- Bounce toward the left at (8,7), and
- Exit the maze from position (8,0).

Finally, a marble that enters the maze from (2,6) traveling toward the bottom will:

- Bounce toward the left at (7,6),
- Bounce toward the right at (7,2),
- Bounce toward the top at (7,6),
- Bounce toward the bottom at (0,6),
- Bounce toward the left at (7,6), and
- Continue bouncing forever (an imperfect maze).

A perfect maze does not have the potential to strand a marble bouncing forever on it. Therefore, the maze in Figure 3 is an imperfect maze. It is left to you to verify that the maze in Figure 4 is a perfect maze because there is no way to strand a marble on it (i.e. all marbles will exit).

```
  0123456789
0 ......\...
1 ..........
2 .../...O..
3 ......\...
4 ..O.......
5 ....O.....
6 ../.../...
7 ........O.
8 ..../.....
9 ..\..O....
```

Figure 4: Sample Perfect Maze

Input

Input to your program will consist of a series of mazes. Each maze consists of exactly 10 lines each with 10 columns. Each line is made up of only those characters in **Table 3**. Your program will read in the mazes one at a time and determine if it is a perfect maze (marbles cannot be trapped) or an imperfect maze (marbles can be trapped). You can be assured that all mazes in the input file are correct and complete. There will be exactly one blank line after each maze (and no other blank lines anywhere in the input file). There will be no extraneous input or embedded or trailing spaces. Your program should read to the end of file.

Output

For each maze from the input file, your program should print the message “Perfect Maze” if the maze cannot trap a marble. If the maze can trap a marble, your program should print the message “Imperfect Maze”. Messages should begin in column 1. There should be no extraneous output including embedded or trailing blanks and blank lines.

Example: Input File

```
.O....OOO.
..\....\O.
...O./....
..//......
/....../.O
.........O
./...O...O
..O.../..O
......./..
..OOOOOO/.

......\...
..........
.../...O..
......\...
..O.......
....O.....
../.../...
........O.
..../.....
..\..O....
```

```
..........
..........
..........
.../.\....
..........
...\./....
..........
..........
..........
..........
```

Output to screen

```
Imperfect Maze
Perfect Maze
Imperfect Maze
```