1. (4pts) The language should support configuration of the following

- (2pts) an entry point (the function that does the drawing)
- (2pts) the width, height, and the default stroke width of the figure

drawObject is the entry point to the program. There will be one drawObject method in each individual program of DrawLab. This function takes two parameters: Size which provides access to width, height which are defined as numbers. And stroke which are defined as numbers.

```
//Entry Point of the Program
drawObject(Size(600, 600), Stroke(5))
{
    draw(Line(true), Size(600, 600), Location(500, 500) );
}
```

2.(6pts) In addition to drawing with explicit sizes, drawing independent of scale should be supported. As an example of the latter, one should be able to draw 10 ovals next to each other horizontally, without specifying absolute values for the oval width and height.

To avoid the need for specifying absolute values for any shape; width, height and stroke. Initially, the values of these variables will be set as 200, 200 and 2 and the location point to start will be set to (0,0). The repetition of drawing will be by another function which is named as drawShapeNTimes. In this function, for loop which will be defined in this language is utilized.

```
// Drawing n numbers of specified object horizontally
drawObject(Size(600, 600), Stroke(5))
{
    drawShapeNTimes(Line(), Size(600, 600), Location(0, 0) ,n);
}


// Iterative function to iterate the number of specified shape to draw
drawShapeNTimes(drawLine(), Size(600, 600), Location(500, 500) ,n)
{
    for(int i = 0; i < n; i++)
    {
        drawShape(Line(), Size(600, 600), Location(i * 600, 0) );
    }
}
```

3.(5pts) The language should be dynamically typed.

- (2pts) It should support the following primitive types: float, integer, string, and boolean.
- (2pts) Basic arithmetic and logical operations should be supported on these types.
- (1pts) Strings should support concatenation via the + operator and conversion from other types.

In this programming language, types of integer, float, string, boolean will be supported. The basic arithmetic and logical operators like +(addition), -(substraction), *(multiplication), /(division), ^(exponent) operators and &(and), |(or), !(not) operators will be supported. Strings will be concatenated by + operator and if there is string and after the string there is +, every types of upcoming variable will be concatenated with string.

In this language, the type assignment of the values will be implicit; there is no need to define a variable's type as integer or float. According to value on the right side, the type of the variable which is on the left side of the equality operator will be determined. After first initialization of the variable, if wanted, variable can still be assigned to another type of variable or same type with different value.

```
//Operators, operations and types

// X is assigned to integer value one so X is now an integer value 1.
X =       1;


// X is assigned to float value one so X is now an float value 1.5.
X =       1.5;


// X is assigned to float value one so X is now an float value 0.5.
X =       .5;


// X is assigned to float value one so X is now an float value 1.0.
X =       1.;


// X is assigned to string value one so X is now a string value "DrawLab".
X =       "DrawLab";


// X is assigned to boolean value false so X is now a boolean value false
X =       false;


// Examples for different cases
// Ex1
x = 1;
y = "DrawLab";
y = y + x;          // y is now string value "DrawLab1"
```

```
// Ex2
x = "Hello";
y = "DrawLab:";
y = y + x;          // y is now string value "DrawLab:Hello"

// Ex3
x = "Hello:";
y = false;
x = x + y;          // x is now string value "Hello:false"

// Ex4
x = 4;
y = 2.2;
x = x + y;          // x is now float value 6.2
```

4.(3pts) The language should also support Location, Size, and Color types.

- (1pts) Location should provide access to two members: x and y.
- (1pts) Size should provide access to two members: width and height.
- (1pts) Color should provide a predefined set of constants, as well as an RGB constructor.

In this language, Location, Size and Color are individual types. Location takes two parameters; x and y. The user can define it by just entering these parameters. Size takes two parameters; width and height. The user can define it by just entering these parameters and there will be Color type which can be defined in two way. In the first way, there are colors whose RGB values is already determined:
BLACK, WHITE, GREEN, YELLOW, RED, ORANGE, BLUE.
The second way is to entering a specific RGB values to get the desired colour.

```
// Demonstration
// x and y parameters are in Location. First for x-axis, Second for y-axis
x = Location(100,100);

// Two parameters in Size. First for width, second for height
x = Size(100,100);

// The specified BLACK color assigned to x
x = Color(BLUE);

// The RGB of the color is set with there parameters Red, Green and Black.
x = Color(120,120,120);
```

5. (18pts) The language should provide a number of built-in shapes.
- (3pts) These shapes should include at least the following: Line, Rectangle, and Oval.
- (15pts) There should be a common syntax for instantiating shapes.
  - (5pts) This syntax should support parameters, including optional ones.
  - (5pts) A shape instance should be scale free, that is it should not reference a location or a size.
  - (5pts) You can assume that any given shape can be drawn inside a bounding box. Thus, that bounding box defines its location and size, which is specified later when the shape is drawn.
    - For instance, a Line should support specifying a direction (NE (north east), NW, SE, SW), arrows (start and end), arrow sizes (relative to line stroke width), etc. However, an Oval should not specify any parameters. A Rectangle can specify a parameter about rounded corners.

Language have shapes as a built in. This shapes are line, rectangle and Oval. They can instantiated by calling their methods (as Rectangle()) as it is stated in requirements they are all scale free. However, within the bounding box the figures can be drawn. When drawing rectangle putting number between 0 and 90 gives rounded corners with stated angle number When drawing line 1st give direction, 2nd make true when you want 1st point to be start and 2nd end, or make false when you want no arrows, 3rd choose 1 to 5 stroke size of line.

```
//drawing normal rectangle (without rounded corners)
x=Rectangle();

//drawing rectangle with rounded corners
x=Rectangle(54);

// drawing a line which lies along upper-left to bottom-right diagonal of its bounding box
// when it is drawn. If the argument is specified as FALSE, the line lies along bottom-left
x=Line(true);
// Oval should not specify any parameters
x=Oval();
```

6) (14pts)The language should support basic drawing statements.
- (10pts) These statements should support parameterization, which can be used to specify
  - (2pts) location, (2pts) size, (2pts) stroke width (relative), (2pts) fill state, and (2pts) fill color.
  - You can assume that some parameters are mandatory, some optional.
- (4pts) Similar to drawing shapes, drawing strings should be supported as well.

The language will support location with Location(x,y), size with Size(x,y), stroke width with Stroke(a) which is between 1 and 5, fill color with fillColor(red,green,black) or fillColor(color(SMTH)) where r g b are RGB values and color(SMTH) is witing the color you want that we have and get RGB values automatically and fill state makes decides whether or not fill the shape as fillState(false), it is automatically set to true when you don't want to fill just write false

```
//drawing normal rectangle with location in 0 x coordinate and 0 y coordinate
```

```
//stroke of rect
draw(Rectangle(),Location(0,0),Size(5,5),Stroke(2),fillColor(color(BLACK)),fillState(fa
lse);

//drawing oval with location of 50 (x) and 10 (y) size of 12 (x) and 23 (y) with Red
//color
draw(Oval(),Location(50,10),Size(12,23),fillColor(11,122,333))

//drawing string “CS315”, with location 50 50 and size of 22 22 and with color of //pale
blue
draw( drawString(“CS315”), location(50,50),size(22,22), fillColor(0,244,255))
```

7) (10pts) The language should support loops that can ease repetitive tasks.

Language will support 2 types of loops. These are for and while loops to ease repetitive tasks. for loops will get extra things other than compare where you can handle control variable.

```
//for loops
for(i=1; i<100; i=i+1)
{
      …
      //do smth
      …
}

//while loops
i=1;
while(i<100)
{
      …
      //do smth
      …
      i=i+1;
}
```

8) The language should support conditionals (if and if/else) as well.

Conditional statements are used in order to have proper programming language. They are supported by if and if/else statements.

```
//if statement
If (i>244){
      ….
      //do smth
      ….
}

// if/else statement
If(i>0) {
      ….
      //do smth
      ….
}
else{
```

```
	….
	//do smth
	….
}

//if/else if statements
If(i>0) {
	….
	//do smth
	….
}
else if (i=0){
	….
	//do smth
	….
}
else{
	….
	//do smth
	….
}
```

9) (10pts) The language should support modularity at the level of functions.

DrawLab allows you to use different functionalities under a defined function. Key name here is func for defining method. Name of the method is give after the func key word, and you can put some argument after the name in parenthesis or you can just keep it blank. Such as below:

function *name-of-it(argument1, …){*
….
}

Example of the function to find our value is zero or not stated as following:

```
function zeroOrNot(a){
		if(a=0)
			draw(drawString(“your value is 0”),Size(5,5), Location(100,0));
		else
			draw(drawString(“your value is not 0”),Size(5,5), Location(100,0));
	return;
}
```

10) (20pts) The language should support extension of shapes.

- (10pts) This extension should be performed via defining composite shapes
- (5pts) These shapes should be parameterizable (including optional and mandatory parameters).
- (5pts) Once a shape is defined, it should be possible to draw it as usual using the shape drawing functions.

Anyone can make shapes of her own desire in DrawLab anytime she wants. IT is relatively easy to draw any shape with the built-in shapes that we have in language. Also after creating new shape user can use his own shapes to create new shapes for herself.

```
newObject Trapezium {
a1=Line(false);
a2=Line(true);
a3=Line(true);
a4=Line(true);

draw(a1,Size(50,50),Location(0,0));
draw(a2,Size(80,0),Location(50,50));
draw(a3,Size(50,50),Location(130,50));
draw(a4,Size(180,0),Location(0,0));
}
```