CMPSCI 119
Fall 2017
Monday, November 6, 2017
Midterm #2 Solution Key
Professor William T. Verts

NAME	

QUESTION	POINTS	SCORE
1	25+5	
2	5	
3	24	
4	20	
5	16	
6	10	
TOTAL	100	

<1> 25 Points – What is the value of each expression below? Answer any 25; answer more for extra credit. Variable `S = "Frogs and Toads"`, `L = [4,9,3,"Newt",5]`, `X = 6.5`, `T = (3,2,7,4)`, and `D = {1:"Dog", 8:"Cat", 4:"Bat", 9:"Rat", 3:"Aardvark"}`. Answer "Error" if an expression cannot be computed for any reason. Incorrect answers will be assessed as –1, correct answers as +1, and blank answers as 0. Your score will be the total (but will not go below zero). For example, if you answer all 30 problems but get 25 right and 5 wrong, your final score will be 25 - 5 = 20. Correct answers with incorrect data types are penalized ½ point each!

1.	`13.0` (float)	`X * 2`
2.	`1` (int)	`T[0] / 2`
3.	`5` (int)	`len(L)`
4.	`4` (int)	`len(L[3])`
5.	ERROR	`len(L[-1])`
6.	`5` (int)	`L[-1]`
7.	`"s"` (lowercase string)	`S[-1]`
8.	`4` (int)	`T[-1]`
9.	ERROR	`X[-1]`
10.	`"Toads"` (mixedcase string)	`S[10:]`
11.	`"Frogs"` (mixedcase string)	`S[:5]`
12.	`"and"` (lowercase string)	`S[6:9]`
13.	ERROR	`S[25]`
14.	`"Newts"` (mixedcase string)	`L[3] + S[4]`
15.	`"Bat"` (mixedcase string)	`D[4]`
16.	ERROR	`D[5]`
17.	`8` (int)	`len(D[3])`
18.	`2.0` (float)	`sqrt(L[0])`
19.	`[4,9,3]` (list)	`L[:3]`
20.	ERROR	`X + [5]`
21.	`16.5` (float)	`X + len(range(10))`
22.	`[0,1,2,3,4,5,6,7,8]` (list)	`range(L[1])`
23.	`[2,5,8]` (list)	`range(T[1],L[1],T[0])`
24.	`[-1,0,1]` (list)	`range(-1,2)`
25.	`[0,0,0,0]` (list)	`[0 for I in range(L[0])]`
26.	`[0,1,2,3]` (list)	`[I for I in range(L[0])]`
27.	ERROR	`[Q for I in range(L[0])]`
28.	`["C","a","t"]` (list)	`[C for C in D[8]]`
29.	`[0,0,0,0]` (list)	`[0] * L[0]`
30.	`[6.5,7.5,8.5]` (list)	`[X+I for I in range(3)]`

<2> 5 Points – Based on the earlier variable definitions, which of the following expressions are legal, which are illegal, and why?

`S[0] = "X"`	`L[0] = "X"`	`T[0] = "X"`
Illegal, Immutable	Legal	Illegal, Immutable

<3> 24 Points – (4 points each answer) What is printed out when **`Main()`** is called:

```
def F1 (Frog,Toad,Newt=4):
    print Frog - Toad + Newt
    return

def F2 (Toad,Newt,Frog=5):
    F1(Newt,Frog)
    print Toad * Frog
    return

def Main():
    Dog = 6
    Cat = 2
    F1(Dog,Cat)
    F1(5,Cat,Dog)
    F2(Dog,Dog)
    F2(3,Cat,Dog)
    return
```

Answer Line #1: **8**

Answer Line #2: **9**

Answer Line #3: **5**

Answer Line #4: **30**

Answer Line #5: **0**

Answer Line #6: **18**

<4> 20 Points – Please refer to the final version of the Image Processor code.

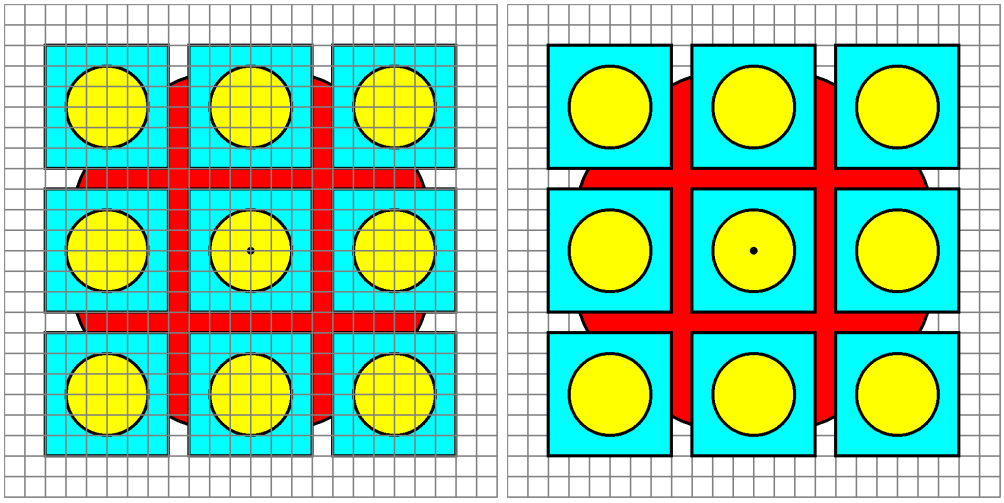
A. (15 points) Create a new pixel transform function called **`Weird_Pixel`** to be passed to **`Process`** and **`Dither`**, where the pixel's new red value will be set to 255 if the old red value is greater than the average of the old values of blue and green, and will be set to 0 otherwise. Write the complete function.

```
def Weird_Pixel (PX):
    if (getRed(PX) > (getGreen(PX) + getBlue(PX))/2):
        setRed(PX,255)
    else:
        setRed(PX,0)
    return
```

B. (5 points, ½ point each – remove ½ point each error, but do not go below zero) We want to create a new filter command, where the 3×3 matrix is all 0 except for a 5 in the center, and -1 in the top-middle, lower-left, and lower-right cells. The scale factor (the divisor) should be 2.0, and the offset should be zero. Fill in the call to the **`Filter`** function appropriately:

```
Canvas =
    Filter(Canvas,[[ 0 ,-1 , 0 ],[ 0 , 5 , 0 ],[-1 , 0 ,-1 ]],2.0, 0 )
```

<5> 16 Points – A **`Widget`** is a red circle of radius 90 pixels, with nine blocks on top, as shown (once with an alignment grid on top, and once with the grid underneath). Each grid square is 10 pixels on a side.



A **`Block`** is a cyan square of radius 30 (the radius of a square is from center-to-side, not center-to-corner); with a yellow circle of radius 20 on top of it. Fill in the blanks below to complete the drawing of a **`Widget`** centered at location <X,Y> (the center is shown with a dot). The **`Circle`** and **`Square`** functions are already provided.

```
def Circle (Canvas, X, Y, Radius, NewColor=black):   …
def Square (Canvas, X, Y, Radius, NewColor=black):   …


def Widget (Canvas, X, Y):

    def Block (X, Y):

(1 pt each)         Square (Canvas, ___X___, ___Y___, ___30___, __cyan__)

(1 pt each)         Circle (Canvas, ___X___, ___Y___, ___20___, __yellow__)

                    return


(1 pt each)     Circle (Canvas, ___X___, ___Y___, ___90___, __red__)

                for Row in range(-1,2):

                    for Column in range(-1,2):

(2 pts each)            Block (__X + Column*70__, __Y + Row*70__)

                return
```

<6> 10 Points – Examine the **`Widget`** program on the previous page, and assume that all of the blanks have been filled in correctly to draw the indicated figure. Answer the following questions:

A. How many individual calls to **`Circle`** and **`Square`** would be required if the **`Widget`** function was not designed as a hierarchical decomposition? (That is, **`Widget`** only contains calls to **`Circle`** and **`Square`**, and does not define the **`Block`** function.)

(2 points) Calls to **`Circle`**: **10**

(2 points) Calls to **`Square`**: **9**

B. (1 point, Yes or No) Can **`Block`** be called from a function outside of **`Widget`**?

NO

C. (2 points) Why doesn't **`Block`** need to be passed **`Canvas`** as a parameter?

`Block` is defined inside **`Widget`**, and **`Widget`** has **`Canvas`** passed in as a parameter, so **`Canvas`** is global to / in the environment of **`Block`**. (Accept anything reasonable that talks about **`Canvas`** being available to **`Block`** because both are inside **`Widget`**.)

E. (3 points) Can **`Block`** be moved, ***without modifications***, outside and above **`Widget`** and have **`Widget`** still work correctly? Why or why not?

NO (1 point). If **`Block`** was moved above **`Widget`**, in that case its internal reference to **`Canvas`** would cause Python to fail with a variable-not-found error (2 points). (Accept anything reasonable that talks about **`Canvas`** not being available to **`Block`**. This is the inverse problem to that in the previous question.)