CMPSCI 119
Spring 2018
Friday, April 6, 2018
Midterm #2 Solution Key
Professor William T. Verts

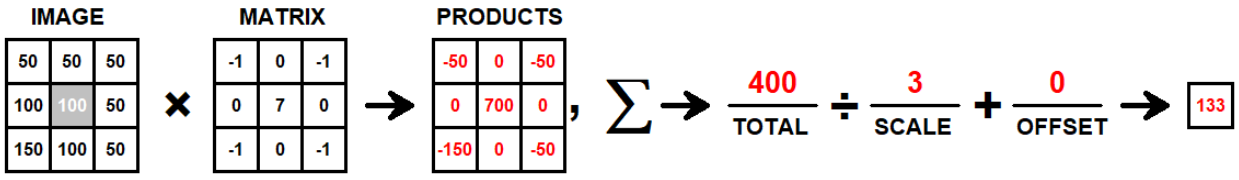
<1> 25 Points – What is the value of each expression below? Answer any 25; answer more for extra credit. Variable `Q = 3.25`, `K = 24`, `S = "FROGS, TOADS, AND NEWTS"`, `L = [4, 2, "CAT", [3,4], "DOG", 5, 1]`, and `T = (7, 2, 3, 4)`. Answer "Error" if an expression cannot be computed for any reason. 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. Thus, it is better to leave an answer blank than to guess incorrectly. Correct answers with incorrect data types are penalized ½ point each! (Don't forget quotes on strings, square brackets on lists, etc.)

1.	`12`	*int*	`K / 2`
2.	`12.0`	*float*	`K / 2.0`
3.	`13.0`	*float*	`Q * 4`
4.	`13.0`	*float*	`Q * 4.0`
5.	`23`	*int*	`len(S)`
6.	`7`	*int*	`len(L)`
7.	`4`	*int*	`len(T)`
8.	`ERROR`		`len(Q)`
9.	`"S"`	*string*	`S[4]`
10.	`"CAT"`	*string*	`L[2]`
11.	`1`	*int*	`L[-1]`
12.	`ERROR`		`T[6]`
13.	`[3,4,5,6]`	*list*	`L[3] + [5,6]`
14.	`ERROR`		`L[3] + 1`
15.	`5`	*int*	`L[0] + 1`
16.	`"CATS"`	*string*	`L[2] + "S"`
17.	`ERROR`		`L + T`
18.	`4`	*int*	`L[1] + T[1]`
19.	`"COW"`	*string*	`L[2][0] + S[2] + S[20]`
20.	`5.25`	*float*	`Q + T[1]`
21.	`[0,1,2,3]`	*list*	`range(T[-1])`
22.	`[1,4,7,10,13,16,19,22]`		`range(1,K,3)`
23.	`[-1,0,1]`		`range(-1,2)`
24.	`[]`		`range(10,1)`
25.	`[0,0,0,0]`		`[0 for N in range(4)]`
26.	`["G","O","A","T"]`		`[CH for CH in "GOAT"]`
27.	`["R","A","T"]`		`[S[N] for N in [1,9,7]]`
28.	`["F","R","O","G","S"]`		`[S[N] for N in range(5)]`
29.	`[4,5]`		`[N+1 for N in L[3]]`
30.	`[19,20,21,22,23]`		`[N for N in range(K) if N > 18]`

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

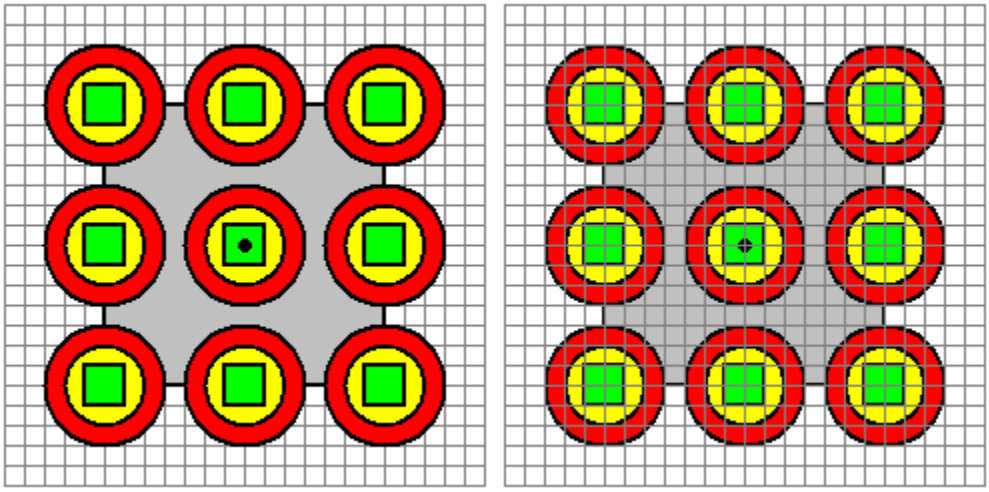
```
def Frog (A, B=3, C=5):          # Answer #1   2
    print A + B - C              #
    return                       # Answer #2   3
                                 #
def Toad (C, D, E=4):            # Answer #3   -1
    Frog(E, C, D)                #
    print E + C                  # Answer #4   6
    return                       #
                                 # Answer #5   0
def Main():                      #
    Fred = 4                     # Answer #6   4
    Sam  = 1                     #
    Mary = 3                     # Answer #7   1
    Frog(Fred)                   #
    Frog(Sam+Fred)               # Answer #8   2
    Frog(Sam, Mary)              #
    Frog(Mary, Fred, Sam)        # Answer #9   5
    Toad(Sam, Fred, Mary)        #
    Toad(Sam, Sam, Sam)          # Answer #10  8
    Toad(Fred, Mary)             #
    return                       #
```

<3> 10 Points – We are now filtering an image to enhance its focus. In the leftmost **IMAGE** grid below, the gray center box represents the current pixel under consideration somewhere in the canvas, and the white boxes around it represents its immediate neighborhood. (We are only considering the red channel, and the current pixel is in the middle of the canvas, far from any edge.) The **MATRIX** grid represents the multipliers for the "focus" filter. Compute the values for the **PRODUCTS** grid, add them up and put the result in the **TOTAL** slot, then apply the appropriate **SCALE** and **OFFSET** to get the final value for the pixel.



Scoring: ½ point for each box (10 boxes for 5 points), 3 points for the total, 1 point for the scale and 1 point for the offset.

<4> 25 Points – A **Thingie** is a gray square of side-radius 70 pixels, with nine **Blot** shapes on top, as shown (on the left with an alignment grid underneath, and on the right with the grid on top; the grids do not normally appear). Each grid square is 10 pixels on a side.



A **Blot** is a red circle of radius 30, with a yellow circle of radius 20 on top of it, and a green square of side radius 10 on top of the circles. Fill in the blanks below to complete the drawing of a **Thingie** centered at location <X,Y> (the center is shown with a dot). 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 Thingie (Canvas, X, Y):

    def Blot (X, Y):
        Circle(Canvas,  X  ,  Y  ,  30  ,  red   )
        Circle(Canvas,  X  ,  Y  ,  20  ,  yellow  )
        Square(Canvas,  X  ,  Y  ,  10  ,  green  )
        return

    Square (Canvas,  X  ,  Y  ,  70  ,  gray  )
    for Row in range(-1,2):
        for Column in range(-1,2):
            Blot (X +  Column*70  , Y +  Row*70  )
    return
```

Scoring: 1 point each for the first 16 slots, 4 points each for the last two slots, and 1 free point if there is any answer at all. Note that the last two slots can be in either order (it doesn't matter if the Blots are painted in row-major or column-major order).

<5> 5 Points – Examine the **Thingie** 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 **Thingie** function was not designed as a hierarchical decomposition? (That is, **Thingie** only contains calls to **Circle** and **Square**, and does not define the **Blot** function.)

(2 points) Calls to **Circle**: **18**

(2 points) Calls to **Square**: **10**

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

NO

<6> 15 Points – Write a complete Python function called **Check** with one integer parameter; the result returned from **Check** should be **"NEGATIVE"** if the parameter is less than zero, **"POSITIVE"** if the parameter is greater than zero, and **"ZERO"** if the parameter is exactly zero.

```
def Check(N):
    if (N < 0): return "NEGATIVE"
    elif (N > 0): return "POSITIVE"
    else: return "ZERO"
```

```
def Check(N):
    if (N < 0): Answer = "NEGATIVE"
    elif (N > 0): Answer = "POSITIVE"
    else: Answer = "ZERO"
    return Answer
```

Scoring: Accept any solution that accomplishes the desired task; two possible solutions are presented here. Remove 1 point per error (indentation, missing colons or quotes, unbalanced parentheses, use of incorrect operators, incorrect use of parameters, etc.). Note that variable names may be different.