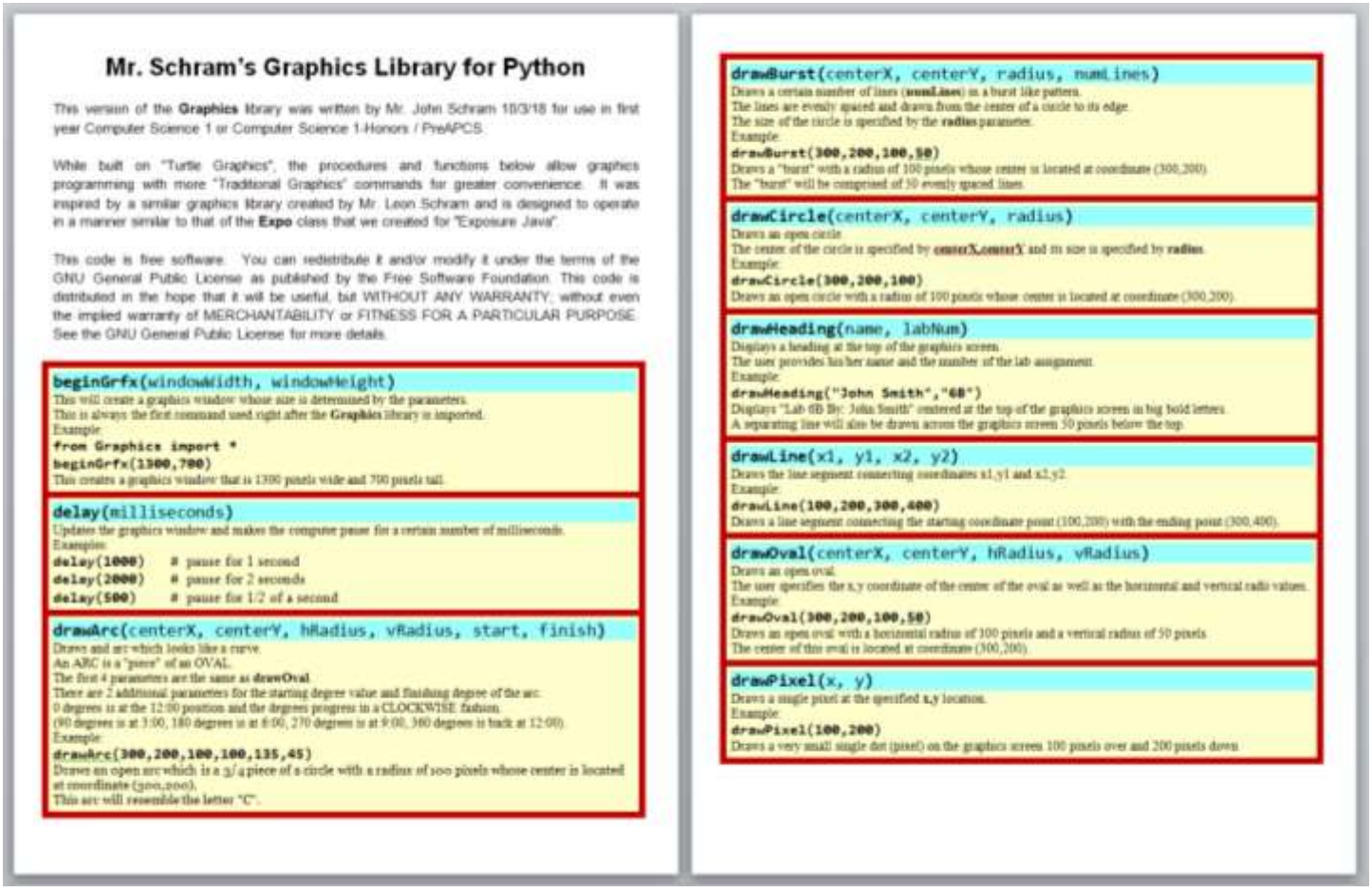
Computer Science 1	Lab 06B Practice/Perform Major Python Assignment
The Graphics Library Program	50, 60, 70, 80, 90, 100 & 110 Point Versions

Assignment Purpose:

The purpose of this program is to demonstrate knowledge of calling, and using correct argument passing with several of the procedures from the **Graphics** library.

Write a program, which displays several geometric designs using the provided **Graphics** library. This is your first Practice/Perform lab. You will have 1 day to *practice* this assignment. On the practice day you can ask questions and get help. Then you need to *perform*. On this day you need to do the lab, from scratch, for a grade – with no help. Some teachers call this "The Day of Reckoning".

Whether practicing or performing, you will be provided with a skeleton program. Your job is to use the proper procedures from the **Graphics** library along with the correct argument values to match the output shown on this assignment. Students are allowed to refer to the **Graphics Library Reference** document while practicing AND when they do the lab for a grade. In fact, students may refer to this document during any lab and any test. The first couple pages of the document are shown below. The entire document can be found in the **IntroCS-Graphics Files** subfolder of your **LearnIntroCS** folder.



Mr. Schram's Graphics Library for Python

This version of the **Graphics** library was written by Mr. John Schram 10/3/18 for use in first year Computer Science 1 or Computer Science 1-Honors / PreAPCS.

While built on "Turtle Graphics", the procedures and functions below allow graphics programming with more "Traditional Graphics" commands for greater convenience. It was inspired by a similar graphics library created by Mr. Leon Schram and is designed to operate in a manner similar to that of the **Expo** class that we created for "Exposure Java".

This code is free software. You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. This code is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

beginGrfx(windowWidth, windowHeight)
This will create a graphics window whose size is determined by the parameters.
This is always the first command used right after the **Graphics** library is imported.
Example:
```
from Graphics import *
beginGrfx(1300,700)
```
This creates a graphics window that is 1300 pixels wide and 700 pixels tall.

delay(milliseconds)
Updates the graphics window and makes the computer pause for a certain number of milliseconds.
Examples:
```
delay(1000)    # pause for 1 second
delay(2000)    # pause for 2 seconds
delay(500)     # pause for 1/2 of a second
```

drawArc(centerX, centerY, hRadius, vRadius, start, finish)
Draws and arc which looks like a curve.
An ARC is a "piece" of an OVAL.
The first 4 parameters are the same as **drawOval**.
There are 2 additional parameters for the starting degree value and finishing degree of the arc.
0 degrees is at the 12:00 position and the degrees progress in a CLOCKWISE fashion.
(90 degrees is at 3:00, 180 degrees is at 6:00, 270 degrees is at 9:00, 360 degrees is back at 12:00)
Example:
```
drawArc(300,200,100,100,135,45)
```
Draws an open arc which is a 3/4 piece of a circle with a radius of 100 pixels whose center is located at coordinate (300,200).
This arc will resemble the letter "C".

drawBurst(centerX, centerY, radius, numLines)
Draws a certain number of lines (**numLines**) in a burst like pattern.
The lines are evenly spaced and drawn from the center of a circle to its edge.
The size of the circle is specified by the **radius** parameter.
Example:
```
drawBurst(300,200,100,50)
```
Draws a "burst" with a radius of 100 pixels whose center is located at coordinate (300,200).
The "burst" will be comprised of 50 evenly spaced lines.

drawCircle(centerX, centerY, radius)
Draws an open circle.
The center of the circle is specified by **centerX,centerY** and its size is specified by **radius**.
Example:
```
drawCircle(300,200,100)
```
Draws an open circle with a radius of 100 pixels whose center is located at coordinate (300,200).

drawHeading(name, labNum)
Displays a heading at the top of the graphics screen.
The user provides his/her name and the number of the lab assignment.
Example:
```
drawHeading("John Smith","6B")
```
Displays "Lab 6B By: John Smith" centered at the top of the graphics screen in big bold letters.
A separating line will also be drawn across the graphics screen 50 pixels below the top.

drawLine(x1, y1, x2, y2)
Draws the line segment connecting coordinates x1,y1 and x2,y2.
Example:
```
drawLine(100,200,300,400)
```
Draws a line segment connecting the starting coordinate point (100,200) with the ending point (300,400).

drawOval(centerX, centerY, hRadius, vRadius)
Draws an open oval.
The user specifies the x,y coordinate of the center of the oval as well as the horizontal and vertical radii values.
Example:
```
drawOval(300,200,100,50)
```
Draws an open oval with a horizontal radius of 100 pixels and a vertical radius of 50 pixels.
The center of this oval is located at coordinate (300,200).

drawPixel(x, y)
Draws a single pixel at the specified x,y location.
Example:
```
drawPixel(100,200)
```
Draws a very small single dot (pixel) on the graphics screen 100 pixels over and 200 pixels down.

Lab 06B Student Version	Do not copy this file, which is provided.

```
 1 # Lab06Bst.py
 2 # "The Graphics Library Program"
 3 # This is the student, starting version of Lab 06B.
 4
 5
 6 from Graphics import *
 7
 8 beginGrfx(1300,700)
 9
10 # Substitute your own name here.
11 drawHeading("John Smith","6B")
12
13
14 # DRAW CUBE
15
16
17
18 # DRAW TARGET
19
20
21
22 # DRAW 7-SIDED DESIGN
23
24
25
26 # DRAW STOPLIGHT
27
28
29
30 # DRAW JPIIHS
31
32
33
34 # DRAW SMILEY FACE
35
36
37
38 # DRAW WEIRD TRIANGLE
39
40
41
42
43 endGrfx()
44
```

50, 60, 70, 80, 90, 100 and 110 Point Versions

The 50-point version displays any 1 of the 7 pictures below.

The 60-point version displays any 2 of the 7 pictures below.

The 70-point version displays any 3 of the 7 pictures below.

The 80-point version displays any 4 of the 7 pictures below.

The 90-point version displays any 5 of the 7 pictures below.

The 100-point version displays any 6 of the 7 pictures below.

The 110-point version displays ALL 7 pictures below.

NOTE
The pictures do not need to look exactly as they appear below. They should be very similar.
They also should not overlap with any other picture.

The *Target* must have 5 *concentric circles*. The spacing between all the circles should be the same.

The *7-Sided-Design* is made by placing a black 7-point burst on top of a white 7-point filled star on top of a black 7-sided filled regular polygon.

The *Face* should be made up of 3 *ovals*, 3 *arcs* and 2 *points*.

In the *Triangle*, each of the 3 *interior lines* connects a *corner* to the *midpoint* of the line segment on the opposite side. The triangle does not need to be *equilateral*, but it does need to be *isosceles*.