1) (~15 mins) Write a program with a GUI (graphical user interface) with an Entry object (a text field, as seen in class) that prompts the user as follows, then does as promised:

I will draw you a circle.
Enter the number of pixels you want for the diameter of your circle.
Then click the spot where you want the circle to appear.

Get check 1

2) In your animated house scene from last lab:
   a) (~15 mins) Create a smattering of stars in the night sky after night falls. (Hint: use a Point object for each star, and for generating the coordinates for each Point, you can use the built-in random number generator randrange() which you used in lab 3.) Don’t worry for now if some stars get drawn on your house. (Finding a way to prevent this is an extra credit bonus you can pursue after you’ve earned all your checks.)

Get check 2

b) (~15 mins) Today in class we wrote a drawButton() function so that it was easy for the caller of the function to draw a button wherever they want in a graphical window. This same idea will now allow the user to make grass on the lawn of the house scene from last lab. We will break this down into smaller components first:
   i) Recall that we saw how we can use mouse click point information (the point returned from the win.getMouse() function) to draw something at the point the user clicks.
   ii) Create a for loop to take 10 of these mouse clicks. In each iteration, you will need to save the Point object returned from getMouse() function (i.e. ‘catch’ the Point object in a variable as in class → pt = win.getMouse() )
   iii) Then, for now create a green placeholder object (a circle of radius 30 perhaps?) and draw it where the user clicked.

Get check 3

iv) (~20 mins) Now, we can make this code cleaner by adding a new function called drawGrass(), that will replace the code that makes the green placeholder. The modified function should create actual grass, as shown in this video: http://oak.conncoll.edu/james-lee/com110_2015S/lab5.html. See the function header below:

# This function draws a grass tuft. It takes two parameters.
# The first one is a GraphWin instance and the second is a Point instance.
def drawGrass(win, point):

v) Inside this function, draw a polygon with at least three points. (See Zelle reference for how to make polygons.) Note that you do NOT want to hard-code in specific points; instead, make your x and y locations relative to the passed-in Point parameter of the drawGrass(win, point) function. To do this, remember that you can get the x and y of a Point object using the Point methods point.getX() and point.getY().

vi) For your reference the below picture shows a polygon made to depict grass. It uses 7 points, but you can come up with your own shape. Sketch this out to find good point coordinates.

![Polygon depicting grass]

vii) After this custom function definition, we can replace the placeholder drawing from the last check with a call to drawGrass(win, point). Make sure to leave the for loop that allows multiple grass polygons to be made in your main function. That way you can get the user clicks in the main function and send them to the drawGrass(win, point) function with your GraphWin instance.

Get check 4

3) [~10 mins] Refer to your custom square drawing program that you created using nested loops in lab 3. It was something like this:

```python
from graphics import *

win = GraphWin("Lab 4 Fun", 600, 600)
for x in range(250, 351):
    for y in range(200, 301):
        p = Point(x,y)
        p.setFill("red")
        p.draw(win)
```
a) On a piece of paper, trace the values of the variables $x$ and $y$, showing how they change over the course of the nested for loop. I.e., make a table that starts like this:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>200</td>
</tr>
<tr>
<td>250</td>
<td>201</td>
</tr>
<tr>
<td>250</td>
<td>202</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

So you don't have to actually write out all 100 points, you may make liberal use of the ellipses (...), which will indicate that the values continue on in the same “pattern.” Just make sure you are explicit about the values again whenever the pattern changes. Draw bold horizontal lines to clearly demarcate the rows where the outer loop iterates. To check your answer, print out to the shell each new $x$ and $y$ value inside the nested for loop.

4) [~5 mins] Functions that return values

Notice that most functions/methods we use not only take some action, but also have a value or “return a value” that we can then ‘catch’ in a variable.

E.g.,
- `name = input("Enter your name: ")`
- `num = str.count(substr)`
- `str = file.read()`
- `pt = win.getMouse()`
- `xcoord = pt.getX()`

Write down the type of value each of the above functions returns. (E.g., string, integer, float, boolean, Zelle Point object, etc.)

[~15 mins] Others functions we’ve used only take actions without returning any value. E.g.,
- `sleep(1.5)`
- `object.draw(win)`
- `object.setFill(color)`
So far, we’ve created our own functions of the second kind, but not the first (the kind that returns something).

Functions that return values can be created by simply adding a return statement in the function definition:

```python
def functionname(parameters):
    <body of function>
    return <value to be returned>
```

- You can return any type of value, an integer, a string, a boolean, a list, an object, etc, etc
- As soon as the return statement is encountered, the execution of the function ends and control goes back to where the function was called from, even if there is more code left in the function!!
- One final note: if a function returns more than one value you can separate them with commas:
  ```python
  return <value1>, <value2>, ...
  ```
- As an example, you can go to graphics.py and see how Zelle defines the .getMouse() function. It looks like something this:

```python
def getMouse():
    ...
    x = <x-coord of mouse click>
    y = <y-coord of mouse click>
    return Point(x,y)
```

- As another example, in your reading this week there is a cute triangle drawing program that lets the user click three points to draw a triangle, then computes the perimeter of the triangle. To accomplish this, it uses a function that calculates the distance between two points!
- Let’s see how useful boolean functions can be by solving the following problem.

```python
#Print out the prime numbers between 1 and a user inputted value.
#Note: a number is prime when it is divisible only by 1 and itself.
#I.e., when it doesn't have any divisors other than 1 and itself.

#returns True if num is prime, returns False otherwise
```
def isPrime(num):
    # check if num has any divisors other than 1
    # so we loop through each integer i between 2 and num-1
    ## your for loop header goes here
    # check if num is divisible by it
    ## your if statement goes here
    # if it is, return False, which ends the function
    # if we made it this far, there must be no divisors
    ## so you can return True

[~15 minutes] Now we’re ready to solve the original problem of printing out all primes between 1 and a user-inputted value. This should be done in a separate function from your isPrime() function above. Here’s the outline.

def main():
    # prompt user for positive integer, n
    # for each integer i between 2 and n
    # call your isPrime() function to check if i is prime
    # if it is, print it

Extra time? In addition to last week’s bonuses, you may also complete the following, in any order, getting checked one at a time:

A. Edit the nested for loops program for drawing a square (from check 5 above) so that it
draws the square in same place as before, but rather than looping x through
range(250,351) it loops through range(100). Then further edit it so that y loops
through range(100).
B. Finish the task we began in class on Tuesday. In a graphical window, draw a zelle Circle
or Rectangle. Upon a user-click, make the shape animate/move to the point where the
user clicked. It will be useful to know that Point() objects have methods .getX() and
.getY() that return the x and y coordinates of the point. E.g.,
    pt = win.getMouse()
    x = pt.getX()
    y = pt.getY()
C. Modify our program from class that inputted a color from the user to color a moving
square. Have it now check whether the user-input is a valid Python graphics color.
(Refer to the list of colors linked from the last lab.) If not, prompt the user with one
more attempt: “Sorry, I don’t recognize that color. Please enter a color then click.”
D. In your house scene, rather than have your sun set along a straight line, have it set along
a curve.
E. Find a way to avoid stars from showing up on your house.

Get these bonuses in any order, one at a time