Project pp7 - Revised 10/25

Programming project due week of Mon 10/28 (wk. 9)

Due dates:
Section 01 (Block H): Wed 10/30 11:00 pm
Section 02 (Block E): Thu 10/31 11:00 pm
Submissions may begin on Mon 10/28 at 5pm.

Project pp7 - Revised 10/25: 6 points. (1 additional style points.) A circle class
NOTE: the original version of this handout contained an error in the output of the program. The output
for pp7main.cpp is now corrected here. (The third line contained 7.5 instead of 5.0 for the radius but this is
illegal since it crosses the 0 boundary on both x and y.) Please also check out a second main function now
provided in /comp/11/lab/pp7mainB.cpp.
In this project you will implement a class for circles which are constrained by a bounding box. A circle
is described by the x,y coordinates of its center and by its (default size) radius. Every circle will have a
bounding box limiting its size; if its center is located so that the circle goes beyond the bounding box with
the default size then its actual radius is reduced so that it does not cross the boundary. The bounding box
is always based at (0,0,0) and defined by its top right corner (tx,ty). The example below illustrates these
concepts.
You should implement this in a class named boundedCircle. The interface of the class must include the
following functions:

- A default constructor: sets the center to (5.0,5.0), the default radius to 1.0, and the top right point of
  the bounding box to (10.0,10.0).
- A constructor that takes 2 arguments (x,y) for the coordinates of the center and set the radius and
  boundary as above.
- A function setRadius(float r) that sets the value of the default radius.
- A boolean function move(float dx, float dy) that adds dx to the x coordinate and dy to the y
  coordinate of the circle’s center. If the resulting coordinate is within the bounding box the center
  is moved and the function returns true; otherwise the center does not change and the function and
  returns false.
- A function doubleBoundary() which doubles the values (tx,ty) of the top right corner of the boundary.
- A function getRadius() that returns the actual radius (that is, the reduced one if the default radius
  goes beyond the boundary).
- A function display(), which prints the x,y coordinates and the actual radius. The format is given in
  the example below. Make sure to print a # sign before each number and to have no other # signs in
  your output.

Notice that the limitation on the actual size of the radius can come into effect (or cease to take effect) after
the setRadius(), move() or doubleBoundary() operations, as well as the second constructor.
All the functions in the interface should be public; all other members of the class should be private. You
should implement your class in files boundedCircle.h and boundedCircle.cpp.
We will test your class with several main functions; one such function, provided in /comp/11/lab/pp7main.cpp,
is given below. Make sure that this program runs with your class. You should also write other main functions
to test your code.
```cpp
#include <iostream>
#include <iomanip>
#include "boundedCircle.h"
using namespace std;

int main()
{
    boundedCircle dog;
    boundedCircle cat(7,8);

    cout << setiosflags(ios::fixed);
    cout << setiosflags(ios::showpoint);
    cout << setprecision(1);
    dog.display();
    dog.setRadius(7.5);
    dog.display();
    dog.doubleBoundary();
    dog.display();

    cat.display();
    if (cat.move(-8,0))
    { cout << "This shouldn’t happen!\n"; }
    else
    { cout << "Move failed!\n"; cat.display();
    cat.move(2.3,0.1);
    cat.display();
    cat.move(-3,-3);
    cat.display();

    return 0;
    }
}
```

When run this program should print:

Circle is at: # 5.0 # 5.0 Radius is # 1.0
Circle is at: # 5.0 # 5.0 Radius is # 5.0
Circle is at: # 5.0 # 5.0 Radius is # 5.0
Circle is at: # 7.0 # 8.0 Radius is # 1.0
Move failed!
Circle is at: # 7.0 # 8.0 Radius is # 1.0
Circle is at: # 9.3 # 8.1 Radius is # 0.7
Circle is at: # 6.3 # 5.1 Radius is # 1.0

**Programming style guidelines for this week**
To get the style point, you must satisfy all of the following requirements: (1) Write the class with public and private sections as described above, and (2) Have an introductory comment at the top of your program. (3) Have a comment for each variable and function describing their purpose. Your first submission will be used for style grading, even if your best grading score is on a different submission.

**Submitting your program**
You must use filenames as specified above. Assuming these are in the current directory on andante, you should submit by typing

provide comp11 pp7 boundedCircle.h boundedCircle.cpp