Objectives

- To practice fundamental object-oriented programming (OOP) concepts such as encapsulation, inheritance, and polymorphism
- To learn how to define an inheritance hierarchy of classes implementing a common interface
- To learn how to define and use virtual functions and how to override them in order to make polymorphism possible in C++
- To learn how to implement and use two-dimensional arrays using **array** and **vector**, the two simplest container class templates in the C++ Standard Template Library (STL)
- To provide an opportunity for you to practice programming!

Geometric Shape Modeling

Using simple geometric shapes, this assignment will give you practice with fundamental principles of OOP: encapsulation, inheritance and polymorphism. The geometric shapes considered are simple two-dimensional shapes that can be reasonably depicted textually on the computer screen, such as squares, rectangles, and specific kinds of triangles and rhombuses.

You will, of course, recall that polymorphism in C++ requires that there must exist:

1. an inheritance hierarchy of classes,
2. a **virtual** member function with the same signature in the classes in the hierarchy,
3. a pointer or a reference of a base class type, which is used to invoke the virtual functions.

To build an inheritance hierarchy that models the shapes used in this assignments, we first need to specify the general aspects to be shared by all objects of classes in the hierarchy.

Common attributes: each shape object is to have:

1. a distinct identity number, an integer
2. a generic name, such as “Rectangle”
3. a descriptive name, such as “Swimming Pool”

Common behavior: each shape object is to provide the following operations:

1. get the values of its attributes
2. set the descriptive name
3. generate a string representation for the shape
4. scale the shape by a given integer factor
5. compute the geometric area and perimeter of the shape
6. draw a textual image of the shape on a given two dimensional grid
7. determine the height and width of the shape's bounding box, the smallest box enclosing the textual image of the shape
8. compute the *screen area* of the shape:
   - the number of characters that form the textual image of the shape
9. compute the *screen perimeter* of the shape:
   - the number of characters on the borders of the textual image of the shape

Let **Shape** be the name of the class that encapsulates the shape properties and operations listed above. Clearly, **Shape** must be *abstract* because operations 3-9 are so general that it cannot possibly know how to implement them. Declaring operations 3-9 as *pure* **virtual** function, **Shape** serves as a common interface to all classes in the hierarchy, including *concrete* classes, which implement all pure virtual functions.

Recall that you cannot create objects of an abstract class, but you can declare pointers and references to that class type:

```
class Shape // an abstract class
{
  public: virtual void area() = 0; // a pure virtual function
  // ...
};
void f1(Shape*); // ok
void f2(Shape&); // ok
void f3(Shape); // error; can't create objects of an abstract class
Shape shp1; // error; can't create objects of an abstract class
```

Concrete Shapes

This assignment picks only four geometric shapes that can be textually rendered into visually identifiable shape patterns:

1. Rectangles of width w and height h
2. Isosceles triangles with *odd* base b and height $h = (b + 1)/2$
3. Right (isosceles) triangles with base b and height $h = b$

4. Rhombus shapes with both equal and *odd* diagonal length $d \geq 1$

Here are some of the specific properties, where lengths are measured in character units:

Rectangle shapes

Construction values:	width $w \geq 1$ and height $h \geq 1$
Sample image:	a rectangle with $w = 9$ and $h = 5$
How to scale(n)	set $w \leftarrow w+n$ and $h \leftarrow h+n$, provided that both $w+n \geq 1$ and $h+n \geq 1$; otherwise, no scale.

Sample Image

```
*********
*********
*********
*********
*********
```

Isosceles triangles with *odd* base b and height $h = (b+1)/2$

Construction value:	base $b \geq 1$ and odd
Sample image:	An isosceles triangle, $b = 9$ and $h = 5$
How to scale(n)	if $b + 2n \geq 1$, set $b \leftarrow b + 2n$ and $h \leftarrow (b+1)/2$, in that order; otherwise, no scale.

Sample Image

```
    *
   ***
  *****
 *******
*********
```

Right (isosceles) triangles with base b and height $h = b$

Construction value:	base $b \geq 1$
Sample image:	a right triangle with $b = 5$.
How to scale(n)	Set both b and h to $b+n$, provided that $b+n \geq 1$; otherwise, no scale.

Sample Image

```
*
**
***
****
*****
```

Rhombus shapes with both equal and *odd* diagonal length $d \geq 1$

Construction value:	diagonal $d \geq 1$ and odd
Sample image:	a rhombus with $d = 5$
How to scale(n)	if $d+2n \geq 1$ set $d \leftarrow d+2n$; otherwise, no scale.

Sample Image

```
  *
 ***
*****
 ***
  *
```

Thus, at construction, a **Rectangle** shape requires the values of both its height and width, whereas the other three shapes each require a single value for the length of their respective horizontal attribute.

The remaining specifics of the concrete shapes above are specified in the following table.

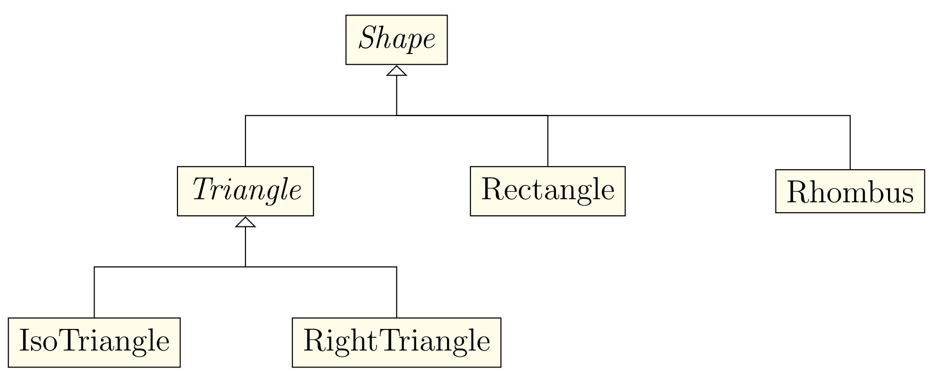
Concrete Shape Specifics

	Rectangle	Rhombus	Right Triangle	Isosceles Triangle
construction values	h, w	d, if d is even set $d \leftarrow d+1$	b	b, if b is even set $b \leftarrow b+1$
computed values			$h = b$	$h = (b+1)/2$
height of bounding box	h	d	h	h
width of bounding box	w	d	b	b
geometric area	hw	$d^2/2$	$hb/2$	$hb/2$
Screen Area	hw	$2n(n+1)+1$, $n = \lfloor d/2 \rfloor$	$h(h+1)/2$	h^2
geometric perimeter	$2(h+w)$	$(2\sqrt{2})d$	$(2+\sqrt{2})h$	$b + 2\sqrt{0.25b^2 + h^2}$
Screen Perimeter	$2(h+w)-4$	$2(d-1)$	$3(h-1)$	$4(h-1)$

Note the the height (vertical length) and width (horizontal length) of the bounding box for a shape are *not* stored anywhere; they are provided on demand.

Task 1 of 2

Implement the class hierarchy below, where ***Shape*** and ***Triangle*** denote abstract classes.



The amount of coding required for this task is not a lot as your shape classes will be small. Be sure that common behavior (shared code) and common attributes (shared data) are pushed toward the top of your class hierarchy.

Here are a couple of examples along with the output they each generate:

```
Rectangle shape1(10, 3);
cout << shape1 << endl;
```

```
Shape Information
-----------------
Static type:   PK5Shape
Dynamic type:  9Rectangle
Generic name:  Rectangle
Description:   Generic Rectangle
id:            1
B. box width:  10
B. box height: 3
Scr area:      30
Geo area:      30.00
Scr perimeter: 22
Geo perimeter: 26.00
```

To get the name of the *static* type of a pointer **p** at runtime use **typeid(p).name()**, and to get the name of its *dynamic* type use **typeid(*p).name()**. You need to include the **<typeinfo>** header for this.

The actual names returned by these calls are implementation defined. For example, the output above was generated under MinGW 4.9.2, where **PK** in **PK5Shape** means "pointer to ~~**konst**~~ **const**", and **5** in **PK5Shape** means that the type name that follows it is **5** character long.

Microsoft VC++ produces more readable output as shown below.

```
Rectangle shape1(10, 15);
cout << shape1 << endl;
```

```
Shape Information
-----------------
Static type:   class Shape const *
Dynamic type:  class Rectangle
Generic name:  Rectangle
Description:   Generic Rectangle
id:            1
B. box width:  10
B. box height: 3
Scr area:      30
Geo area:      30.00
Scr perimeter: 22
Geo perimeter: 26.00
```

The ID number 1 for the shape is assigned during the construction of the object. The ID number of the next shape will be 2, the one after 3, and so on. These unique ID numbers are generated and assigned when shape objects are first constructed.

The generic name for a shape is the name of its class; it is set when the shape object is constructed.

The descriptive name for a shape defaults to the word `Generic` followed by the class name but can be supplied when the shape object is created:

```cpp
3 Rhombus ace(16, "Ace of diamond");
4 cout << ace.toString() << endl;
5 // or, equivalently:
6 cout << ace << endl;
```

```
Shape Information
-----------------
Static type:    PK5Shape
Dynamic type:   7Rhombus
Generic name:   Rhombus
Description:    Ace of diamond
id:             2
B. box width:   17
B. box height:  17
Scr area:       145
Geo area:       144.50
Scr perimeter:  32
Geo perimeter:  48.08
```

Note 1: Lines 4 and 6 of the code segment above show equivalent ways for printing shape information. The explicit call to the **toString()** function in line 4 generates string representation for the **ace** object. In line 6, the call to **toString()** is implicit.

Note 2: In line 3, the supplied height, 16, is invalid because it is even; to correct it, **Rhombus**'s constructor uses the next odd integer, 17, as the diagonal of object **ace**.

Here are two other examples of **Shape** objects.

```cpp
7  Isosceles iso(17);
8  // the following call is polymorphic but
9  // iso is neither a reference nor a pointer
10 cout << iso << endl; // how so?
11
12 /* equivalently:
13
14 Shape *isoptr = &iso;
15 cout << *isoptr << endl; // polymorphic call
16
17 Shape &isoref = iso;
18 cout << isoref << endl; // polymorphic call
19 */
```

```
Shape Information
-----------------
Static type:    PK5Shape
Dynamic type:   9Isosceles
Generic name:   Isosceles
Description:    Generic Isosceles
id:             3
B. box width:   17
B. box height:  9
Scr area:       81
Geo area:       76.50
Scr perimeter:  32
Geo perimeter:  41.76
```

```
RightTriangle rt(10, "Carpenter's square");
cout << rt << endl;
```

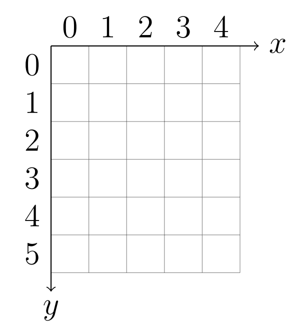
```
Shape Information
-----------------
Static type:   PK5Shape
Dynamic type:  13RightTriangle
Generic name:  Right Triangle
Description:   Carpenter's square
id:            4
B. box width:  10
B. box height: 10
Scr area:      55
Geo area:      50.00
Scr perimeter: 27
Geo perimeter: 34.14
```

Now a few words on **Shape**'s **draw** function prototyped as follows:

```
virtual vector<vector<char>> draw(char penChar = '*', char fillChar = ' ')const = 0;
```

It simply renders the textual image of the invoking shape object on a two-dimensional grid of type **vector<vector<char>>** and returns resulting image grid.

The image grid is a bounding box representation for the invoking shape. The grid rows are parallel to the x-axis, with row numbers increasing down. The grid columns are parallel to the y-axis, with column numbers increasing to the right. The origin of the grid is located at the top-left grid cell $(0, 0)$ at row 0 and column 0.



The **draw** function uses the supplied pen character **penChar** to render the shape image. Any cell not on the image is filled with the supplied fill character **fillChar**. As indicated in the function prototype above, **penChar** defaults to '$*$' and **fillChar** to the blank character.

To display an image grid on the screen we overload the **operator<<** as follows:

```
ostream& operator<< (ostream& sout, const vector<vector<char>> &grid)
{
   for (size_t r = 0; r < grid.size(); ++r)
   {
      for (size_t c = 0; c < grid[r].size(); ++c)
      {
         sout << grid[r][c];
      }
      sout << '\n';
   }

   /* or equivalently,

   for (vector<char> vec : grid)
   {
      for (char ch : vec)
      {
         sout << ch;
      }
      sout << '\n';
   }
    */
  return sout;
}
```

Here are some examples:

```
cout << shape1.draw() << endl;
```

```
**********
**********
**********
```

```
cout << ace.draw('o') << endl;
```

```
        o
       ooo
      ooooo
     ooooooo
    ooooooooo
   ooooooooooo
  ooooooooooooo
 ooooooooooooooo
ooooooooooooooooo
 ooooooooooooooo
  ooooooooooooo
   ooooooooooo
    ooooooooo
     ooooooo
      ooooo
       ooo
        o
```

```
cout << iso.draw('\\', '.') << endl;
```

```
........\........
.......\\\.......
......\\\\\......
.....\\\\\\\.....
....\\\\\\\\\....
...\\\\\\\\\\\...
..\\\\\\\\\\\\\..
.\\\\\\\\\\\\\\\.
\\\\\\\\\\\\\\\\\
```

```
cout << rt.draw('+', '-') << endl;
```

```
+---------
++--------
+++-------
++++------
+++++-----
++++++----
+++++++---
++++++++--
+++++++++-
++++++++++
```

Clearly, the draw and output operations above should also be wrapped into a member function of **Shape**, say, **draw_on_screen**, that writes the image of the invoking object on the screen:

```
ace.draw_on_screen(' ', 'o');
```

```
oooooooo oooooooo
ooooooo   ooooooo
oooooo     oooooo
ooooo       ooooo
oooo         oooo
ooo           ooo
oo             oo
o               o

o               o
oo             oo
ooo           ooo
oooo         oooo
ooooo       ooooo
oooooo     oooooo
ooooooo   ooooooo
oooooooo oooooooo
```

or into a member function **write_image_to_stream** that writes the image of the invoking object to a given **ostream&**:

```
ace.scale(-4);
ace.write_image_to_stream(cout, '1');
```

```
    1
   111
  11111
 1111111
111111111
 1111111
  11111
   111
    1
```

```
ace.scale(2);
ace.write_image_to_stream(cout, 'A', '.');
```

```
......A......
.....AAA.....
....AAAAA....
...AAAAAAA...
..AAAAAAAAA..
.AAAAAAAAAAA.
AAAAAAAAAAAAA
.AAAAAAAAAAA.
..AAAAAAAAA..
...AAAAAAA...
....AAAAA....
.....AAA.....
......A......
```

Task 2 of 2

Design and implement a class to model a simple slot machine, using the geometric shapes you created above as visual symbols. This slot machine has three reels, each with 4 symbols, and each symbol in 25 available sizes. Thus each reel can display a total of 100 distinct shapes.

Here is a sample run of our slot machine:

```
int main()
{
   // create a slot machine object
   SlotMachine slot_machine;
   // run the slot machine until the player decides to stop,
   // or until the player runs out of tokens
   slot_machine.run();
   return 0;
}
```

```
Welcome to this 3-Reel Slot Machine Game!
Each reel will randomly display one of four shapes, each in 25 sizes.
To win 3 times your bet you need 3 similar shapes of the same size.
To win 2 times your bet you need 3 similar shapes.
To win or lose nothing you need 2 similar shapes.
Otherwise, you lose your bet.
You start with 10 free tokens!

How much would you like to bet (enter 0 to quit)? 3
+----+-----------------+---------+
| *  |        *        |    *    |
| ** |       ***       |   ***   |
|    |      *****      |  *****  |
|    |     *******     | ******* |
|    |    *********    |  *****  |
|    |   ***********   |   ***   |
|    |  *************  |    *    |
|    | *************** |         |
+----+-----------------+---------+
(Right Triangle, 2, 2) (Isosceles, 15, 8) (Rhombus, 7, 7)
You lose your bet
You now have 7 tokens!

How much would you like to bet (enter 0 to quit)? 2
```

```
+-----+---+-----------------------+
|  *  | * | ********************* |
| *** |   | ********************* |
|  *  |   | ********************* |
|     |   | ********************* |
|     |   | ********************* |
|     |   | ********************* |
|     |   | ********************* |
|     |   | ********************* |
+-----+---+-----------------------+
(Rhombus, 3, 3) (Isosceles, 1, 1) (Rectangle, 21, 8)
You lose your bet
You now have 5 tokens!

How much would you like to bet (enter 0 to quit)?
```

Internally, the **slot_machine** object maintains an array of three pointers, each pointing to a newly created concrete **shape** object. To create its own visual representation, object **slot_machine** first invokes **shape**'s member function **draw** on each of the three **shape** objects and then displays them, placing the resulting image girds side by side vertically as shown above. To make the output look a bit nicer, **slot_machine** decorates the grids as shown, separating them from decoration by one space on all four sides. For simplicity, the image grids are top aligned.

In its **run()** function, **slot_machine** repeatedly performs the following algorithm until the user runs out of tokens or decides to stop playing:

1. Prompt for and read a bet
2. For each reel r, $r = 0, 1, 2$
   2.1. let reel r point to a newly created shape object of random type and random size
3. Display the reels[1]
4. Report outcome, payout, and tokens left
5. Free dynamic memory consumed by the reels

Step 2.1 expands to these steps:

(a) generate a random integer n, $0 \le n \le 3$
(b) generate a random width w, $1 \le w \le 25$
(c) if $n = 0$ then let reel r point to a **Rhombus** object of width w
(d) if $n = 1$ then let reel r point to a **Isosceles** object of width w
(e) if $n = 2$ then let reel r point to a **RightTriangle** object of width w
(f) if $n = 3$ then
   i. generate a random height h, $1 \le h \le 25$
   ii. let reel r point to a **Rectangle** object of width w and height h

Use an **array** container to represent the reels in the algorithm:

[1]This part gives you practice with using variable-sized two dimensional **vector<vector<T>>** arrays, and indirectly with fixed $m \times n$ two dimensional **array<array<T, n>, m>** of **T** objects.

```
class SlotMachine
{
private:
   std::array<Shape*, 3> shape_reel{}; // an array of 3 pointers to Shape

   // implemets step 2
   void make_shapes(); // makes shape reels point at newly created dynamic shape objects

      // implemets step 2.1
   void make_shape(int r); // makes shape_reel[r] point at a newly created dynamic shape object

   // implements step 3
   void display_shapes(); // displays the shape reels

   // implements step 4
   void report_status(); // displays outcome, payout, and tokens left

   // implements step 5
   void release_shapes(); // frees dynamic objects currently pointed at by the shape reels
public:
   // enable default constructor
   SlotMachine() = default;
   // disable copy constructor and assignment
   SlotMachine(const ShapeSlotMachine&) = delete;
   SlotMachine& operator=(const ShapeSlotMachine&) = delete;
   void run(); // implements the algorithm descibed above
   virtual ~SlotMachine(); // frees dynamic objects currently pointed at by the shape reels
};
```

Note that in a GUI environment, we would implement the concepts of a slot machine reel into a class **Reel** whose objects are each responsible for managing their own internal needs, including the use of dynamic memory.

Deliverables

Header files:	Shape.h, Triangle.h, Rectangle.h, Rhombus.h, Triangle.h, Isosceles.h, RightTriangle.h
Implementation files:	Shape.cpp, Triangle.cpp, Rectangle.cpp, Rhombus.cpp, Triangle.cpp, Isosceles.cpp, RightTriangle.cpp, and **shapes_driver.cpp**
README.txt	A text file (see the course outline).

Marking scheme

55%	Implementation of the Shape class Hierarchy Shape, Rectangle, Rhombus, Triangle, Isosceles, RightTriangle
25%	Slot Machine
10%	Format, clarity, completeness of output
10%	Concise documentation of nontrivial steps in code, choice of variable names, indentation and readability of program

```
Welcome to this 3-Reel Slot Machine Game!
Each reel will randomly display one of four shapes, each in 25 sizes.
To win 3 times your bet you need 3 similar shapes of the same size.
To win 2 times your bet you need 3 similar shapes.
To win or lose nothing you need 2 similar shapes.
Otherwise, you lose your bet.
You start with 10 free tokens!

How much would you like to bet (enter 0 to quit)? 3
+-------+-------------+-----------------+
|   *   |      *      | *************** |
|  ***  |     ***     | *************** |
| ***** |    *****    | *************** |
|       |   *******   | *************** |
|       |  *********  | *************** |
|       | *********** | *************** |
|       |  *********  | *************** |
|       |   *******   | *************** |
|       |    *****    | *************** |
|       |     ***     | *************** |
|       |      *      | *************** |
|       |             | *************** |
|       |             | *************** |
+-------+-------------+-----------------+
(Isosceles, 5, 3) (Rhombus, 11, 11) (Rectangle, 15, 13)
You lose your bet
You now have 7 tokens!

How much would you like to bet (enter 0 to quit)? 3
+---------------+------------+---------------------+
|      *        | ********** |          *          |
|     ***       |            |         ***         |
|    *****      |            |        *****        |
|   *******     |            |       *******       |
|  *********    |            |      *********      |
| ***********   |            |     ***********     |
|               |            |    *************    |
|               |            |   ***************   |
|               |            |  *****************  |
|               |            | ******************* |
|               |            |  *****************  |
|               |            |   ***************   |
|               |            |    *************    |
|               |            |     ***********     |
|               |            |      *********      |
|               |            |       *******       |
|               |            |        *****        |
|               |            |         ***         |
|               |            |          *          |
+---------------+------------+---------------------+
(Isosceles, 13, 6) (Rectangle, 10, 1) (Rhombus, 19, 19)
You lose your bet
You now have 4 tokens!
```

```

How much would you like to bet (enter 0 to quit)? 1
+-----------------+---------------+---------+
|        *        |       *       |    *    |
|       ***       |      ***      |   ***   |
|      *****      |     *****     |  *****  |
|     *******     |    *******    | ******* |
|    *********    |   *********   |  *****  |
|   ***********   |  ***********  |   ***   |
|  *************  | ************* |    *    |
| *************** |  ***********  |         |
|                 |   *********   |         |
|                 |    *******    |         |
|                 |     *****     |         |
|                 |      ***      |         |
|                 |       *       |         |
+-----------------+---------------+---------+
(Isosceles, 15, 8) (Rhombus, 13, 13) (Rhombus, 7, 7)
You don't win, you don't lose, your are safe!
You now have 4 tokens!

How much would you like to bet (enter 0 to quit)? 1
+---------------+-----------+---------+
|      *        | *         | *       |
|     ***       | **        | **      |
|    *****      | ***       | ***     |
|   *******     | ****      | ****    |
|  *********    | *****     | *****   |
| ***********   | ******    | ******  |
|               | *******   | ******* |
|               | ********  |         |
|               | ********* |         |
+---------------+-----------+---------+
(Isosceles, 13, 6) (Right Triangle, 9, 9) (Right Triangle, 7, 7)
You don't win, you don't lose, your are safe!
You now have 4 tokens!

How much would you like to bet (enter 0 to quit)? 1
```

```
+---------------+-------------------+-------+
| ************* |         *         |   *   |
| ************* |        ***        |  ***  |
| ************* |       *****       | ***** |
| ************* |      *******      |  ***  |
| ************* |     *********     |   *   |
| ************* |    ***********    |       |
| ************* |   *************   |       |
| ************* |  ***************  |       |
| ************* | ***************** |       |
| ************* |  ***************  |       |
| ************* |   *************   |       |
| ************* |    ***********    |       |
| ************* |     *********     |       |
| ************* |      *******      |       |
| ************* |       *****       |       |
| ************* |        ***        |       |
| ************* |         *         |       |
| ************* |                   |       |
+---------------+-------------------+-------+
(Rectangle, 13, 18) (Rhombus, 17, 17) (Rhombus, 5, 5)
You don't win, you don't lose, your are safe!
You now have 4 tokens!

How much would you like to bet (enter 0 to quit)? 1
+---+---------+-----+
| * |   *     |  *  |
|   |  ***    | *** |
|   | *****   |  *  |
+---+---------+-----+
(Rhombus, 1, 1) (Isosceles, 7, 3) (Rhombus, 3, 3)
You don't win, you don't lose, your are safe!
You now have 4 tokens!

How much would you like to bet (enter 0 to quit)? 1
+-----------------------+-------+-------------------+
| *                     |  *    | ****************  |
| **                    | ***   |                   |
| ***                   |       |                   |
| ****                  |       |                   |
| *****                 |       |                   |
| ******                |       |                   |
| *******               |       |                   |
| ********              |       |                   |
| *********             |       |                   |
| **********            |       |                   |
| ***********           |       |                   |
| ************          |       |                   |
| *************         |       |                   |
| **************        |       |                   |
| ***************       |       |                   |
| ****************      |       |                   |
| *****************     |       |                   |
| ******************    |       |                   |
| *******************   |       |                   |
| ********************  |       |                   |
| ********************* |       |                   |
+-----------------------+-------+-------------------+
```

```
(Right Triangle, 21, 21) (Isosceles, 5, 2) (Rectangle, 17, 1)
You lose your bet
You now have 3 tokens!

How much would you like to bet (enter 0 to quit)? 1
+-------------------------+-----------------------+-------------------+
|            *            |          *            |        *          |
|           ***           |         ***           |       ***         |
|          *****          |        *****          |      *****        |
|         *******         |       *******         |     *******       |
|        *********        |      *********        |    *********      |
|       ***********       |     ***********       |   ***********     |
|      *************      |    *************      |  *************    |
|     ***************     |   ***************     | ***************   |
|    *****************    |  *****************    |                   |
|   *******************   | *******************   |                   |
|  *********************  |                       |                   |
| *********************** |                       |                   |
+-------------------------+-----------------------+-------------------+
(Isosceles, 23, 12) (Isosceles, 21, 10) (Isosceles, 17, 8)
Congratulations! you win 2 times your bet: 2
You now have 5 tokens!

How much would you like to bet (enter 0 to quit)? 1
+-----------------+---------------------+-------------------------+
|        *        |          *          |            *            |
|       ***       |         ***         |           ***           |
|      *****      |        *****        |          *****          |
|     *******     |       *******       |         *******         |
|    *********    |      *********      |        *********        |
|   ***********   |     ***********     |       ***********       |
|  *************  |    *************    |      *************      |
| *************** |   ***************   |     ***************     |
|  *************  |  *****************  |    *****************    |
|   ***********   | ******************* |   *******************   |
|    *********    |  *****************  |  *********************  |
|     *******     |   ***************   | *********************** |
|      *****      |    *************    |  *********************  |
|       ***       |     ***********     |   *******************   |
|        *        |      *********      |    *****************    |
|                 |       *******       |     ***************     |
|                 |        *****        |      *************      |
|                 |         ***         |       ***********       |
|                 |          *          |        *********        |
|                 |                     |         *******         |
|                 |                     |          *****          |
|                 |                     |           ***           |
|                 |                     |            *            |
+-----------------+---------------------+-------------------------+
(Rhombus, 15, 15) (Rhombus, 19, 19) (Rhombus, 23, 23)
Congratulations! you win 2 times your bet: 2
You now have 7 tokens!

How much would you like to bet (enter 0 to quit)? 1
```

```
+----------------+------------------------+-------+
| *              | *                      | ***** |
| **             | **                     | ***** |
| ***            | ***                    | ***** |
| ****           | ****                   | ***** |
| *****          | *****                  | ***** |
| ******         | ******                 | ***** |
| *******        | *******                | ***** |
| ********       | ********               | ***** |
| *********      | *********              | ***** |
| **********     | **********             | ***** |
| ***********    | ***********            | ***** |
| ************   | ************           | ***** |
| *************  | *************          |       |
| ************** | **************         |       |
|                | ***************        |       |
|                | ****************       |       |
|                | *****************      |       |
|                | ******************     |       |
|                | *******************    |       |
|                | ********************   |       |
|                | *********************  |       |
|                | ********************** |       |
+----------------+------------------------+-------+
(Right Triangle, 14, 14) (Right Triangle, 22, 22) (Rectangle, 5, 12)
You don't win, you don't lose, your are safe!
You now have 7 tokens!

How much would you like to bet (enter 0 to quit)? 1
+-------------------------+---+---------+
|            *            | * |    *    |
|           ***           |   |   ***   |
|          *****          |   |  *****  |
|         *******         |   | ******* |
|        *********        |   |         |
|       ***********       |   |         |
|      *************      |   |         |
|     ***************     |   |         |
|    *****************    |   |         |
|   *******************   |   |         |
|  *********************  |   |         |
| *********************** |   |         |
|  *********************  |   |         |
|   *******************   |   |         |
|    *****************    |   |         |
|     ***************     |   |         |
|      *************      |   |         |
|       ***********       |   |         |
|        *********        |   |         |
|         *******         |   |         |
|          *****          |   |         |
|           ***           |   |         |
|            *            |   |         |
+-------------------------+---+---------+
(Rhombus, 23, 23) (Rhombus, 1, 1) (Isosceles, 7, 4)
You don't win, you don't lose, your are safe!
You now have 7 tokens!
```

```

How much would you like to bet (enter 0 to quit)? 1
+--------+---------------------------+--------------+
| ****** | *                         | ************ |
| ****** | **                        | ************ |
| ****** | ***                       | ************ |
| ****** | ****                      | ************ |
| ****** | *****                     | ************ |
| ****** | ******                    |              |
| ****** | *******                   |              |
| ****** | ********                  |              |
| ****** | *********                 |              |
| ****** | **********                |              |
| ****** | ***********               |              |
| ****** | ************              |              |
| ****** | *************             |              |
|        | **************            |              |
|        | ***************           |              |
|        | ****************          |              |
|        | *****************         |              |
|        | ******************        |              |
|        | *******************       |              |
|        | ********************      |              |
|        | *********************     |              |
|        | **********************    |              |
|        | ***********************   |              |
|        | ************************  |              |
|        | ************************* |              |
+--------+---------------------------+--------------+
(Rectangle, 6, 13) (Right Triangle, 25, 25) (Rectangle, 12, 5)
You don't win, you don't lose, your are safe!
You now have 7 tokens!

How much would you like to bet (enter 0 to quit)? 1
+--------------+-------------+-----+
| ************ | *********** | *** |
| ************ | *********** | *** |
| ************ | *********** | *** |
| ************ | *********** | *** |
|              | *********** | *** |
|              | *********** | *** |
|              | *********** | *** |
|              | *********** | *** |
|              | *********** | *** |
|              |             | *** |
|              |             | *** |
|              |             | *** |
|              |             | *** |
|              |             | *** |
+--------------+-------------+-----+
(Rectangle, 12, 4) (Rectangle, 11, 9) (Rectangle, 3, 14)
Congratulations! you win 2 times your bet: 2
You now have 9 tokens!

How much would you like to bet (enter 0 to quit)? -2

How much would you like to bet (enter 0 to quit)? 0
Game Over. You now have 9 tokens!
```