Solution of Tutorial (Chap. 3)
(Circle Class)

Tutorial: Write definitions of a Circle class with its methods:

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

Data guarding will be added for the following conditions:

- **Circle radius must be positive.**
- **Circle objects remain whole on screen of resolution ResX by ResY.**

```
#include <math.h>
#include <iostream.h>
const int ResX=800, ResY=600;

class Circle {
    private :
        int x,y;
        double r;
    public :
        Circle( int, int, int );                    // x, y, r
        Circle( int );                              // r, x=y=0
        Circle( double );                           // r, x=y=0
        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( int rfactor );                 //increase r by int rfactor
        void resize( double rfactor );              //resize r by real rfactor
        void shift( int dx, int dy );               //shift x by dx, y by dy
        void shift( int d );                        //shift x & y by d
        void shift( char f );                       //shift by 2r: f='R','L','U','D'
        void print();
    private :
        double pi() { return 3.1415927; }
        void validate();
        void validateR();
        void validateXY();
};
```

```
Circle::Circle( int cx, int cy, int rad ) {
    x=cx; y=cy; r=rad;
    validate(); }

Circle::Circle( int rad ) {
    x=0; y=0; r=rad;
    validate(); }

Circle::Circle( double rad ) {
    x=0; y=0; r=rad;
    validate(); }

void Circle::validate() {
    validateR();
    validateXY(); }

void Circle::validateR() {
    r=fabs( r );
    if( r==0 ) r=10.0;
    double rmax=(ResY-1)/2.0;
    if( r>rmax ) r=rmax; }

void Circle::validateXY() {
    int rr=int( r+0.5 );
    if( x-rr<0 ) x=rr;
    else if( x+rr>=ResX ) x=ResX-1-rr;
    if( y-rr<0 ) y=rr;
    else if( y+rr>=ResY ) y=ResY-1-rr; }

void Circle::resize( int rfactor ) {
    r=r*fabs( rfactor );
    validate(); }

void Circle::resize( double rfactor ) {
    r=r*rfactor;
    validate(); }

void Circle::shift( int dx, int dy ) {
    x += dx; y += dy;
    validateXY(); }

void Circle::shift( int d ) {
    x += d; y += d;
    validateXY(); }
```

```
void Circle::shift( char f ) {
    int dia = int( 2*r+0.5 );
    switch (  f ) {
        case 'R' : x += dia; break;
        case 'L' : x −= dia; break;
        case 'U' : y += dia; break;
        case 'D' : y −= dia; break;
        default  : ;
    }
    validateXY(); }

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

Run Example

```
void main() {
    Circle c = Circle( 50, 60, 20 );
    c.print();
    c.shift( -10, 60 ); c.print();
    c.resize( 5 ); c.print();
    c.shift( 'R' ); c.print();
    cout << "area=" << c.area() << '\n';
    cout << "circ=" << c.circum() << '\n';
}
```