

CMPSCI 119
Fall 2018
Friday, October 10, 2018
Midterm #1
Solution Key
Professor William T. Verts

<1> 20 Points – Do any 20; do more for extra credit. Correct answers are worth +1 point, blank answers are worth 0 points, but wrong answers are incur a -½ point penalty; if you don't know an answer, leaving it blank is usually better than a bad guess. The following statements have all been executed:

```
Frodo = 4.5
Bilbo = 25
Sauron = [3, "EYE", 2.5, [7,1,5,9], 17]
Arwen = "EVENSTAR"
Gollum = (5, 2, [6,9,4], 8)
```

Show the computed result for each problem; all are independent of one another. Indicate where a computation fails because of some form of error. Be careful about the *type* of the result, particularly int, float, long, bool, and complex types, and put proper quotes around string results, square brackets around lists, and parentheses around tuples.

	Question	Answer
1.	Frodo * 2	9.0 float
2.	Frodo * 2.0	9.0 float
3.	Bilbo / 2	12 int
4.	Bilbo / 2.0	12.5 float
5.	len(Sauron)	5 int
6.	len(Gollum[2])	3 int
7.	len(Bilbo)	ERROR len(int)
8.	len(Sauron[1])	3 int
9.	len(Sauron[3])	4 int
10.	Sauron[-1]	17 int
11.	Sauron[3][2]	5 int
12.	Sauron[3]+[Gollum[0]]	[7,1,5,9,5] list
13.	Sauron[3]+Gollum[1]	ERROR list+int
14.	Sauron[3]+Gollum[2]	[7,1,5,9,6,9,4] list
15.	Sauron+Gollum	ERROR list+tuple
16.	Sauron[1]+"S"	"EYES" string
17.	[Gollum[0], Bilbo, Arwen[1]]	[5,25,"V"] list
18.	(Frodo, Bilbo, Sauron[1])	(4.5,25,"EYE") tuple
19.	Arwen[10]	ERROR 10>7
20.	Arwen[-1]	"R" string
21.	range(Gollum[0])	[0,1,2,3,4] list
22.	range(len(Gollum))	[0,1,2,3] list
23.	range(3, Bilbo, 4)	[3,7,11,15,19,23] list
24.	[X*2 for X in range(5)]	[0,2,4,6,8] list
25.	[0 for X in range(Sauron[0])]	[0,0,0] list

<2> 10 points – Show what is printed by the following code fragment for each given case:

<pre>try: X = Y / Z except: X = -1 print X</pre>	<pre>Y=4 , 2 Z=2</pre>
	<pre>Y=4.0 , 2.0 Z=2</pre>
	<pre>Y=4.0 , -1 (divide by zero error) Z=0</pre>
	<pre>Y=2 , 20.0 Z=0.1</pre>
	<pre>Y=2 , -1 (type error in division) Z="1"</pre>

<3> 15 Points – Write a while-loop with an integer counter variable called **Count**, where **Count** starts at 13, is increased by 5 after each pass through the loop, and then ends if the variable ever reaches 47 or larger. The “payload” of the loop is to print **Count**.

```
Count = 13           # 3 points
while (Count < 47): # 5 points
    print Count      # 2 points
    Count = Count + 5 # 5 points Can be: Count += 5
```

For each line, start with the given points and remove 1 point per syntax error. Remove 1 point overall if the print and the increment are reversed. Do not go below zero

<4> 20 Points – Show what is printed out as the result from calling **Main()** (four lines total):

```
def Frog (Cat,Dog,Bat=1):
    Newt = Cat - Dog
    return Newt * Bat

def Toad (Dog,Bunny):
    return Frog (Bunny,Dog)

def Newt (Cat,Bunny,Snake):
    return Toad (Cat,Snake) + Bunny

def Main():
    print Frog (5,2,3)      # Answer #1: 9 (5 points)
    print Toad (4,8)       # Answer #2: 4 (5 points)
    print Newt (3,2,4)     # Answer #3: 3 (5 points)
    print Frog (6,-1)     # Answer #4: 7 (5 points)
    return
```

- <5> 15 Points – What is printed when the user enters **Dog** at the first prompt, and enters **3** at the second prompt?

```
S = raw_input("Enter string to repeat --- ")
N = input("Enter number of repeats --- ")
Answer = ""
for I in range(N): Answer = Answer + S + str(I)
print Answer
```

Dog0Dog1Dog2

Remove 3 points if the range is more than or less than [0,1,2]

Remove 3 points if the Dog0, Dog1, and Dog2 are on different lines.

Remove 3 points if numbers appear to the left of each Dog.

Remove 3 points if Dog is not capitalized as shown.

Remove 3 points if there are any spaces between Dog and the numbers.

- <6> 15 Points – Complete the following function to return **-1** if integer parameter **N** is less than zero, return **+1** if **N** is greater than zero, and return **0** if **N** is exactly zero. Your solution must **NOT** contain any **print** statements.

```
def MyFunction(N):
    if (N < 0): return -1
    if (N > 0): return +1
    return 0
```

-or-

```
def MyFunction(N):
    if (N < 0): Result = -1
    if (N > 0): Result = +1
    if (N == 0): Result = 0
    return Result
```

-or-

```
def MyFunction(N):
    if (N < 0): Result = -1
    elif (N > 0): Result = +1
    else: Result = 0
    return Result
```

Remove 1 point per syntax error (indentation, capitalization, using = instead of ==, etc.).

Remove 5 points per major design error.

Spacing after the start of a statement is irrelevant.

<7> 5 Points – Short Answer – How are the following statements all similar? You may use the back of this page to answer.

```
N = N + 1      # where N is an int  
S = S + "1"   # where S is a string  
L = L + [1]   # where L is a list  
T = T + (1)   # where T is a tuple
```

All four take a variable's old value, make a change to that value, then replaces the variable's value with the new result. (The last three append the new value onto the end of the string, list, or tuple, respectively.)

Accept anything reasonable.