CMPSCI 119
Spring 2017
Final Exam
Wednesday, May 10, 2017
Professor William T. Verts

NAME: _____

Question	Points	Score
1	15+10	
2	10	
3	5	
4	5	
5	10	
6	10	
7	10	
8	15	
9	10	
10	10	
TOTAL	100	

<1> 15 Points – Answer any **15** questions. Answer more for extra credit. Blank answers will be ignored, correct answers as +1 and incorrect answers as -1. In each case, show the final value assigned to the variable on the left side of the equals sign. Be careful about which variables and constants are ***integers***, which are ***floating-point numbers***, and which are ***strings***, ***lists***, ***tuples*** or ***dictionaries***! Indicate all floating-point answers with a decimal point, even if they are whole numbers, and indicate all string answers surrounded by quotes. Be careful of upper and lower case letters in strings. If a statement contains a bug, answer **`error`**. Questions are all independent of one another: for each question, assume variables have values as follows:

```
N = 91
F = 37.25
S = "The eagle has landed."
L = range(10)
P = [[4,5], [6,9], [3,1], [2,4], [7,2]]
M = [65.3, 5, (3+2j), "frog", (6,1,3), 7]
T = (5, 9, [2, 1], 6.7)
D = {2:"two", 7:"seven", 4:"four", 0:"zero"}
```

30	int	`N / 3`
30.33333…	float	`N / 3.0`
38.0	float	`F + 0.75`
37	int	`int(F)`
37.0	float	`round(F)`
91.0	float	`float(N)`
"91"	string	`str(N)`
5	int	`len(P)`
21	int	`len(S)`
error		`len(N)`
10	int	`len(L)`
2	int	`len(T[2])`
"The eagle"	string	`S[0:9]`
"."	string	`S[-1]`
error		`S[42]`
9	int	`L[-1]`
"seven"	string	`D[7]`
error		`D[9]`
"four"	string	`D[P[0][0]]`
(4+2j)	complex	`M[2]+1`
"The frog has landed."	string	`S[:4] + M[3] + S[9:]`
[0,2,4,6,8]	list	`range(0,T[1],2)`
[4,5,6,9]	list	`P[0] + P[1]`
["t","w","o"]	list	`[CH for CH in D[2]]`
[0,0,0,0,0,0,0]	list	`[0] * M[-1]`

<2> 10 Points – What is printed when the blank contains **`if`**? What is printed when the blank contains **`while`**?

```
I = 0
N = 5
______ (I < N):
    print I
    I = I + 1
```

Blank is `if`	Blank is `while`
0	0
	1
	2
	3
	4

5 points for each answer or set of answers.
Remove 2 points for the `while` section if the loop stops early or late.

<3> 5 Points – A function definition looks like this (note the default values):

```
def Strange(A=1.0, B=7, C=14, D=99, E="99"):
```

In each of the following function calls, list the actual values assigned to the five formal parameters when the function starts running (fill in all 25 of the little boxes):

Function Call	A	B	C	D	E
`Strange()`	`1.0`	`7`	`14`	`99`	`"99"`
`Strange("X",6.4)`	`"X"`	`6.4`	`14`	`99`	`"99"`
`Strange(3,"Bat"+"Man",[2])`	`3`	`"BatMan"`	`[2]`	`99`	`"99"`
`Strange(9,3,4,range(2))`	`9`	`3`	`4`	`[0,1]`	`"99"`
`Strange(9,3,4,1,2)`	`9`	`3`	`4`	`1`	`2`

One point each line. Remove ½ point for 1-3 errors, remove full point for 4-5 errors.

<4> 5 Points – **`S`** is of type string and **`L`** is of type list. Initially, **`S=""`** and **`L=[]`**. What are the values of **`S`** and **`L`** after the following statements are each executed five times?

```
S = S + "X"          L = L + ["X"]
```

```
"XXXXX"              ["X","X","X","X","X"]
```

`S` must be a single string (2 points), and `L` must be a list of strings (2 points). Remove 1 point if either contains the wrong number of `X`.

<5> 10 Points – Complete the function below to compute if string **S** is a palindrome (the same text backwards as forwards). For example, **IsPalindrome("MADAM")** returns **True**, but **IsPalindrome("FROG")** returns **False**. Assume **S** contains only uppercase letters; don't worry about differences in upper and lower case or special characters.

```
def IsPalindrome(S):
    Result = True
    for I in range(len(S)/2):
        if (S[I] <> S[len(S)-1-I]):
            Result = False
    return Result
```

```
def IsPalindrome(S):
    for I in range(len(S)/2):
        if (S[I] <> S[len(S)-1-I]):
            return False
    return True
```

```
def IsPalindrome(S):
    I1 = 0
    I2 = len(S)-1
    while (I1 < I2):
        if (S[I1] <> S[I2]):
            return False
        I1 = I1 + 1
        I2 = I2 - 1
    return True
```

Here are three possible solutions to the palindrome problem, but not all possible answers. The final return must return some value, either a bool variable or a bool constant (it cannot be just return by itself). Note that all loops will run correctly (if inefficiently) if they scan the whole list instead of just halfway through, so allow this in their answers. For any chosen method, remove points for each error, up to 10.

<6> 10 Points – What is printed by the following code when **`Main`** is run?

```
def FNA(P,Q=1):
    return P-Q

def FNB(Z,R,Q):
    print R + Q
    print FNA(Z)
    return

def Main():
    print FNA(5,2)
    FNB(6,1,3)
    FNB(5,4,2)
    return
```

Two points each number

```
3    FNA(5,2) → 3
4    FNB(6,1,3) → print 1+3 = 4
5    FNB(6,1,3) → FNA(6) → 5
6    FNB(5,4,2) → print 4+2 = 6
4    FNB(5,4,2) → FNA(5) → 4
```

<7> 10 Points – Complete the function below to read in the contents of the text file into a list of strings, strip the terminating line-break off the end of each string, then print the strings from the list in reverse order (last string first, first string last).

```
def GetStuff (N):
    Filename = pickAFolder() + "Data.txt"
    Infile = open(Filename,"r")

    X = Infile.readlines()                  # See page 264
    Infile.close()                          # in the Companion
    X = [C.rstrip("\n") for C in X]         # on reading files

    for I in range(len(X)-1,-1,-1):
        print X[I]

    return
```

2 points for each line. The first three lines should be pretty much verbatim, as we covered it in class and it is in the Companion on page 264.

+2 points extra credit for any comment about variable N being useless.
-2 points for using N in the answer.

<8> 15 Points – For each pixel in **Canvas**, dither just the red values as follows: if the red value of the pixel is greater than 128 make it go to 255, otherwise make it go to 0, saving the difference between the old value of red and the new value of red in an error variable. Make half of the error add onto the right (east) neighbor and the other half of the error add onto the bottom (south) neighbor. Be careful of the edges of the canvas.

```
def DitherRed(Canvas):
    for Y in range(getHeight(Canvas)):
        for X in range(getWidth(Canvas)):

            PX = getPixel(Canvas,X,Y)
            OldRed = getRed(PX)
            if (OldRed > 128):
                NewRed = 255
            else:
                NewRed = 0
            setRed(PX,NewRed)
            Error = OldRed - NewRed

            if (X < getWidth(Canvas)-1):
                PX = getPixel(Canvas,X+1,Y)
                setRed(PX, getRed(PX) + Error/2)

            if (Y < getHeight(Canvas)-1):
                PX = getPixel(Canvas,X,Y+1)
                setRed(PX, getRed(PX) + Error/2)

        repaint(Canvas)
    return
```

Score as 5 points for the replacement of red in the current pixel and computation of the error, 5 points for the east neighbor, and 5 points for the south neighbor. Computation of the error is correct as shown but allow `NewRed - OldRed` (it's wrong, but a reasonable guess). Neighbor code must check for edge cases. Assess partial credit within each section as appropriate.

<9> 10 Points – For each of the following problems assume that **`import random,time`** has been placed at the beginning of the program. Each answer is a single Python expression.

A. How do I sleep for a random integer number of seconds between 1 and 10 seconds?

```
time.sleep(random.randrange(1,11))        -or-
time.sleep(random.randint(1,10))
```

B. How do I pick a random integer between -25 and +25 (inclusive)?

```
random.randrange(-25,26)              -or-
random.randint(-25,25)
```

C. How do I create a list of ten random integer values between 1 and 6 (inclusive)?

```
[random.randrange(1,7) for I in range(10)]        -or-
[random.randint(1,6) for I in range(10)]
```

D. How do I pick a random value which is either 4.6, 9.2, or 10.79?

```
random.choice([4.6, 9.2, 10.79])
```

E. How do I create a color (using **`makeColor`**) where the red value is a random int between 0 and 255 (inclusive), the green value is a random int between 128 and 255 (inclusive) and the blue value is simply 64?

```
makeColor(random.randrange(256),random.randint(128,255),64)
makeColor(random.randint(0,255),random.randint(128,255),64)
```

Score as 2 points for each section. Accept any correct solution, even variations not shown here as long as they work. Assess -1 per section for minor errors.

<10> 10 Points – Refer to pages 320-321 of the Companion (3D Orthographic Projection) for this question. I have called **`SetOrigin2D([200,400])`** to establish the location on the canvas of the center of the 3D axes **`[0,0,0]`**, and have called **`SetScale2D(10.0)`**. What 2D point is returned from calling **`Project3D([4,5,2])`**? That is, where is the location of the corresponding pixel on the canvas? Remember that **`Sine30`** is exactly 0.5, and to simplify your computations assume that **`Cosine30`** is exactly 0.8 instead of its true value (0.866025…).

```
From Project3D:
    X = Origin2D[0] + (P3D[0] + P3D[2] * Cosine30) * Scale2D
    Y = Origin2D[1] - (P3D[1] + P3D[2] * Sine30) * Scale2D

Substituting Values from this problem:
    X = 200 + (4 + 2 * 0.8) * 10.0
    Y = 400 - (5 + 2 * 0.5) * 10.0

Reducing:
    X = 200 + (5.6) * 10.0  =  200 + 56.0  =  256.0
    Y = 400 - (6.0) * 10.0  =  400 - 60.0  =  340.0

Answer = [256.0, 340.0]
```

(Using correct cosine value gives **`[257.32, 340.0]`**)

Score as 5 points for X and 5 points for Y. Given partial credit as you see fit, but be consistent.