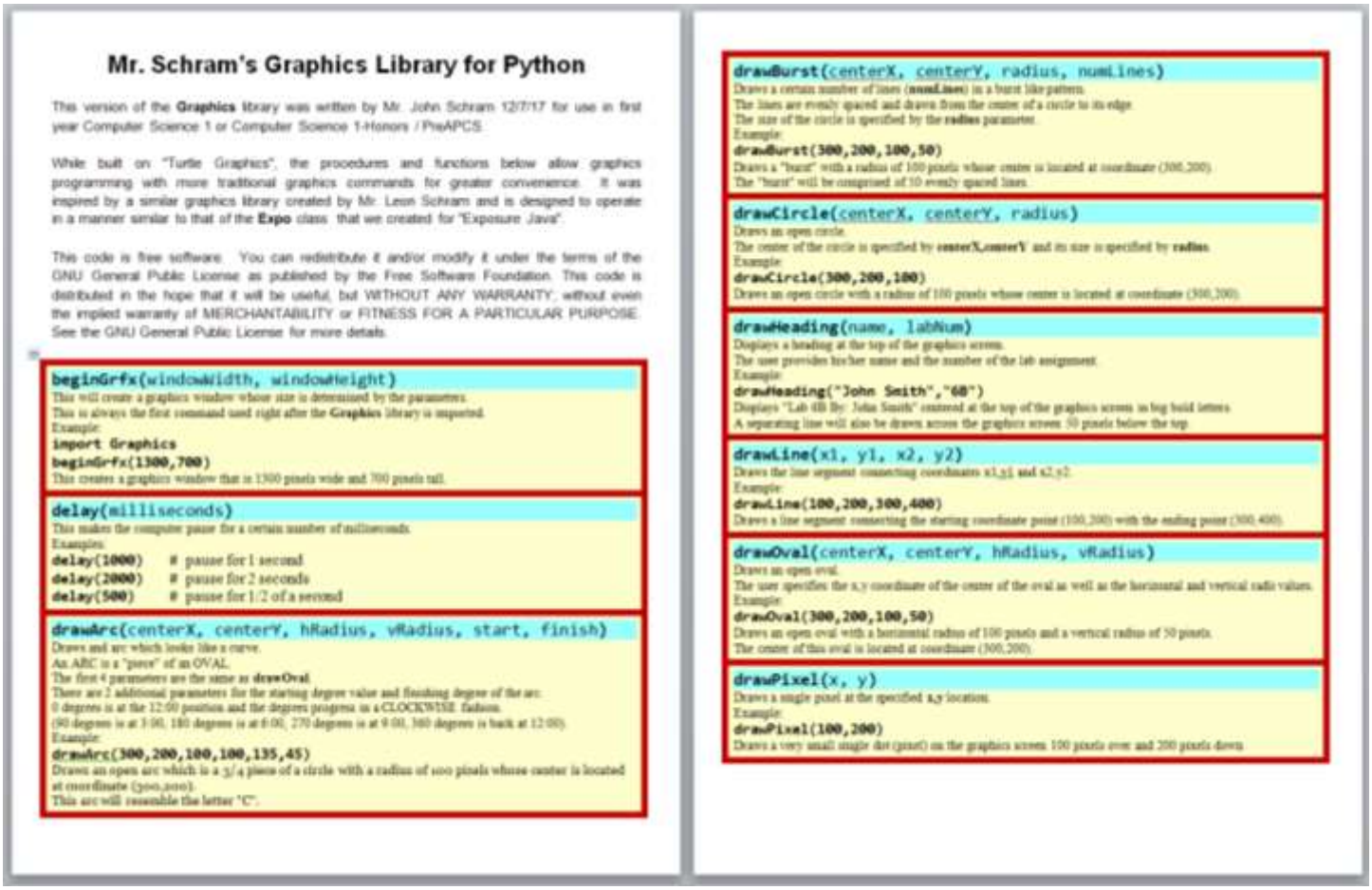
Computer Science 1	Lab 06C Practice/Perform Major Python Assignment
The Graphics Library Program II	**50, 60, 70, 80, 90, 100 & 110 Point Versions**

Assignment Purpose:

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

You will write another program, which displays several geometric designs using the provided **Graphics** library. The difference between this lab, and **Lab 06B**, is that you do not know what these geometric designs will be ahead of time. Also, you do not get separate practice day for this lab. You practiced when you did **Lab 06B**. As before, you will be provided with a skeleton program. Your job still is to use the proper procedures from the **Graphics** library along with the correct argument values to match the output shown on this assignment – which will be provided by your teacher when you do this for a grade. Remember, students are still allowed to refer to the **Graphics Library Reference** document while practicing AND when they do the lab for a grade. 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 12/7/17 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:
```
import Graphics
beginGrfx(1300,700)
```
This creates a graphics window that is 1300 pixels wide and 700 pixels tall.

delay(milliseconds)
This 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 06C Student Version	Do not copy this file, which is provided.

```
 1 # Lab06Cst.py
 2 # "The Graphics Library Program II"
 3 # This is the student, starting version of Lab 06C.
 4
 5
 6 from Graphics import *
 7
 8 beginGrfx(1300,700)
 9
10 # Substitute your own name here.
11 drawHeading("John Smith","6C")
12
13
14
15
16
17
18
19
20 endGrfx()
21
```