

## The Shapes

Implement an abstract parent Shape class and its polymorphic children Circle, Rectangle, and Triangle. Shape is a 2D character array which requires the use of dynamic memory allocation, and its children are their shapes held character-by-character within that 2D array. Additionally, each shape is represented by its perimeter populated by ASCII characters within the range [48, 126] in order, and character choice from this range wraps around back to 48 when 127 is reached.

### Task 1

Define and implement the abstract class Shape, which contains the following methods:

```
Shape(const int &width, const int &height); // Parameterized Constructor
```

```
// Getters
```

```
int getEdges( );
```

```
int getWidth( );
```

```
int getHeight( );
```

```
char **getDisplayChars( );
```

```
// Setters
```

```
void setEdges(const int& edges);
```

```
void setWidth(const int& new_width);
```

```
void setHeight(const int &new_height);
```

```
void setDisplayChars(char **display);
```

```
// Mutators
```

```
void rotateRight( ); //rotate by 90 degrees
```

```
void rotateLeft( ); //rotate by 90 degrees
```

```
// Pure Virtual Methods (no implementation)
```

```
virtual double getSurfaceArea( ) = 0;
```

```
virtual double get3DVolume(const double& depth) = 0;
```

```
// Display : iterate through 2D array and print chars
```

```
void display();
```

### Note:

- reflect( ) : takes the parameter 'x' or 'y'.
- getSurfaceArea( ) : uses the formulae for surface area given each respective shape.
- get3DVolume( ) : yields the volume of the caller shape if it was projected into 3 dimensions using the depth parameter as the z-axis value. For a circle, this function gives the volume of the related sphere. For a rectangle, this function gives the volume of the related rectangular cuboid. For a triangle, this function gives the volume of the related right triangular prism, since Triangle only considers right triangles. Use the formulae for volume given each respective surface.

## Task 2

Define and implement a class Circle that inherits from Shape and implements its pure virtual functions. The Circle class must contain the following methods:

Circle(const int& diameter); //Parameterized constructor, takes the diameter as either width or height.

double getSurfaceArea();

double get3DVolume(const double& depth);

### Here is the constructor:

```
Circle::Circle(const int &diameter) : Shape(diameter, diameter)
{
    setEdges(0);

    // Populate 2D array with empty chars
    char **arr = new char *[getHeight()];
    for (int row = 0; row < getHeight(); row++){
        arr[row] = new char[getWidth()];
        for (int col = 0; col < getWidth(); col++){
            arr[row][col] = ' ';
        }
    }

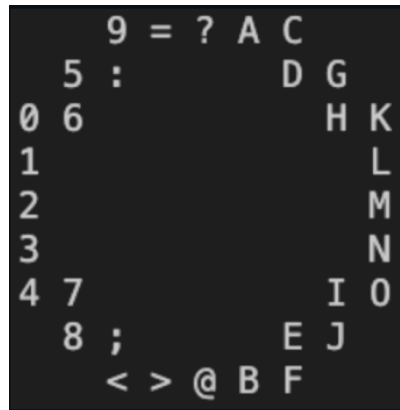
    // Populate the proper positions with *'s
    int x_radius = diameter / 2;
    int y_radius = (diameter / 2) - 1;
    float dist = 0;
    char ascii_counter = 48;

    for (int col = 0; col <= getWidth() + 1; col++){
        for (int row = 0; row <= getHeight() + 5; row++){
            dist = sqrt((row - y_radius) * (row - y_radius) + (col - x_radius) * (col - x_radius));

            // dist in range: (radius - 0.5) to (radius + 0.5)
            if (dist > y_radius - 0.5 && dist < y_radius + 0.5){
                arr[row][col] = ascii_counter;

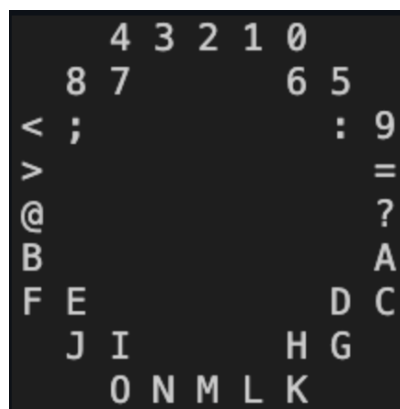
                // fix ascii_counter to wrap around after
                ascii_counter++;
                if (ascii_counter > 126){
                    ascii_counter = 48;
                }
            }
        }
    }
    setDisplayChars(arr);
}
```

\*What display( ) yields for a Circle of diameter 10:



A circle of diameter 10 with characters arranged in a circular pattern. The characters are: 9, =, ?, A, C, D, G, H, K, L, M, N, O, I, J, E, F, B, <, >, @, 8, 7, 6, 5, 4, 3, 2, 1, 0. The characters are arranged in a circular pattern, with 9 at the top and 0 at the bottom.

\*What display( ) yields after rotateRight( ) for this circle:



A circle of diameter 10 with characters arranged in a circular pattern after a right rotation. The characters are: 4, 3, 2, 1, 0, 8, 7, 6, 5, 9, =, ?, A, C, D, G, H, K, L, M, N, O, I, J, E, F, B, <, >, @. The characters are arranged in a circular pattern, with 4 at the top and 0 at the bottom.

\*What display( ) yields after rotateLeft( ) for this circle:



A circle of diameter 10 with characters arranged in a circular pattern after a left rotation. The characters are: K, L, M, N, O, G, H, I, J, E, F, B, <, >, @, 9, =, ?, A, C, D, 5, 6, 7, 8, 0, 1, 2, 3, 4. The characters are arranged in a circular pattern, with K at the top and 4 at the bottom.

### Task 3

## IMPLEMENT RECTANGLE

Define and implement a class 'Rectangle' that inherits from Shape and implements its pure virtual functions. The 'Rectangle' class must contain the following methods:

```
// Parameterized constructor; takes in width and height, iterates through the 2D
// array to populate it with the necessary characters given the parameter dimensions
Rectangle(const int& width, const int& height);

double getSurfaceArea();
double get3DVolume(const double &depth);
```

\*What `display()` yields for a Rectangle of dimensions 10x10:

```

0 1 2 3 4 5 6 7 8 9
:                               ;
<                               =
>                               ?
@                               A
B                               C
D                               E
F                               G
H                               I
J K L M N O P Q R S

```

\*What `display()` yields after `rotateRight()` for this rectangle:

J	H	F	D	B	@	>	<	:	0
K									1
L									2
M									3
N									4
O									5
P									6
Q									7
R									8
S	I	G	E	C	A	?	=	;	9

\*What `display()` yields after `rotateLeft()` for this rectangle:

[illegible]

## Task 4

### IMPLEMENT TRIANGLE

Define and implement a class Triangle that inherits from Shape and implements its pure virtual functions. The Triangle class must contain the following methods:

```
/* Parameterized constructor; takes in side length as a parameter, iterates
through the 2D array to draw the right triangle using ASCII chars */
```

```
Triangle(const int &side);
```

```
double getSurfaceArea();
```

```
double get3DVolume(const double &depth);
```

\*What display( ) yields for a Triangle of side length 10:

```
0
1 2
3 4
5 6
7 8
9 :
; <
= >
? @
A B C D E F G H I J
```

\*What display( ) yields after rotateRight( ) for this triangle:

```
A ? = ; 9 7 5 3 1 0
B
C
D
E
F
G
H
I @
J
```

\*What display( ) yields after rotateLeft( ) for this triangle:

```
0 1 3 5 7 9 ; = ? A
2
4
6
8
:
<
>
@
J
I
H
G
F
E
D
C
B
```

**Submit the following files:**

Shape.hpp, Circle.hpp, Triangle.hpp, & Rectangle.hpp

Shape.cpp, Circle.cpp, Triangle.cpp, & Rectangle.cpp