Objectives

To practice fundamental object-oriented programming (OOP) concepts such as encap-sulation, inheritance, and polymorphism

To learn how to dene an inheritance hierarchy of classes implementing a common interface

To learn how to dene 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 specic 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. apointerorareferenceofabaseclasstype, whichisusedtoinvokethevirtualfunctions.

To build an inheritance hierarchy that models the shapes used in this assignments, we rst 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

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 identiable 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

2

4. Rhombus shapes with both equal and odd diagonal length d 1

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

Rectangle shapes

Construction values:

Sample image:

How to scale(n)

width w 1 and height h 1

a rectangle with w = 9 and h = 5

set w w+n and h h+n, provided that both w + n 1 and h + n 1; otherwise, no scale.

Sample Image \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

Isosceles triangles with odd base b and height h = (b+1)=2 Sample Image

Construction value:

Sample image:

How to scale(n)

base b 1 and odd

An isosceles triangle, b = 9 and h = 5

if b + 2n 1, set b b + 2n and h (b+1)=2, in that order; otherwise, no scale.

\* \*\*\*

\*\*\*\*\* \*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*

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

Construction value:

Sample image:

How to scale(n)

base b 1

a right triangle with b = 5.

Set both b and h to b+n, provided that b+n 1; otherwise, no scale.

\* \*\* \*\*\*

\*\*\*\* \*\*\*\*\*

Rhombus shapes with both equal and odd diagonal length d 1 Sample Image

Construction value:

Sample image:

How to scale(n)

diagonal d 1 and odd

a rhombus with d = 5

if d+2n 1 set d d+2n; otherwise, no scale.

\* \*\*\*

\*\*\*\*\* \*\*\* \*

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 specics of the concrete shapes above are specied in the following table.

3

Concrete Shape Specics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Rectangle | Rhombus | Right Triangle | Isosceles Triangle |
| construction values | h;w | d, if d is even set d d+1 | b | b, if b is even set b 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 | d2=2 | hb=2 | hb=2 |
| Screen Area | hw | 2n(n+1)+1, n = bd=2c | h(h+1)=2 | h2 |
| geometric perimeter | 2(h+w) | (2p2)d | (2+ p2)h | b+2p0:25b2 +h2 |
| 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.

Shape

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | | | | |
|  |  | | | |  | | |  |
| Triangle | |  | | Rectangle | |  | Rhombus | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | |  | | |
|  |  | | | |  |
| IsoTriangle | |  | | RightTriangle | |

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:

4

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 dened. 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.

1 Rectangle shape1(10, 15); 2 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 rst 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:

5

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.

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

6

20 RightTriangle rt(10, "Carpenter’s square"); 21 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.

0 1 2 3 4 The image grid is a bounding box representation for the invoking shape. 0

The grid rows are parallel to the x-axis, with row numbers increasing 1 down. The grid columns are parallel to the y-axis, with column numbers 2

increasing to the right. The origin of the grid is located at the top-left 3 grid cell (0;0) at row 0 and column 0. 4 5

x

y

The draw function uses the supplied pen character penChar to render the shape image. Any cell not on the image is lled with the supplied ll character llChar. As indicated in the function prototype above, penChar defaults to ‘’ and llChar to the blank character.

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

7

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:

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

23 cout << ace.draw(’o’) << endl;

\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*

o ooo

ooooo ooooooo ooooooooo ooooooooooo

ooooooooooooo ooooooooooooooo ooooooooooooooooo ooooooooooooooo ooooooooooooo ooooooooooo ooooooooo ooooooo ooooo ooo o

8

24 cout << iso.draw(’\\’, ’.’) << endl;

25 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:

oooooooo ooooooo oooooo ooooo oooo

ooo oo o

26 ace.draw\_on\_screen(’ ’, ’o’);

o oo ooo

oooo ooooo oooooo ooooooo oooooooo

oooooooo ooooooo oooooo ooooo oooo

ooo oo o

o oo ooo

oooo ooooo oooooo ooooooo oooooooo

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

9

27 ace.scale(-4);

28 ace.write\_image\_to\_stream(cout, ’1’);

29 ace.scale(2);

30 ace.write\_image\_to\_stream(cout, ’A’, ’.’);

1 111

11111 1111111 111111111 1111111 11111 111 1

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

10

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:

1 int main() 2 {

3 // create a slot machine object 4 SlotMachine slot\_machine;

5 // run the slot machine until the player decides to stop, 6 // or until the player runs out of tokens

7 slot\_machine.run(); 8 return 0;

9 }

1 Welcome to this 3-Reel Slot Machine Game!

2 Each reel will randomly display one of four shapes, each 3 To win 3 times your bet you need 3 similar shapes of the 4 To win 2 times your bet you need 3 similar shapes.

5 To win or lose nothing you need 2 similar shapes. 6 Otherwise , you lose your bet.

7 You start with 10 free tokens!

8

in 25 sizes. same size.

9 How much would you like to bet (enter 0 to quit)? 3 10 +----+-----------------+---------+

11 | \* | \* | \* | 12 | \*\* | \*\*\* | \*\*\* | 13 | | \*\*\*\*\* | \*\*\*\*\* | 14 | | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | 15 | | \*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 16 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 17 | | \*\*\*\*\*\*\*\*\*\*\*\*\* | \* | 18 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 19 +----+-----------------+---------+

20 (Right Triangle, 2, 2) (Isosceles , 15, 8) (Rhombus, 7, 7) 21 You lose your bet

22 You now have 7 tokens!

23

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

11

25 +-----+---+-----------------------+ 26 | \* | \* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 27 | \*\*\* | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 28 | \* | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 29 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 30 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 31 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 32 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 33 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 34 +-----+---+-----------------------+

35 (Rhombus, 3, 3) (Isosceles , 1, 1) (Rectangle , 21, 8) 36 You lose your bet

37 You now have 5 tokens!

38

39 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 rst 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 reels1

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 n 3 (b) generate a random width w, 1 w 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 h 25

ii. let reel r point to a Rectangle object of width w and height h

Use an [array](http://www.cplusplus.com/reference/array/array/) container to represent the reels in the algorithm:

1This part gives you practice with using variable-sized two dimensional vector<vector<T>> arrays, and indirectly with xed mn two dimensional array<array<T, n>, m> of T objects.

12

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.

13

1 Welcome to this 3-Reel Slot Machine Game!

2 Each reel will randomly display one of four shapes, each 3 To win 3 times your bet you need 3 similar shapes of the 4 To win 2 times your bet you need 3 similar shapes.

5 To win or lose nothing you need 2 similar shapes. 6 Otherwise , you lose your bet.

7 You start with 10 free tokens!

8

in 25 sizes. same size.

9 How much would you like to bet (enter 0 to quit)? 3 10 +-------+-------------+-----------------+

11 | \* | \* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 12 | \*\*\* | \*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 13 | \*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 14 | | \*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 15 | | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 16 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 17 | | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 18 | | \*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 19 | | \*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 20 | | \*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 21 | | \* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 22 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 23 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 24 +-------+-------------+-----------------+

25 (Isosceles , 5, 3) (Rhombus, 11, 11) (Rectangle , 15, 13) 26 You lose your bet

27 You now have 7 tokens!

28

29 How much would you like to bet (enter 0 to quit)? 3 30 +---------------+------------+---------------------+ 31 | \* | \*\*\*\*\*\*\*\*\*\* | \* | 32 | \*\*\* | | \*\*\* | 33 | \*\*\*\*\* | | \*\*\*\*\* | 34 | \*\*\*\*\*\*\* | | \*\*\*\*\*\*\* | 35 | \*\*\*\*\*\*\*\*\* | | \*\*\*\*\*\*\*\*\* | 36 | \*\*\*\*\*\*\*\*\*\*\* | | \*\*\*\*\*\*\*\*\*\*\* | 37 | | | \*\*\*\*\*\*\*\*\*\*\*\*\* | 38 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 39 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 40 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 41 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 42 | | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 43 | | | \*\*\*\*\*\*\*\*\*\*\*\*\* | 44 | | | \*\*\*\*\*\*\*\*\*\*\* | 45 | | | \*\*\*\*\*\*\*\*\* | 46 | | | \*\*\*\*\*\*\* | 47 | | | \*\*\*\*\* | 48 | | | \*\*\* | 49 | | | \* | 50 +---------------+------------+---------------------+

51 (Isosceles , 13, 6) (Rectangle , 10, 1) (Rhombus, 19, 19) 52 You lose your bet

53 You now have 4 tokens!

15

54

55 How much would you like to bet (enter 0 to quit)? 1 56 +-----------------+---------------+---------+

57 | \* | \* | \* | 58 | \*\*\* | \*\*\* | \*\*\* | 59 | \*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 60 | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | 61 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 62 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 63 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | \* | 64 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | | 65 | | \*\*\*\*\*\*\*\*\* | | 66 | | \*\*\*\*\*\*\* | | 67 | | \*\*\*\*\* | | 68 | | \*\*\* | | 69 | | \* | | 70 +-----------------+---------------+---------+

71 (Isosceles , 15, 8) (Rhombus, 13, 13) (Rhombus, 7, 7) 72 You don’t win, you don’t lose, your are safe!

73 You now have 4 tokens!

74

75 How much would you like to bet (enter 0 to quit)? 1 76 +---------------+-----------+---------+

77 | \* | \* | \* | 78 | \*\*\* | \*\* | \*\* | 79 | \*\*\*\*\* | \*\*\* | \*\*\* | 80 | \*\*\*\*\*\*\* | \*\*\*\* | \*\*\*\* | 81 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 82 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\* | \*\*\*\*\*\* | 83 | | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | 84 | | \*\*\*\*\*\*\*\* | | 85 | | \*\*\*\*\*\*\*\*\* | | 86 +---------------+-----------+---------+

87 (Isosceles , 13, 6) (Right Triangle, 9, 9) (Right Triangle, 7, 7) 88 You don’t win, you don’t lose, your are safe!

89 You now have 4 tokens!

90

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

16

92 +---------------+-------------------+-------+ 93 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \* | \* | 94 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\* | \*\*\* | 95 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 96 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\* | \*\*\* | 97 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | \* | 98 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | | 99 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | | 100 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 101 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 102 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 103 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | | 104 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | | 105 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | | 106 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\* | | 107 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\* | | 108 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\* | | 109 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \* | | 110 | \*\*\*\*\*\*\*\*\*\*\*\*\* | | | 111 +---------------+-------------------+-------+

112 (Rectangle , 13, 18) (Rhombus, 17, 17) (Rhombus, 5, 5) 113 You don’t win, you don’t lose, your are safe!

114 You now have 4 tokens!

115

116 How much would you like to bet 117 +---+---------+-----+

118 | \* | \* | \* | 119 | | \*\*\* | \*\*\* | 120 | | \*\*\*\*\* | \* | 121 +---+---------+-----+

122 (Rhombus, 1, 1) (Isosceles , 7, 123 You don’t win, you don’t lose, 124 You now have 4 tokens!

125

(enter 0 to quit)? 1

3) (Rhombus, 3, 3) your are safe!

126 How much would you like to bet (enter 0 to quit)? 1 127 +-----------------------+-------+-------------------+ 128 | \* | \* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 129 | \*\* | \*\*\* | | 130 | \*\*\* | | | 131 | \*\*\*\* | | | 132 | \*\*\*\*\* | | | 133 | \*\*\*\*\*\* | | | 134 | \*\*\*\*\*\*\* | | | 135 | \*\*\*\*\*\*\*\* | | | 136 | \*\*\*\*\*\*\*\*\* | | | 137 | \*\*\*\*\*\*\*\*\*\* | | | 138 | \*\*\*\*\*\*\*\*\*\*\* | | | 139 | \*\*\*\*\*\*\*\*\*\*\*\* | | | 140 | \*\*\*\*\*\*\*\*\*\*\*\*\* | | | 141 | \*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 142 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 143 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 144 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 145 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 146 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | |

147 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 17 | 148 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | |

149 +-----------------------+-------+-------------------+

150 (Right Triangle, 21, 21) (Isosceles , 5, 2) (Rectangle , 17, 1) 151 You lose your bet

152 You now have 3 tokens!

153

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

155 +-------------------------+-----------------------+-------------------+ 156 | \* | \* | \* | 157 | \*\*\* | \*\*\* | \*\*\* | 158 | \*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 159 | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | 160 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | 161 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | 162 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | 163 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 164 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 165 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 166 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 167 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 168 +-------------------------+-----------------------+-------------------+ 169 (Isosceles , 23, 12) (Isosceles , 21, 10) (Isosceles , 17, 8)

170 Congratulations! you win 2 times your bet: 2 171 You now have 5 tokens!

172

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

174 +-----------------+---------------------+-------------------------+ 175 | \* | \* | \* | 176 | \*\*\* | \*\*\* | \*\*\* | 177 | \*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 178 | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | 179 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | 180 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | 181 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | 182 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 183 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 184 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 185 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 186 | \*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 187 | \*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 188 | \*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 189 | \* | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 190 | | \*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 191 | | \*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | 192 | | \*\*\* | \*\*\*\*\*\*\*\*\*\*\* | 193 | | \* | \*\*\*\*\*\*\*\*\* | 194 | | | \*\*\*\*\*\*\* | 195 | | | \*\*\*\*\* | 196 | | | \*\*\* | 197 | | | \* | 198 +-----------------+---------------------+-------------------------+ 199 (Rhombus, 15, 15) (Rhombus, 19, 19) (Rhombus, 23, 23)

200 Congratulations! you win 2 times your bet: 2 201 You now have 7 tokens!

202

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

18

204 +----------------+------------------------+-------+ 205 | \* | \* | \*\*\*\*\* | 206 | \*\* | \*\* | \*\*\*\*\* | 207 | \*\*\* | \*\*\* | \*\*\*\*\* | 208 | \*\*\*\* | \*\*\*\* | \*\*\*\*\* | 209 | \*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\* | 210 | \*\*\*\*\*\* | \*\*\*\*\*\* | \*\*\*\*\* | 211 | \*\*\*\*\*\*\* | \*\*\*\*\*\*\* | \*\*\*\*\* | 212 | \*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\* | \*\*\*\*\* | 213 | \*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 214 | \*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 215 | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 216 | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\* | 217 | \*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | | 218 | \*\*\*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 219 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 220 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 221 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 222 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 223 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 224 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 225 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 226 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 227 +----------------+------------------------+-------+

9

228 (Right Triangle, 14, 14) (Right Triangle, 22, 22) (Rectangle , 5, 12) 229 You don’t win, you don’t lose, your are safe!

230 You now have 7 tokens!

231

232 How much would you like to bet (enter 0 to quit)? 1 233 +-------------------------+---+---------+

234 | \* | \* | \* | 235 | \*\*\* | | \*\*\* | 236 | \*\*\*\*\* | | \*\*\*\*\* | 237 | \*\*\*\*\*\*\* | | \*\*\*\*\*\*\* | 238 | \*\*\*\*\*\*\*\*\* | | | 239 | \*\*\*\*\*\*\*\*\*\*\* | | | 240 | \*\*\*\*\*\*\*\*\*\*\*\*\* | | | 241 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 242 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 243 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 244 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 245 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 246 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 247 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 248 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 249 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | | 250 | \*\*\*\*\*\*\*\*\*\*\*\*\* | | | 251 | \*\*\*\*\*\*\*\*\*\*\* | | | 252 | \*\*\*\*\*\*\*\*\* | | | 253 | \*\*\*\*\*\*\* | | | 254 | \*\*\*\*\* | | | 255 | \*\*\* | | | 256 | \* | | | 257 +-------------------------+---+---------+

258 (Rhombus, 23, 23) (Rhombus, 1, 1) (Isosceles , 7, 4)

259 You don’t win, you don’t lose, your 1are safe! 260 You now have 7 tokens!

261

262 How much would you like to bet (enter 0 to quit)? 1 263 +--------+---------------------------+--------------+ 264 | \*\*\*\*\*\* | \* | \*\*\*\*\*\*\*\*\*\*\*\* | 265 | \*\*\*\*\*\* | \*\* | \*\*\*\*\*\*\*\*\*\*\*\* | 266 | \*\*\*\*\*\* | \*\*\* | \*\*\*\*\*\*\*\*\*\*\*\* | 267 | \*\*\*\*\*\* | \*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\* | 268 | \*\*\*\*\*\* | \*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\* | 269 | \*\*\*\*\*\* | \*\*\*\*\*\* | | 270 | \*\*\*\*\*\* | \*\*\*\*\*\*\* | | 271 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\* | | 272 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\*\* | | 273 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\* | | 274 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | | 275 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\* | | 276 | \*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\*\*\* | | 277 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 278 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 279 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 280 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 281 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 282 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 283 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 284 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 285 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 286 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 287 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 288 | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | | 289 +--------+---------------------------+--------------+

290 (Rectangle , 6, 13) (Right Triangle, 25, 25) (Rectangle , 12, 5) 291 You don’t win, you don’t lose, your are safe!

292 You now have 7 tokens!

293

294 How much would you like to bet (enter 295 +--------------+-------------+-----+ 296 | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 297 | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 298 | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 299 | \*\*\*\*\*\*\*\*\*\*\*\* | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 300 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 301 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 302 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 303 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 304 | | \*\*\*\*\*\*\*\*\*\*\* | \*\*\* | 305 | | | \*\*\* | 306 | | | \*\*\* | 307 | | | \*\*\* | 308 | | | \*\*\* | 309 | | | \*\*\* | 310 +--------------+-------------+-----+ 311 (Rectangle , 12, 4) (Rectangle , 11, 9) 312 Congratulations! you win 2 times your 313 You now have 9 tokens!

314

0 to quit)? 1

(Rectangle , 3, 14) bet: 2

315 How much would you 316

317 How much would you 318 Game Over. You now

like to bet (enter 0 to

20

like to bet (enter 0 to

have 9 tokens!

quit)? -2

quit)? 0