The purpose of this assignment is to design a polymorphic type. We are going to create a collection of shapes. We will represent each different shape by a different class, but the base class will provide all of the things needed for the shape. These are: 1. A locator point, 2. A template for the function "Area" and a template for the function "Ident". Begin by defining the class *Point*, then the class *Shape* and then the classes for the shapes *Rectangle*, *Parallelogram*, *Triangle*, *Circle*, and *Trapezoid*. The specifications for each of the classes are given below. Most of the implementations are straightforward.

The class Shape is an abstract class, because it defines functions with the body "=0". No instance of this class can be created, but pointers to the class can be used. The derived classes are non-abstract, because they are required to define the "Area" and "Ident" functions, and because they do not introduce any new "=0" functions.

Ident returns one of the following strings:

"Rectangle: " "Parallelogram: " "Triangle: " "Circle: " "Trapezoid: "

All points, lengths, and so forth are relative to the locator point of the "Shape" base class, so the area formulas are
Rectangle: $w \times h$ (TopRight.X * TopRight.Y)
Parallelogram: $w \times h$ (Length=width, height=y coordinate of TopLeft)
Triangle: $(w \times h)/2$ (Length=width, height=y coordinate of Apex)
Circle: πr^2
Trapezoid: Average of top and bottom times height.

Use the following definition for PI.
#define PI (float)3.1415926535

Put your class definitions in the file: "Shapes.h"
Put your function bodies in the file: "Shapes.cpp"

Put your main program in the file "Poly.cpp".

Your main program must open the file "Poly.txt" and read shape data from it. Each line in this file defines one shape. The first character on the line will be 'R', 'P', 'T', 'C', or 'Z' (Z=Trapezoid, the others are obvious.)

Each letter will be followed by a bunch of floating point numbers, the exact number depends on the type of shape. The first two numbers after any letter are the X and Y coordinates of the locator point (respectively). The rest are as follows:

R – X of Top Right, Y of Top Right
P – X of Top Left, Y of Top Left, Length
T – X of Apex, Y of Apex, Length
C – Radius
Z – X of Top Left, Y of Top Left, Length, Top Length

Create an array of pointers to the "Shape" class. (A fixed size of 100 is OK.) Read each shape from the "Poly.txt" file, create an appropriate object for it (using *new*), and store the pointer to the object in the array. Count how many shapes you have read. Read until end of file. Then go through the array and print out the ident and area of each shape on a single line as follows:

Triangle: 25
Rectangle: 45.6

And so forth.

Here are the specifications for each of the classes.

Class Point: inheritance: NONE

Data: float x and float y. (private)

1. Default Constructor (init to zero)
2. Destructor
3. float, float constructor (copy arguments to x and y)
4. copy constructor
5. assignment overload
6. GetX and GetY functions to return values of x and y.

Class Shape: Inheritance: NONE, abstract class.

Data: Point Location(private)

1. Default constructor (default constructor for Point)
2. virtual destructor
3. Point constructor (copy constructor for Point)
4. float, float constructor (float float constructor for Point)
5. virtual float Area(void)=0;
6. virtual char * Ident(void)=0;

Class Rectangle: Inheritance: public Shape non-abstract class.

Data: Point TopRight (protected)

1. Default Constructor (default of Shape)
2. virtual destructor
3. Point, Point, constructor (NewLoc, & items)
   Use appropriate constructors for aggregate & base classes.
4. Define virtual float Area(void);
5. Define virtual char * Ident(void);

Class Parallelogram: Inheritance: public Shape non-abstract class.

Data: float Length Point TopLeft (protected)

1. Default Constructor (default of Shape)
2. virtual destructor
3. Point, Point, float constructor (NewLoc, & items)
   Use appropriate constructors for aggregate & base classes.
4. Define virtual float Area(void);
5. Define virtual char * Ident(void);

Class Triangle : Inheritance: public Shape non-abstract class.

Data: float Length Point Apex (protected)

1. Default Constructor (default of Shape)
2. virtual destructor
3. Point, Point, float constructor (NewLoc, & items)
   Use appropriate constructors for aggregate & base classes.
4. Define virtual float Area(void);
5. Define virtual char * Ident(void);

Class Circle : Inheritance: public Shape non-abstract class.

Data: float Radius (float)

1. Default Constructor (default of Shape)
2. virtual destructor
3. Point, float constructor (NewLoc, & items)
   Use appropriate constructors for aggregate & base classes.
4. Define virtual float Area(void);
5. Define virtual char * Ident(void);

Class Trapezoid : Inheritance: public Shape non-abstract class.

Data: float TopLength, Point TopLeft, float Length (protected)
Data: float Radius (float)

1. Default Constructor (default of Shape)
2. virtual destructor
3. Point, Point, float, float constructor (NewLoc, & items)
   Use appropriate constructors for aggregate & base classes.
4. Define virtual float Area(void);
5. Define virtual char * Ident(void);