CMPSCI 119
Spring 2019
Wednesday, April 17, 2019
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)

```
N = 21
F = 4.5
S = "SNAKE"
L = [2, "FROG", 1, 3]
T = (6, 2, [1,4,1], 9.0, 3)
D = {3:"TOAD", 4: "NEWT", 1:"DOG"}
```

1.	`10.5`	`N / 2`
2.	`10`	`N // 2`
3.	`9.0`	`F * 2`
4.	`9`	`int(F * 2.0)`
5.	`10.0`	`float(N // 2)`
6.	`5`	`len(S)`
7.	`4`	`len(L)`
8.	`ERROR`	`len(N)`
9.	`3`	`len(T[2])`
10.	`3`	`len(D)`
11.	`"FROGS"`	`L[1] + S[0]`
12.	`"TOAD"`	`D[T[-1]]`
13.	`"SNAKEFROG"`	`S + L[1]`
14.	`ERROR`	`D[2]`
15.	`[1,4,1]`	`T[L[0]]`
16.	`ERROR`	`L[T[0]]`
17.	`2`	`L[N % 3]`
18.	`"NEWT"`	`D[T[2][1]]`
19.	`[0,1,2]`	`range(L[-1])`
20.	`[1,7,13,19]`	`range(1,N,T[0])`
21.	`[]`	`range(N,T[4])`
22.	`[0,1,2,3]`	`range(int(F))`
23.	`[0,1,2,3,4]`	`range(len(S))`
24.	`[0,0,0,0,0]`	`[0 for X in range(len(T))]`
25.	`[]`	`[X for X in range(N,5)]`
26.	`["S","N","A","K","E"]`	`[X for X in S]`
27.	`[0,1,2,3]`	`[X for X in range(len(L[1]))]`
28.	`[65,66,67]`	`[ord("A")+X for X in range(3)]`
29.	`["A","B","C"]`	`[chr(65+X) for X in range(3)]`
30.	`[0,1,4]`	`[X*X for X in range(3)]`

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

```
def Chalk(A,B,C=5):                    # Line  1   1
    global Blackboard
    Blackboard = A + B + C             # Line  2   -3
    print (Blackboard)
    return                             # Line  3   6

def Erasers(X,Y,Z):                    # Line  4   0
    global Blackboard
    Q = Y + Blackboard                 # Line  5   9
    Chalk(Z,Q+X)
    Blackboard = Q                     # Line  6   13
    print (Blackboard)
    return                             # Line  7   11

def Main():                            # Line  8   3
    global Blackboard
    Blackboard = 1                     # Line  9   8
    print (Blackboard)
    Chalk(2,-5, 0)                     # Line 10   3
    Erasers(5,3,-4)
    Chalk(3,1)
    Erasers(-4,2,1)
    Chalk(1,1,1)
    Erasers(0,0,0)
    return
```

<3> 10 Points – Write a ***for-loop*** using **`Index`** as the counter variable, which starts at 19, goes up to but does not include 31, and counts by 2s. The payload of the loop is to call the **`Weird`** function with **`Index`** as its (only) parameter.

```
for Index in range(19,31,2):
    Weird(Index)
```

Scoring:

In the following sections remove -1 per error from the given total for minor infractions:

- 3 points for **`for ___ in _____:`**
- 2 points for **`range(__,__,__)`**
- 2 points for **`19`**, **`31`**, and **`2`** in proper slots
- 2 points for **`Weird(____)`**
- 1 point for **`Index`** as the argument to **`Weird`**.

Also remove points for the following problems, but do not go below zero:

- -1 point for **`def Weird(Index)`**
- -1 point for other extraneous code

It is allowed to write the code as one line:

```
for Index in range(19,31,2): Weird(Index)
```

<4> 10 Points – Now write a ***counter-loop*** using **`Index`** as the counter variable, which starts at 19, goes up to but does not include 31, and counts by 2s. The payload of the loop is to call the **`Weird`** function with **`Index`** as its (only) parameter.

```
Index = 19
while (Index < 31):
    Weird(Index)
    Index = Index + 2
```

Scoring:

In the following sections remove -1 per error from the given total for minor infractions:

- 2 points for **`Index = 19`**
- 3 points for **`while (Index < 31):`**
- 2 points for **`Weird(____)`**
- 1 point for **`Index`** as the argument to **`Weird`**.
- 2 points for **`Index = Index + 2`** *or* **`Index += 2`**

Also remove points for the following problems, but do not go below zero:

- -1 point for **`def Weird(Index)`**
- -1 point for other extraneous code

<5> 20 Points – I have an image-processing program that deals with pixels as three-element lists containing its red, green, and blue values (between 0 and 255), in that order. For example, the list **`[0,0,0]`** is black, **`[255,0,0]`** is red, **`[0,255,0]`** is green, **`[0,0,255]`** is blue, **`[255,255,0]`** is yellow, **`[255,255,255]`** is white, etc. The gray value of a pixel is the average of the red, green, and blue values. Complete the function below to return the black pixel value if the gray value of **`PX`** is less than 128 and return the white pixel value otherwise.

Most Expected Answer:

```
def Process(PX):          # PX is an [R,G,B] list
    Average = (PX[0] + PX[1] + PX[2]) / 3
    if Average < 128:
        Result = [0,0,0]
    else:
        Result = [255,255,255]
    return Result
```

```
def Process(PX):          # PX is an [R,G,B] list
    Total = 0
    for I in range(3): Total = Total + PX[I]
    Average = Total / 3
    if Average < 128:
        Result = [0,0,0]
    else:
        Result = [255,255,255]
    return Result
```

```
def Process(PX):          # PX is an [R,G,B] list
    Total = 0
    I = 0
    while (I < len(PX)):
        Total = Total + PX[I]
        I = I + 1
    Average = Total / len(PX)
    if Average < 128:
        Result = [0,0,0]
    else:
        Result = [255,255,255]
    return Result
```

Least Expected Answer:

```
def Process(PX):          # PX is an [R,G,B] list
    Average = (PX[0] + PX[1] + PX[2]) / 3
    Value = int(Average) // 128 * 255    # either 0 or 255
    return [Value,Value,Value]
```

Scoring:

Any of the given approaches is fine. Each of the following sections are worth some number of points. In each case remove points for infractions such as syntax errors, failed methods, using **<=** instead of **<**, returning tuples instead of lists, etc., but do not go below zero for that section.

Section #1: Compute the average gray level (9 points)

The idea of computing the gray level is to add the three elements of the list defined by **PX** and divide the total by 3. This can be done in a number of ways, from a specific solution that assumes **PX** has exactly 3 elements to a more general solution that works based on the length of **PX**. I don't care which approach is taken.

Section #2: Compute the correct return value (6 points)

The two possible return values are black **[0,0,0]** or white **[255,255,255]**. There should be some way of determining one or the other, usually via an **if** statement.

Section #3: Return the correct value (5 points)

It is possible that some students will return the correct value as soon as it is found, but the problem shows that a value is required on the single final **return** statement, which means that it must return a value computed earlier.

-2 for embedding the **return** into the **if**, as in:

```
if (Average < 128):
    return [0,0,0]
else:
    return [255,255,255]
```

<6> 10 Points – Write a ***complete function*** called **`Thingie`**, with one string parameter containing a file name, that opens a file for writing with the value of the parameter as its name, writes the letters from **`A`** to **`Z`** to the file one per line, and closes the file.

Most Expected Answer:

```
def Thingie(Filename):
    Handle = open(Filename,"w")
    for I in range(65,91):
        Handle.write(chr(I) + "\n")
    Handle.close()
    return

def Thingie(Filename):
    Handle = open(Filename,"w")
    I = 65
    while (I < 91):
        Handle.write(chr(I) + "\n")
        I = I + 1
    Handle.close()
    return

def Thingie(Filename):
    Handle = open(Filename,"w")
    for I in range(ord("A"), ord("Z")+1):
        Handle.write(chr(I) + "\n")
    Handle.close()
    return

def Thingie(Filename):
    Handle = open(Filename,"w")
    I = ord("A")
    while (I <= ord("Z")):                  # Note the <=
        Handle.write(chr(I) + "\n")
        I = I + 1
    Handle.close()
    return
```

Least Expected Answer:

```
def Thingie(Filename):
    Letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"     # or
    Letters = [chr(I) for I in range(65,91)]   # or
    Letters = string.ascii_uppercase  # Must import string

    Handle = open(Filename,"w")
    for I in Letters:
        Handle.write(I + "\n")
    Handle.close()
    return
```

Scoring:

Any of the given approaches is fine. Remove 1 point per infractions such as syntax errors, failed methods, forgetting the `"\n"` line breaks, miscomputing the range of values, etc., but do not go below zero.

<7> 5 Points – What is wrong with the following code fragment?

```
N = 0
while (N < 10):
    print (N)
    N + 1
```

Last statement must be `N = N + 1`

Scoring: All or nothing.