CMPSCI 119
Fall 2018
Wednesday, November 14, 2018
Midterm #2 Solution Key
Professor William T. Verts

<1> 25 Points – What is the value of each expression below? Answer any 25; answer more for extra credit. Answer "Error" if an expression cannot be computed for any reason. Scoring:

+1: Completely correct answers
+½: Incorrect data types, lists without square brackets, strings without quotes, etc.
0: Blank answers
–½: Incorrect answers (better to leave it blank than to guess)

```
X15   = 7
ME262 = 10L
F104  = "PLANES AND SPACE"
B17   = ["AIR", 7, "TRAINS", 8.5, 3]
XF85  = ("SPACECRAFT", [4, 1, 3], 5L, 6.2)
```

1.	**3**	int	**X15 / 2**
2.	**17**	int	**X15 + ME262**
3.	**16.2**	float	**ME262 + XF85[-1]**
4.	**16**	int	**len(F104)**
5.	**3**	int	**len(B17[0])**
6.	**ERROR**		**len(ME262)**
7.	**3**	int	**len(XF85[1])**
8.	**15L** *(or ERROR)*	long	**ME262 + XF65[2]** *(should be* **XF85[2]***)*
9.	**ERROR**		**B17 + XF85**
10.	**8.0**	float	**X15 + 1.0**
11.	**[4,1,3,7]**	list	**XF85[1] + [X15]**
12.	**"AIRPLANES AND SPACE"**		**B17[0] + F104**
13.	**[0,1,2,3,4,5,6]**		**range(X15)**
14.	**[3,4,5,6]**		**range(B17[-1],X15)**
15.	**[1,8,15,22]**		**range(1,25,X15)**
16.	**ERROR**		**range(F104)**
17.	**ERROR**		**XF85[0] + ME262**
18.	**9**	int	**len(B17[0]+B17[2])**
19.	**1**	int	**X15 % 2**
20.	**False**	bool	**X15 < 3**
21.	**5**	int	**5 * (X15 > 4)**
22.	**49.0**	float	**X15**2.0**
23.	**[4,1,3,0,1,2]**		**XF85[1] + [I for I in range(3)]**
24.	**[-7,0,7]**		**range(-X15,X15+1,X15)**
25.	**8.5**	float	**B17[B17[-1]]**
26.	**[0,0,0,0,0,0,0]**		**[0 for I in range(X15)]**
27.	**[10,15,9]**		**[X15 + I for I in [3,8,2]]**
28.	**["A","I","R"]**		**[CH for CH in B17[0]]**
29.	**[1,1,1]**		**[len(CH) for CH in B17[0]]**
30.	**["R","I","A"]**		**[B17[0][len(B17[0])-1-I] for I in range(len(B17[0]))]**

<2> 20 Points – (2 points each answer) When **`Main()`** is called there will be exactly ten lines of output printed. What are they?

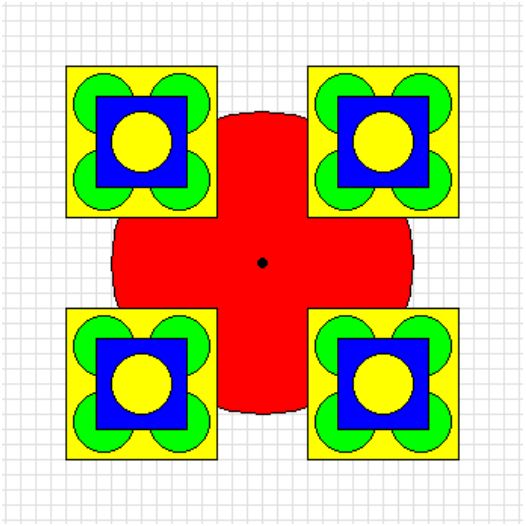
```
def FN1 (F,R,O,G=1):              # Answer #1    4
    global Newt                   #
    Newt = (F * R) + (O * G)      # Answer #2    18
    print Newt + F                #
    return                        # Answer #3    14
                                  #
def FN2 (T,O,A,D):                # Answer #4    3
    global Newt                   #
    FN1 (D+Newt,O,A,T)            # Answer #5    7
    print Newt                    #
    return                        # Answer #6    4
                                  #
def Main():                       # Answer #7    11
    FN1 (1,2,1)                   #
    FN2 (3,2,2,1)                 # Answer #8    6
    FN1 (1,1,1,1)                 #
    FN2 (1,1,1,1)                 # Answer #9    28
    FN2 (1,1,1,1)                 #
    FN2 (2,2,2,2)                 # Answer #10   20
    return                        #
```

<3> 10 Points – I have a program that scans every pixel in an image, and calls function **`Process`** with each pixel in variable **`PX`**. Complete the **`Process`** function to transform the pixel by converting it to black if the brightness is less than 50, white if the brightness is greater than 200, and gray otherwise. (The only JES functions you need are **`getRed`**, **`getGreen`**, **`getBlue`**, and **`setColor`**., but no canvases or loops are needed.)

```
def Process(PX):
    Brightness = (getRed(PX)+getGreen(PX)+getBlue(PX))/3
    if   (Brightness > 200): setColor(PX,white)
    elif (Brightness <  50): setColor(PX,black)
    else:                    setColor(PX,gray)
    return
```

Scoring: 2 points for computing the brightness, 3 points for the `if-elif-else` structure (can be `if-if-if`), 1 point each for the three `setColor` calls, 2 points for syntax errors.

<4> 25 Points – Each grid square in the gray alignment grid is 10 pixels on a side. A **`Widget`**, centered at location <X,Y> (shown with a dot), is a red circle of radius 100 pixels, with four **`Corner`** shapes on top. Each **`Corner`** is centered 80 pixels in both X and Y away from the **`Widget`** center.



A **`Corner`** is, relative to its own <X,Y>, a centered yellow square of side-radius 50, with four green circles of radius 20 on top of it, 25 pixels in both X and Y away from its center, a blue square of side-radius 30 on centered top of the circles, and a yellow circle of radius 20 centered on top of all. Fill in the blanks below to complete the drawing of a **`Widget`**.

The **`Circle`** and **`Square`** functions are already provided.

```
def Circle (Canvas, Xc, Yc, Radius, NewColor=black):  …
def Square (Canvas, Xc, Yc, Radius, NewColor=black):  …

def Widget (Canvas, X, Y):

    def Corner (X, Y):
        Square(Canvas,  X    ,  Y    ,  50  , yellow)
        Circle(Canvas,  X-25 ,  Y-25 ,  20  , green )
        Circle(Canvas,  X+25 ,  Y-25 ,  20  , green )
        Circle(Canvas,  X-25 ,  Y+25 ,  20  , green )
        Circle(Canvas,  X+25 ,  Y+25 ,  20  , green )
        Square(Canvas,  X    ,  Y    ,  30  , blue  )
        Circle(Canvas,  X    ,  Y    ,  20  , yellow)
        return

    Circle (Canvas,  X  ,  Y  ,  100  ,  red  )
    for Row in [-1,1]:
        for Column in [-1,1]:
            Corner (  X + 80 * Column  ,  Y + 80 * Row  )
    return
```

Scoring: There are 34 slots to be filled in; 24 have either a single variable (`X` or `Y`), a number, or a color: score those as ½ point each for 12 points. There are 8 that are `±25`: score those as 1 point each for 8 points. The last two slots are complicated: score those as 2 points each for 4 points. Give 1 point free for any non-blank answers. NOTE: the four sequential `Circle` commands can be in any order.

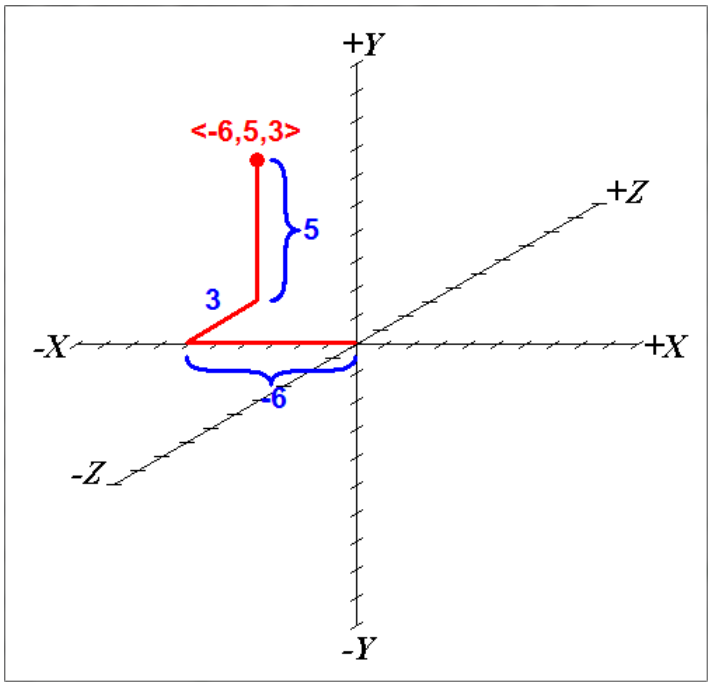
<5> 5 Points – Refer to the **`BlendPoints`** function we developed in class (or see the bottom of page 298 of the Companion). What is returned by the following function call?

```
BlendPoints ([4,12,8], [20,24,40], 0.25)
```

```
[8.0, 15.0, 16.0]
```

Scoring: 1 point for each of the three values, 1 point for the list `[]` brackets, and 1 point for other syntax problems.

<6> 5 Points – Show on the orthographic axes below where the point <-6,5,3> is located. Refer to page 320 in the Companion for how the answer should be drawn.



<7> 10 Points – Write a complete Python function called **`Guess`**, with no parameters, that use the python **`input`** function to ask the user for a numeric input between 0 and 100 (inclusive). It continues to do so if the user enters anything outside that range, but then returns the entered value when it is inside the correct range.

```
def Guess():
    N = input("Enter a number in [0...100]")
    while (N < 0) or (N > 100):
        N = input("Enter a number in [0...100]")
    return N
```

Scoring: 2 points for `def Guess():`, 2 points for the first `input` statement, 3 points for the `while` loop, 2 points for the second `input` statement, and 1 point for the `return`.