The goal of this Python programming assignment is again to write your own code inside a provided program framework, with some new graphical and mathematical considerations. You are to perform this assignment in the PC lab at your scheduled lab time.

In the JES environment, type in the following program code exactly as you see it here and save the program as `Lab8.py` in your Python folder. Where you see my name in the comment code, replace it with your own name. As always, be very careful about indentation and capitalization. Where you see the `|` symbols open up several blank lines; this is where you will be writing your own code at a later time.

```python
# William T. Verts - Lab #8 - Fish Tank

import random, time

def addCircleFilled(Canvas, Xc, Yc, R, NewColor):
    addOvalFilled(Canvas, Xc-R, Yc-R, 2*R+1, 2*R+1, NewColor)
    return

def addEllipseFilled(Canvas, Xc, Yc, Xr, Yr, NewColor):
    addOvalFilled(Canvas, Xc-Xr, Yc-Yr, 2*Xr+1, 2*Yr+1, NewColor)
    return

def FishLeft(Canvas, Xc, Yc, NewColor):
    addEllipseFilled(Canvas, Xc+14, Yc, 3, 14, white)  # Tail
    addEllipseFilled(Canvas, Xc, Yc, 17, 9, NewColor)  # Body
    addCircleFilled(Canvas, Xc-10, Yc-2, 3, white)    # Eye
    addLine(Canvas, Xc-16, Yc+4, Xc-7, Yc+4, white)  # Mouth
    return

def FishRight(Canvas, Xc, Yc, NewColor):
    addEllipseFilled(Canvas, Xc-14, Yc, 3, 14, white)  # Tail
    addEllipseFilled(Canvas, Xc, Yc, 17, 9, NewColor)  # Body
    addCircleFilled(Canvas, Xc+10, Yc-2, 3, white)    # Eye
    addLine(Canvas, Xc+16, Yc+4, Xc+7, Yc+4, white)  # Mouth
    return

def FishTank(Canvas):
    TotalFish = 20
    |
    |
    |
    return

def Run():
    FishTank(makeEmptyPicture(640,480))
    return
```
In JES click the Load Program button. At the >>> prompt, type Run() with the parentheses and press Enter. The program should run, but should not do anything visible. Fix any syntax errors or other mistakes. We will not change Run while developing the program, but we will fill in code for the other functions.

**The Goal**

The goal of this project is to create an animated fish tank with an arbitrary number of fish swimming around. A typical screen is shown as follows, with 20 fish, after they have been swimming around for a while:

We will have you write the code so that you can change the number of fish and the program will still work correctly.

We have provided the addCircleFilled function, which you should recognize, along with a similar routine, addEllipseFilled, that plots an ellipse (oval) centered at location <Xc, Yc> with radii Xr (horizontal) and Yr (vertical). We have also provided two “fish” routines, FishLeft for a fish swimming to the left and FishRight for a fish swimming to the right. The two fish-drawing routines are left-right symmetric. You are allowed to modify these routines to create unique fish of your own design, but please make sure that the two routines preserve the left-right symmetry. The FishTank function is currently a stub, and this function is what you must finish. Notice that the Run function combines the makeEmptyPicture into the parameter list of the call to FishTank, and that there is no initial color for the canvas. Both are by design, and are not errors.

**Initialization**

Inside the FishTank function you will need to create five separate lists, one each for the X coordinates, the Y coordinates, the X direction, the Y direction, and the color of each individual fish. The number of entries in each list will be determined by the TotalFish variable, shown set to 20 in the code on the previous page but allowed to be any value from 3 to 1000. You’ll need to build these lists using loops controlled by TotalFish.
The initial values for the X and Y coordinates should be the middle of the canvas, plus a random number between -25 and +25. The initial values for the X and Y directions should be a random number between -1 and +1 (including 0). To get a random number between -N and +N, for any value of N, use the function `random.randrange(-N,N+1)`. For example, to get a random number between -1 and +1 you would use `random.randrange(-1,+2)`. The initial color for each fish will be chosen at random from the list `[black, blue, green, cyan, red, magenta, yellow]` (by intent the list does not include white, as the eye, tail, and mouth of each fish are white).

**The Big Loop**

Once the five lists have been initialized, the program should go into an infinite loop (really!), controlled by a while loop where the test condition is always true (notice the capitalization):

```python
while (True):
```

The loop clears the screen to dark blue, paints all the fish at their current locations, updates the positions and directions of each fish, repaints the screen, then sleeps for 0.05 seconds.

**Dark Blue**

To create the dark blue background color, you may use the JES functions `setAllPixelsToAColor` and `makeColor(R,G,B)`. I suggest using 0 for both R and G, and 128 for B.

**Painting Fish**

Paint each defined fish at the location indicated by the corresponding values in the X and Y coordinate lists. Use the corresponding value from the X direction list to determine direction: if the X direction is -1 (fish is swimming to the left) paint a `FishLeft` and if the X direction is 0 or +1 (fish is swimming to the right) paint a `FishRight`. Use the corresponding value from the color list to determine the color of each fish. You will not use the Y direction list in painting the fish on the screen.

**Updating Positions**

For each fish, update its position by adding its X direction to its X coordinate, and then adding its Y direction to its Y coordinate. If the fish goes off the left or right sides of the canvas, negate its X direction, and if the fish goes off the top or bottom sides of the canvas, negate its Y direction. This will keep the fish always visible on screen.

For added realism, we next use random numbers to give each fish the ability to change its direction in the middle of the canvas. Pick a random number between 0 and 9; if that value is 0 reinitialize the X direction for that fish to a random number between -1 and +1. This means that one time out of ten the fish will suddenly “decide” to go left, go right, or stand still, regardless of what it was doing before.

Similarly, pick a random number between 0 and 19; if that value is 0 reinitialize the Y direction for that fish to a random number between -1 and +1. This means that one time out of twenty the fish will suddenly “decide” to float up, float down, or hover.
Repaint and Sleep

When the fish are painted and their positions updated, repaint the canvas and then sleep for 0.05 seconds. This will show the finished fish on the screen for a while before the next update of their positions.

Finishing Up

That is enough information for you to figure out how to fill out the FishTank function and run the program. You will need to hit the Stop button in the JES environment to break out of the infinite loop. Try changing the size of the screen and the number of fish. Design your own fish if you like.

When you are finished and everything runs correctly, print out (File-Print) the source code to turn in. Also, capture and print a screen shot that shows the result of running the program – make certain that before you capture the screen the JES window shows as much of your code as possible (including the part with your name) AND the result of the execution (the image).

(In Windows, hit the Prt Scn or Prnt Scrn or PrtSc button to copy the image to the clipboard, paste the image into Windows Paint (or your favorite image processing program), and print it from there; it may come out on multiple pages. On the Mac, hitting the key combination Cmd-Shift-3 will save the screen as a .png file on the desktop; double-click the .png file to bring it into the Preview program, and then print it from there.)

Turn in to the TA the printout from JES and the screen shot, stapled together in that order (the source code version printed from JES on top, the screen shot image on the bottom).