```
// ************************************************
// Header file Sphere.h for class Sphere.
// ************************************************
const double PI = 3.14159;

#ifndef SPHERE_H
#define SPHERE_H

class Sphere
{
public:
   Sphere();
   // Default constructor: Creates a sphere and
   // initializes its radius to a default value.
   // Precondition: None.
   // Postcondition: A sphere of radius 1 exists.

   Sphere(double initialRadius);
   // Constructor: Creates a sphere and initializes
   // its radius.
   // Precondition: initialRadius is the desired
   // radius.
   // Postcondition: A sphere of radius initialRadius
   // exists.

   void setRadius(double newRadius);
   // Sets (alters) the radius of an existing sphere.
   // Precondition: newRadius is the desired radius.
   // Postcondition: The sphere’s radius is newRadius.

   double getRadius() const;
   // Determines a sphere's radius.
   // Precondition: None.
   // Postcondition: Returns the radius.

   double getDiameter() const;
   // Determines a sphere's diameter.
   // Precondition: None.
   // Postcondition: Returns the diameter.

   double getCircumference() const;
   // Determines a sphere's circumference.
   // Precondition: PI is a named constant.
   // Postcondition: Returns the circumference.

   double getArea() const;
   // Determines a sphere's surface area.
   // Precondition: PI is a named constant.
   // Postcondition: Returns the surface area.

   double getVolume() const;
   // Determines a sphere's volume.
   // Precondition: PI is a named constant.
   // Postcondition: Returns the volume.

   void displayStatistics() const;
   // Displays statistics of a sphere.
```

```
   // Precondition: None.
   // Postcondition: Displays the radius, diameter,
   // circumference, area, and volume.

private:
   double theRadius; // the sphere's radius
}; // end class

// End of header file.
#endif

// ********************************************************
// Implementation file Sphere.cpp for the class Sphere.
// ********************************************************

#include "Sphere.h" // header file
#include <iostream>
using namespace std;

Sphere::Sphere(): theRadius(1.0)
{
} // end default constructor

Sphere::Sphere(double initialRadius)
{
   if (initialRadius > 0)
      theRadius = initialRadius;
   else
      theRadius = 1.0;
} // end constructor

void Sphere::setRadius(double newRadius)
{
   if (newRadius > 0)
      theRadius = newRadius;
   else
      theRadius = 1.0;
} // end setRadius

double Sphere::getRadius() const
{
   return theRadius;
} // end getRadius

double Sphere::getDiameter() const
{
   return 2.0 * theRadius;
} // end getDiameter

double Sphere::getCircumference() const
{
   return PI * getDiameter();
} // end getCircumference

double Sphere::getArea() const
{
   return 4.0 * PI * theRadius * theRadius;
```

```
} // end getArea

double Sphere::getVolume() const
{
   double radiusCubed = theRadius * theRadius * theRadius;
   return (4.0 * PI * radiusCubed)/3.0;
} // end getVolume

void Sphere::displayStatistics() const
{
   cout << "\nRadius = " << getRadius()
        << "\nDiameter = " << getDiameter()
        << "\nCircumference = " << getCircumference()
        << "\nArea = " << getArea()
        << "\nVolume = " << getVolume() << endl;
} // end displayStatistics
// End of implementation file.

// ****************************************************
//         Client Program using ADT Sphere
// ****************************************************
#include <iostream>
#include "Sphere.h"
using namespace std;

int main()
{
   Sphere unitSphere;    // radius is 1.0
   Sphere mySphere(5.1); // radius is 5.0

   unitSphere.displayStatistics();
   mySphere.setRadius(4.2); // resets radius to 4.2
   cout << mySphere.getDiameter() << endl;

   return 0;

} // end main
```