20 Points – Rewrite the following code to use for-loops instead of while-loops. Use as few Python statements as possible without changing how the function works.

```python
def Flood(Canvas, NewRed):
    W = getWidth(Canvas)
    H = getHeight(Canvas)
    Y = 0
    while (Y < H):
        X = 0
        while (X < W):
            PX = getPixel(Canvas, X, Y)
            setRed(PX, NewRed)
            X = X + 1
        Y = Y + 1
    return
```

There are two acceptable approaches for this problem. The most straightforward of the two is to mechanically replace each while-loop with the equivalent for-loop, while also embedding the calls to getWidth, getHeight, and getPixel directly into the code where they are used:

```python
def Flood(Canvas, NewRed):
    for Y in range(getHeight(Canvas)):
        for X in range(getWidth(Canvas)):
            PX = getPixel(Canvas, X, Y)
            setRed(PX, NewRed)
        Y = Y + 1
    return
```

The second acceptable approach is to recognize that all pixels in the canvas are being scanned, and use the JES getPixels function as part of a single for-loop:

```python
def Flood(Canvas, NewRed):
    for PX in getPixels(Canvas):
        setRed(PX, NewRed)
    return
```

Of course, this last version can be made even shorter by the rule that if the body of an if, while, or for contains a single statement, it can be placed on the same line:

```python
def Flood(Canvas, NewRed):
    for PX in getPixels(Canvas): setRed(PX, NewRed)
    return
```
20 Points – What is printed out when `Main()` is called?

```python
def Toad(W, H):
    printNow(W*2+H)
    return

def Frog(W, X):
    if (W > X):
        Toad(X, W)
    else:
        Toad(X, X)
    return

def Main():
    Frog(4, 7)  # prints 21
    Frog(7, 4)  # prints 15
    return
```

In the first call `Frog(4, 7)`, 4 goes into `Frog` as `W`, and 7 goes in as `X`. `W` is not greater than `X`, so it makes the second call to `Toad` as: `Toad(7, 7)`. Inside `Toad`, `W`=7 and `H`=7, and the result of `W*2+H` is `7*2+7 = 21`, which is printed.

In the second call `Frog(7, 4)`, 7 goes into `Frog` as `W`, and 4 goes in as `X`. `W` is greater than `X`, so it makes the first call to `Toad` as: `Toad(4, 7)`. Inside `Toad`, `W`=4 and `H`=7, and the result of `W*2+H` is `4*2+7 = 15`, which is printed.
20 Points – Using the 3D axes below and orthographic projection, identify (mark and label) the location of each 3D point $[4,0,0]$, $[2,0,5]$, $[-3,4,0]$, $[-5,1,2]$, $[2,-6,0]$. Consider the tic marks to have a 1-unit separation.
20 Points – Consider the code we’ve developed to blend 3D points:

```python
def Blend (N0,N1,T):
    return (N1 - N0) * float(T) + N0

def BlendPoints3D (P0,P1,T):
    X = Blend(P0[0], P1[0], T)
    Y = Blend(P0[1], P1[1], T)
    Z = Blend(P0[2], P1[2], T)
    return [X,Y,Z]
```

What are the results of the following function calls?

A. `Blend(5, 11, 0.5)`
   
   **8.0**  (floating point, not integer)

B. `BlendPoints3D ([10,20,30], [36,67,49], 0)`
   
   `[10.0, 20.0, 30.0]`

C. `BlendPoints3D ([10,20,30], [36,67,49], 0.5)`
   
   `[23.0, 43.5, 39.5]`

D. `BlendPoints3D ([10,20,30], [36,67,49], 1)`
   
   `[36.0, 67.0, 49.0]`
20 Points – In the image at right of a funny object, grid lines are 10 pixels apart and do not appear in the final drawing. You are given two functions for drawing circles and rectangles with one color for the interior (FillColor) and one for outlines (OutlineColor). A FunnyObject is defined as a blue square with radius 100 centered at <X,Y> (indicated by the white dot), with a WeirdThing centered at each corner. A WeirdThing is a green square with radius 50 and a red circle centered at each of its corners. The radius of each circle is 30. The outline color in every case is black. The reference point for drawing each object (the FunnyObject, each WeirdThing, and each circle) is in its center. Complete the functions below to draw a proper FunnyObject.

```python
def addCircle (Canvas,Xc,Yc,R,FillColor,OutlineColor):
    addOvalFilled(Canvas,Xc-R,Yc-R,2*R+1,2*R+1,FillColor)
    addOval(Canvas,Xc-R,Yc-R,2*R+1,2*R+1,OutlineColor)
    return

def addRectangle (Canvas,X,Y,W,H,FillColor,OutlineColor):
    addRectFilled(Canvas,X,Y,W,H,FillColor)
    addRect(Canvas,X,Y,W,H,OutlineColor)
    return

def addWeirdThing (Canvas,X,Y):
    addRectangle (Canvas,X-50,Y-50,100,100,green,black)
    addCircle (Canvas,X-50,Y-50,30,red,black)
    addCircle (Canvas,X+50,Y-50,30,red,black)
    addCircle (Canvas,X-50,Y+50,30,red,black)
    addCircle (Canvas,X+50,Y+50,30,red,black)
    return

def addFunnyObject (Canvas,X,Y):
    addRectangle (Canvas,X-100,Y-100,200,200,blue,black)
    addWeirdThing (Canvas,X-100,Y-100)
    addWeirdThing (Canvas,X+100,Y-100)
    addWeirdThing (Canvas,X-100,Y+100)
    addWeirdThing (Canvas,X+100,Y+100)
    return
```

The addCircle statements can be in any order, as can the addWeirdThing statements.