Name: Solutions (sketches)

Part 1: Short Answer

(a) What is the difference between a *function* and a *method*? Explain your answer, using at least one example of a function and one example of a method.

A function stands alone in a module and does not belong to a particular instance of a class. A method belongs to a class and is called on a specific instance. For example, `binary_search(elem, lst)` is a function and `die.set_value(3)` is a method.

(b) What is the difference between an *iterative* approach and a *recursive* approach?

An iterative approach uses a loop to iterate over pieces of the problem, ultimately building up a solution. A recursive function calls itself over and over to reduce the size of the problem.

(c) In class we used a while loop to perform shuffle sort (shown below). Rewrite this function to make it recursive (assume we have the `random` module and a correct `in_order` function).

```
def shuffle_sort_while(lst):
    while in_order(lst) == False:
        random.shuffle(lst)
```

```
def shuffle_sort(lst): # TODO complete function using recursion
    if in_order(lst) == False:
        random.shuffle(lst)
        shuffle_sort(lst)
```

(d) Does a shuffle sort function need to return anything? Why or why not?

No - `random.shuffle(...)` modifies (mutates) the list in place, so we don't need to return anything to successfully sort the list. We can return the number of shuffles if we want to.

Part 2: Functions and while loops

Analyze the code below and answer the following questions.

```
import math
def mystery_function(n):

    for d in range(2,int(math.sqrt(n))+1):
        if n % d == 0:
            return False
    return True
```

(a) For each argument n below, what does the function return? Square roots are provided below.

- n=2, $\sqrt{2} \approx 1.41$, return value: True (never enters loop)
- n=3, $\sqrt{3} \approx 1.73$, return value: True (" " ")
- n=4, $\sqrt{4} = 2$, return value: False $4 \% 2 = 0$
- n=5, $\sqrt{5} \approx 2.24$, return value: True $5 \% 2 = 1$ (only checks 2)
- n=6, $\sqrt{6} \approx 2.49$, return value: False $6 \% 2 = 0$
- n=7, $\sqrt{7} \approx 2.65$, return value: True $7 \% 2 = 1$ (" " ")
- n=8, $\sqrt{8} \approx 2.83$, return value: False $8 \% 2 = 0$
- n=9, $\sqrt{9} = 3$, return value: False $9 \% 3 = 0$

(b) Based on your answers to (a), describe what this function is doing in words.

Returns True if n is prime, False otherwise.

(c) Rewrite the same function using a *while loop*. Although a while loop is not necessary for improving the function since we return right away, it is good practice to be able to use a different type of structure for the same problem.

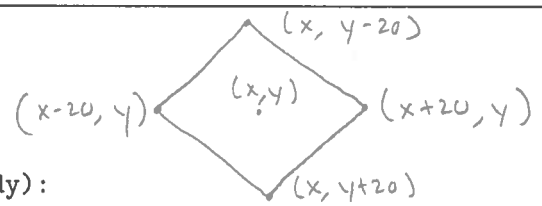
```
import math
def mystery_while_loop(n):

    d = 2
    while d <= int(math.sqrt(n)):
        if n % d == 0:
            return False
        d += 1
    return True
```

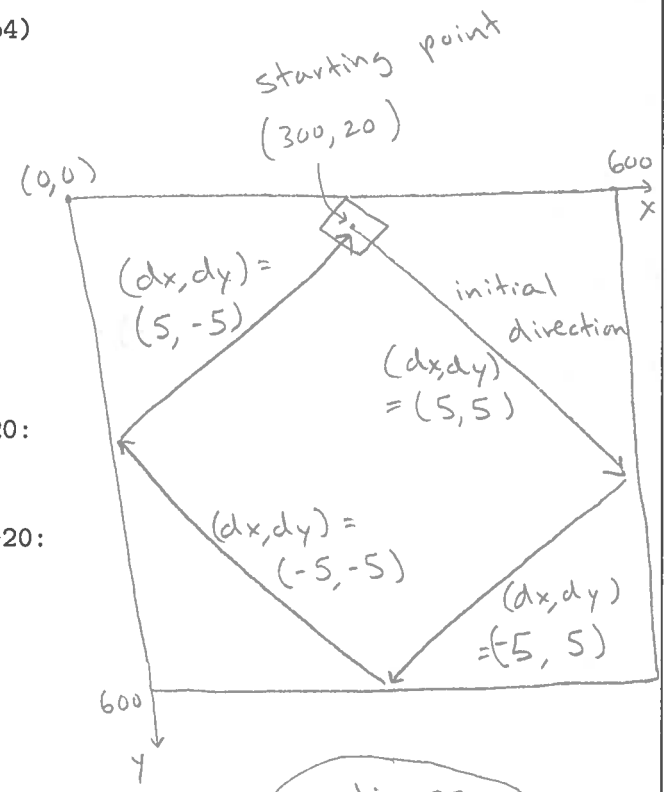
Part 3: Classes in Graphics

Analyze the code below and answer the following questions (draw in the space beside the code).

```
from graphics import *

class MysteryShape:

    def __init__(self, x, y, dx, dy):
        p1,p2,p3,p4 = Point(x-20,y), Point(x,y-20), Point(x+20,y), Point(x,y+20)

        self.shape = Polygon(p1, p2, p3, p4)
        self.shape.setFill("silver")

        self.x, self.y = x, y
        self.dx, self.dy = dx, dy

    def draw(self, window):
        self.shape.draw(window)

    def move(self, width, height):

        if self.x < 20 or self.x > width-20:
            self.dx = -self.dx

        if self.y < 20 or self.y > height-20:
            self.dy = -self.dy

        self.shape.move(self.dx,self.dy)
        self.x += self.dx
        self.y += self.dy

def main():
    width, height = 600, 600
    win = GraphWin("Mystery Movement", width, height)

    shape = MysteryShape(width/2, 20, 5, 5)
    shape.draw(win)
    while True:
        shape.move(width, height)
main()
```

diamond shape





continues forever in a diamond shaped pattern

(a) How many *instance variables* does this class have? 5

(b) Draw a picture of the initial setup (after constructor and draw method are called).

(c) On your picture, show the trajectory of the shape using lines and arrows.

(d) Will the shape ever leave the viewable window? Why or why not?

No - travel direction is flipped every time it encounters an edge.

Part 4: Writing Classes

In this question the goal is to write a `GameHost` class. Instances of this class will know a secret number, which the user is trying to guess. They will check if the user's guess matches their secret answer. Below is the main function (left) and some example output of running main (right).

```
def main():
    low, high = 1, 10
    host = GameHost(low, high)

    while not host.get_is_guessed():
        number = random.randint(low,high)
        host.try_guess(number)
main()
```

```
3 is wrong! Guess again.
10 is wrong! Guess again.
7 is wrong! Guess again.
6 is wrong! Guess again.
9 is wrong! Guess again.
2 is correct! You took 6 tries.
```

Use the steps below to write the `GameHost` class, which will interact correctly with the existing main function and product the desired output.

(a) Write the constructor. Define an instance variable that will be the secret number (pick a random number between `low` and `high`). Then define instance variables that will keep track of whether or not the user has guessed correctly and how many tries the user has made.

(b) Write a getter for the instance variable which says if the user has guessed correctly.

(c) Write the `try_guess` method, which will take in a guess and check it against the secret number. It will also update the relevant instance variables and print out information.

```
import random
class GameHost:
    def __init__(self, low, high):
        self.answer = random.randint(low, high) # secret answer
        self.is_guessed = False
        self.num_tries = 0

    def get_is_guessed(self):
        return self.is_guessed

    def try_guess(self, number):
        self.num_tries += 1
        if self.answer == number:
            print(number, "is correct! You took", self.num_tries,
                  "tries.")
            self.is_guessed = True
        else:
            print(number, "is wrong! Guess again.")
```