CMPSCI 119
Spring 2016
Monday, February 22, 2016
Midterm #1 Solution Key
Professor William T. Verts
<1> 15 Points – Do any 15; do more for extra credit. In this problem assume that all of the statements $X = 5$, $S1 = "ABC", S2 = "MNO", L = [5, 3, 8, 2, 4, 6]$, and $T = (X, X, [4, 5], S1+S2, 5.6, 9)$ have been executed. Show the computed result for each statement (all are independent of one another). Indicate where the computation fails because of some form of error. Be careful about the type of the result, particularly integers, floats, long integers, bools, and complexes, and put proper quotes around string results, square brackets around lists, and parentheses around tuples.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $X + 3$</td>
<td>8 (must be int)</td>
</tr>
<tr>
<td>2. $X / 2$</td>
<td>2 (must be int)</td>
</tr>
<tr>
<td>3. $X * 2.0$</td>
<td>10.0 (must be float)</td>
</tr>
<tr>
<td>4. $X + L[2]$</td>
<td>13 (must be int)</td>
</tr>
<tr>
<td>5. len(L)</td>
<td>6</td>
</tr>
<tr>
<td>6. $L[-1]$</td>
<td>6 (last item of L)</td>
</tr>
<tr>
<td>7. $L[6]$</td>
<td>Error (off end of list)</td>
</tr>
<tr>
<td>8. len(T)</td>
<td>6</td>
</tr>
<tr>
<td>10. $T[3][2:5]$</td>
<td>&quot;CMN&quot; (characters 2…4)</td>
</tr>
<tr>
<td>11. $T[2] + [9]$</td>
<td>[4, 5, 9] (list + list)</td>
</tr>
<tr>
<td>12. $T[2] + (9, 6)$</td>
<td>Error (list + tuple)</td>
</tr>
<tr>
<td>14. $S2[1:]$</td>
<td>&quot;NO&quot; (characters 1…)</td>
</tr>
<tr>
<td>15. round($T[4]$)</td>
<td>6.0 (still a float)</td>
</tr>
<tr>
<td>16. $L[0:3]$</td>
<td>[5, 3, 8]</td>
</tr>
<tr>
<td>17. range($X$)</td>
<td>[0, 1, 2, 3, 4]</td>
</tr>
<tr>
<td>18. [Z*X for Z in range(3)]</td>
<td>[0, 5, 10]</td>
</tr>
<tr>
<td>19. [0 for Q in L]</td>
<td>[0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>20. [Q+1 for Q in L]</td>
<td>[6, 4, 9, 3, 5, 7]</td>
</tr>
</tbody>
</table>

1 point each. Ignore blank or incorrect answers, and give credit only for correct answers. Give ½ partial credit where warranted (probably not many places).
<2> 20 points – Write a complete, well-formed function, called Roller, with no parameters, that returns a random integer from the following set: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

```python
import random

def Roller ():
    return random.randrange(1,11)

-or-

def Roller ():
    return int(random.random()*10) + 1

-or-

def Roller ():
    Choices = [1,2,3,4,5,6,7,8,9,10]
    return random.choice(Choices)
```

5 points for the `import random` statement (can be inside the function)
5 points for `def Roller():` (-1 per syntax error)
5 points for any one of the correct computations (-1 per syntax error)
5 points for returning the value as the result of the function

<3> 10 Points – Completely rewrite the following code fragment to perform exactly the same task using a for-loop instead of a while-loop:

```python
P = 5
while (P < 70):
    print P
    P = P + 3

for P in range(5,70,3):
    print P

-or-

for P in range(5,70,3): print P
```

4 points for `for P in _____:` (-1 per syntax error)
4 points for the correct `range(5,70,3)` (-1 per incorrect number)
2 points for the `print P` statement
15 Points – Show what is printed out as the result from calling `Main()`

```python
def F1(A,G,R):
    X = A + G
    return X * 2 + R

def F2(X,R,W):
    return F1(R,X+X,W)

def F3(G,A,X):
    return F2(A+1,X,G)

def Main():
    print F1(4,5,2)  # 20
    print F3(2,6,3)  # 36
    print F2(5,3,2)  # 28
```

5 Points each answer.

\[ \text{F1}(4, 5, 2) \rightarrow \text{F1}(A=4, G=5, R=2) \rightarrow \]
\[ X = 9 \rightarrow \]
\[ \text{return } 9 \times 2 + 2 \rightarrow \]
\[ 20. \]

\[ \text{F3}(2, 6, 3) \rightarrow \text{F3}(G=2, A=6, X=3) \rightarrow \]
\[ \text{calls F2}(7, 3, 2) \rightarrow \text{F2}(X=7, R=3, W=2) \rightarrow \]
\[ \text{calls F1}(3, 14, 2) \rightarrow \text{F1}(A=3, G=14, R=2) \rightarrow \]
\[ X = 17 \rightarrow \]
\[ \text{return } 17 \times 2 + 2 \rightarrow \]
\[ 36. \]

\[ \text{F2}(5, 3, 2) \rightarrow \text{F2}(X=5, R=3, W=2) \rightarrow \]
\[ \text{calls F1}(3, 10, 2) \rightarrow \text{F1}(A=3, G=10, R=2) \rightarrow \]
\[ X = 13 \rightarrow \]
\[ \text{return } 13 \times 2 + 2 \rightarrow \]
\[ 28. \]
15 Points – The following code contains both syntax errors and run-time errors. Locate each one and indicate what the correction(s) should be.

```python
def Signum(N):
    if (N < 0): Result = -1
elseif (N > 0): Result = 1
else: Result = 0
return

def PrintAnswer(N):
    X = float(Signum(N))
    print "The signum of ", N, " is ", X
    return

def Main():
    PrintAnswer(input("Enter a number "))
    return
```

There are three syntax errors (worth 3 points each) and one run-time error (worth 6 points) in this program. As is, when Python attempts to load the program, the error message shown is:

There is something wrong with the text of the file you had me try to load.
You may have not have as many closing parentheses as opening parentheses, left the ending quote off of a string, or tried to use a Jython keyword (if, def, etc...) as a function.

This does not narrow it down much. Fixing the missing right parenthesis gives a different error:

Invalid syntax.
Your code contains at least one syntax error, meaning it is not legal Jython.
The error is on line 3.

This error is complaining about the `elseif`. Fixing that gives the same error message again, but this time on line 10. That is the mismatched quote on the string: " is ' (either the ' could be replaced by a ", or the " could be replaced by a '). This fixes all the syntax errors.

Running the program will cause it to fail on the `X = float(Signum(N))` line, but that is not where the error is actually located. The `Signum` function computes an answer in variable `Result`, but it discards that answer and returns the special Python value `None` instead, which cannot be floated. Returning `Result` fixes everything.
25 Points – The image on the right is 290×160 pixels in size, and contains twelve rectangles as shown (the grid lines have a 10-pixel spacing and are there for reference only; they should not appear in the image). Complete the following code to create the appropriate canvas and paint the twelve rectangles in the positions shown.

```python
def Main():
    Canvas = makeEmptyPicture(290, 160)
    show(Canvas)

    for Y in range(10, 111, 50):
        for X in range(10, 221, 70):
            addRect(Canvas, X, Y, 60, 40, black)

    repaint(Canvas)
    return
```

The `makeEmptyPicture`:
- 1 point for the width (290)
- 1 point for the height (160)

The `for Y` (or `while Y`) loop:
- 2 points for basic loop syntax: `for variable in range(__):`
- 2 points for the correct starting value (10)
- 2 points for the correct ending value (may be 111 through 160)
- 2 points for the correct step value (50)

The `for X` (or `while X`) loop:
- 2 points for basic syntax: `for variable in range(__):`
- 2 points for the correct starting value (10)
- 2 points for the correct ending value (may be 221 through 290)
- 2 points for the correct step value (70)

The `addRect` statement (allowed to be an `addRectFilled` instead):
- 3 points for basic syntax: `addRect(Canvas, X, Y, __, __)`
- 2 points for rectangle width (60)
- 2 points for rectangle height (40)

The `black` color parameter is allowed to be omitted.
Alternate Solutions:

```python
for X in range(10, 221, 70):
    for Y in range(10, 111, 50):
        addRect(Canvas, X, Y, 60, 40, black)

-or-

for Y in range(10, 111, 50):
    for X in range(10, 221, 70):
        addRect(Canvas, X, Y, 60, 40)

-or-

Y = 10
while (Y <= 110):
    X = 10
    while (X <= 220):
        addRect(Canvas, X, Y, 60, 40, black)
        X = X + 70
    Y = Y + 50

-or-

Y = 10
while (Y < 111):
    X = 10
    while (X < 221):
        addRect(Canvas, X, Y, 60, 40, black)
        X = X + 70
    Y = Y + 50

-or-

X = 10
while (X <= 220):
    Y = 10
    while (Y <= 110):
        addRect(Canvas, X, Y, 60, 40, black)
        Y = Y + 50
    X = X + 70
```