Solution of Tutorial (Chap. 4)
(Sphere Class)

Tutorial: Define a Sphere class derived from the Circle class that was previously defined for chapter 3 tutorial. Public methods are:

- **Constructors for given int center coordinates and radius, int radius with center at (0,0,0), and double radius with center at (0,0,0).**
- **Methods for obtaining volume and surface area.**
- **Methods for getting the values of the data members.**
- **Methods for resizing and shifting.**
- **Method for printing values of a Sphere object in a clear form.**

Solution:

```
#include <iostream.h>
#include <math.h>

// Definition of Circle class without validation as stated for chapter 3 tutorial.
// This is not required for problem solution, but is needed for the full program.

class Circle{
        int x,y;
        double r;
        double pi() { return 3.1415927; }

    public:
        Circle( int cx, int cy, int rad ) { x=cx; y=cy; r=rad; }
        Circle( int rad ) { x=y=0; r=rad; }
        Circle( double rad ) { x=y=0; r=rad; }
        double area() { return pi()*r*r; }
        double circum() { return 2*pi()*r; }
        int getX() { return x; }
        int getY() { return y; }
        double getR() { return r; }
        void resize( double rfac ) { r=r*fabs(rfac); }
        void shift( int dx, int dy ) { x+=dx; y+=dy; }
        void shift( int d ) { x+=d; y+=d; }
        void print();
};

void Circle::print() { cout<<"Circle @ ("<<x<<','<<y<<"), rad="<<r<<'\n'; }
```

```
// Definition of Sphere class

class Sphere : public Circle {

    private:
        int z;
        double pi() { return 3.1415927; }

    public:
        Sphere( int, int, int, int );          // x, y, z, r
        Sphere( int );                         // r, x=y=z=0
        Sphere( double );                      // r, x=y=z=0
        double volume();
        double area();
        double circum() { return 0.0; }        // override
        int getZ() { return z; }
        void shift( int, int, int );           // dx, dy, dz
        void shift( int );                     // d for x,y,z
        void print();
};

Sphere::Sphere( int cx, int cy, int cz, int rad )
    : Circle( cx, cy, rad ) {
        z=cz; }

Sphere::Sphere( int rad )
    : Circle( rad ) {
        z=0; }

Sphere::Sphere( double rad )
    : Circle( rad ) {
        z=0; }

double Sphere::volume() {
    return 4.0/3.0*pi()*pow( getR(), 3 ); }

double Sphere::area() {
    return 4*pi()*pow( getR(), 2 ); }

void Sphere::shift( int dx, int dy, int dz ) {
    Circle::shift( dx, dy );
    z+=dz; }

void Sphere::shift( int d ) {
    Circle::shift( d );
    z+=d; }
```

```
void Sphere::print() {
    cout << "Sphere @ (" << getX() << ',' << getY()
        << ',' << z << "), rad=" << getR() << '\n'; }


// Run Example

void main() {
    Sphere s1 = Sphere( 50, 60, 80, 25);
    Sphere s2 = Sphere( s1.getR()/2 );
    cout << "Initially:\n";
    s1.print(); s2.print();

    s1.shift( -30, 60, -20);
    s1.resize( 1.5 );
    cout << "Finally:\n";
    s1.print(); s2.print();

    cout<< "Volumes: " << s1.volume() << " & " << s2.volume() << '\n';
    cout<< "Areas:    " << s1.area() << " & " << s2.area() << '\n';
}
```