20 Points – What is the value of each expression below? Answer any 20; answer more for extra credit. Variable $L = [3, 9, \text{"OSTRICH"}, [8, 2], 6.7]$

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$32$ (integer multiplication)</td>
<td>$4 \times 8$</td>
</tr>
<tr>
<td>$0$ (integer division)</td>
<td>$4 / 8$</td>
</tr>
<tr>
<td>$0.5$ (floating point division)</td>
<td>$4 / 8.0$</td>
</tr>
<tr>
<td>$32L$ (long integer multiplication)</td>
<td>$4L \times 8L$</td>
</tr>
<tr>
<td>&quot;DOG&quot; (string concatenation)</td>
<td>&quot;D&quot; + $L[2][0]$ + &quot;G&quot;</td>
</tr>
<tr>
<td>$[2, 3, 4, 5]$ (list concatenation)</td>
<td>$[2, 3] + [4, 5]$</td>
</tr>
<tr>
<td>$(2, 3, 4, 5)$ (tuple concatenation)</td>
<td>$(2, 3) + (4, 5)$</td>
</tr>
<tr>
<td>$5$ (int, int, string, list, float)</td>
<td>$\text{len}(L)$</td>
</tr>
<tr>
<td>$3$ (first item in list)</td>
<td>$L[0]$</td>
</tr>
<tr>
<td>&quot;R&quot; (4\textsuperscript{th} character in OSTRICH)</td>
<td>$L[2][3]$</td>
</tr>
<tr>
<td>$2$ (2\textsuperscript{nd} item in list [8, 2])</td>
<td>$L[3][1]$</td>
</tr>
<tr>
<td>$6.7$ (last item in list)</td>
<td>$L[-1]$</td>
</tr>
<tr>
<td>$7$ (length of OSTRICH)</td>
<td>$\text{len}(L[2])$</td>
</tr>
<tr>
<td>&quot;H&quot; (last character in OSTRICH)</td>
<td>$L[2][-1]$</td>
</tr>
<tr>
<td>$12$ (3 + 9)</td>
<td>$L[0] + L[1]$</td>
</tr>
<tr>
<td>$2$ (last item in list [8, 2])</td>
<td>$L[-2][-1]$</td>
</tr>
<tr>
<td>$[8, 2, 4, 5]$ ([8, 2] + [4, 5])</td>
<td>$L[3] + [4, 5]$</td>
</tr>
<tr>
<td>[&quot;O&quot;, &quot;S&quot;, &quot;T&quot;, &quot;R&quot;, &quot;I&quot;, &quot;C&quot;, &quot;H&quot;]</td>
<td>$[X \text{ for } X \text{ in } L[2]]$</td>
</tr>
<tr>
<td>$[0, 0, 0, 0, 0]$ (See note below)</td>
<td>$[0 \text{ for } X \text{ in } \text{range(}$\text{len}(L)$\text{)}$</td>
</tr>
<tr>
<td>$[0,1,2,3,4]$ [X for X in range(5)]</td>
<td>$[X \text{ for } X \text{ in } \text{range(5)}$</td>
</tr>
<tr>
<td>$[1,2,3]$ [X+1 for X in range(3)]</td>
<td>$[X+1 \text{ for } X \text{ in } \text{range(3)}$</td>
</tr>
<tr>
<td>$[-3,0,3]$ [X*3 for X in range(-1,2)]</td>
<td>$[X*3 \text{ for } X \text{ in } \text{range(-1,2)}$</td>
</tr>
<tr>
<td>$[-3,0,3]$ range(-3,4,3)</td>
<td>range(-3,4,3)</td>
</tr>
<tr>
<td>$[3,0,-3]$ range(3,-4,-3)</td>
<td>range(3,-4,-3)</td>
</tr>
<tr>
<td>[] (empty list, range fails)</td>
<td>range(3,10,-1)</td>
</tr>
</tbody>
</table>

The second list comprehension had an error where the intended \texttt{range(...)} was missing, as shown in red. If students identified this as an error, give full credit.

Scoring: Ignore any incorrect or blank answers. Give +1 point for every correct answer. Assess -$\frac{1}{2}$ point penalty for identifying integers as floats, missing $L$ on long integers, missing "." symbols on strings, missing $[...]$ on lists, or missing $(...)$ on tuples.
20 Points – (10 points each):

A. Rewrite the following code fragment to use an explicit \texttt{for}-loop.

\begin{verbatim}
Result = []
Z = 6
while (Z <= 50):
    Result = Result + [Z]
    Z = Z + 3

Result = []
for Z in range(6,51,3):
    Result = Result + [Z]
\end{verbatim}

B. Now rewrite the code fragment as a \textit{list comprehension}.

\begin{verbatim}
Result = [Z for Z in range(6,51,3)]
\end{verbatim}

Notice that in the \texttt{while}-loop the presence of the \texttt{<=} symbol. Normally, this operator is simply a \texttt{<} symbol, which closely matches how the \texttt{range} statement works. Since this is a \texttt{<=}, the corresponding range limit has to be increased from 50 to 51.

Scoring: Start with 10 points for each problem where an answer is present (zero points for blank answers). Accept any answer that works and follows the basic guidelines of the questions (that is, determine if the answers generate the correct list, even if they are not quite what I’ve given here).

Remove points as follows for part A:
-1 for using \texttt{range(6,50,3)}
-1 for \texttt{range(51)} or \texttt{range(51,3)} or \texttt{range(6,51)}
-1 for omitting the \texttt{Result = []} statement
-1 for omitting the \texttt{[]} around \texttt{[Z]} (that is, \texttt{Result = Result + Z})
-1 for any and all capitalization errors (FOR, result, Range, etc.)
-1 per additional syntax error, up to 5

Remove points as follows for part B:
-1 for using a different range from part A
-1 for omitting the \texttt{Result = portion of the expression}
-1 for any and all capitalization errors (FOR, result, Range, etc.)
-1 per additional syntax error, up to 5
20 Points – What is printed out when `Main()` is called?

```python
def Apple (M, N):
    P = M + N
    print P
    return

def Pear (P, Z, M):
    Apple (P, M)
    Apple (Z, P)
    return

def Main():
    Pear (2, 7, 3)
    Apple ([2, 7], [3])
    Pear ("Dog","Cat","House")
    return
```

<table>
<thead>
<tr>
<th>Answers</th>
<th>Explanation of Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><code>Pear(2,7,3) ➔ Apple(2,3) ➔ print 2+3</code></td>
</tr>
<tr>
<td>9</td>
<td><code>Pear(2,7,3) ➔ Apple(7,2) ➔ print 7+2</code></td>
</tr>
<tr>
<td><code>[2,7,3]</code></td>
<td><code>Apple([2,7],[3]) ➔ print [2,7]+[3]</code></td>
</tr>
<tr>
<td><code>DogHouse</code></td>
<td><code>Pear(&quot;Dog&quot;,&quot;Cat&quot;,&quot;House&quot;) ➔ Apple(&quot;Dog&quot;,&quot;House&quot;) ➔ print &quot;Dog&quot;+&quot;House&quot;</code></td>
</tr>
<tr>
<td><code>CatDog</code></td>
<td><code>Pear(&quot;Dog&quot;,&quot;Cat&quot;,&quot;House&quot;) ➔ Apple(&quot;Cat&quot;,&quot;Dog&quot;) ➔ print &quot;Cat&quot;+&quot;Dog&quot;</code></td>
</tr>
</tbody>
</table>

Scoring: 3 points per line (total of 15 points). Remove the 3 points if any line is missing.

For the remaining 5 points, assess as follows:
- 1 for any mistakes in calculating 5 or 9
- 1 for any mistakes in `[2, 7, 3]`
- 1 for any mistakes in `DogHouse`, including capitalization
- 1 for any mistakes in `CatDog`, including capitalization
- 1 for any extant lines out of order

Notice that the function `Apple` doesn’t really care about the type of its arguments, as long as addition has some meaning for those types. This is why addition returns a result for integers, lists, and strings.
20 Points – The grid to the right shows the current red, green, and blue values for a pixel and its eight immediate neighbors. How will the new red, green, and blue values for the center pixel be computed when the Focus filter (shown below) is applied?

You do NOT have to do the arithmetic, but you must write formulae so that you could do the arithmetic and get the right answers if you had a calculator available.

\[
\text{Red} = \frac{((-1 \times 123) + (-1 \times 128) + (7 \times 129) + (-1 \times 120) + (-1 \times 132))}{3} + 0
\]

\[
\text{Green} = \frac{((-1 \times 27) + (-1 \times 30) + (7 \times 40) + (-1 \times 30) + (-1 \times 32))}{3} + 0
\]

\[
\text{Blue} = \frac{((-1 \times 209) + (-1 \times 212) + (7 \times 210) + (-1 \times 212) + (-1 \times 215))}{3} + 0
\]

Scoring: 6 points per answer. Remove 1 point for each missing or incorrect portion up to 6 points, including missing multiplication terms or missing the final division. DO NOT penalize for omitting the + 0 at the end, and DO NOT penalize for omitting the multiplication by zero terms. Accept any answers that are mathematically correct, even if they cannot be entered as-is into Python (including using square brackets as parentheses, use of ÷ for division, · for multiplication, use of vulgar fractions, etc.).
<5>  20 Points – A **Splot** is a gray circle of radius 190, with each **Widget** (of four) centered 70 pixels diagonally away from the center. A **Widget** is a red ellipse of $x$-radius 60 and $y$-radius 20, with a cyan ellipse of $x$-radius 20 and $y$-radius 60 on top, and each **Blot** (of four) centered 40 pixels away on each corner, as shown. A **Blot** is a green circle of radius 20, with a yellow circle of radius 10 on top. Fill in the blanks in the functions below to complete the drawing of a **Splot** at location $<X,Y>$ (which is the center of the **Splot**, shown with a dot). You must use the `addEllipse` function to draw both ellipses and circles.

```python
def addEllipse (Canvas, Xc, Yc, Xr, Yr, NewColor):
    addOvalFilled(Canvas,Xc-Xr,Yc-Yr,2*Xr+1,2*Yr+1,NewColor)
    addOval(Canvas,Xc-Xr,Yc-Yr,2*Xr+1,2*Yr+1,black)
    return

def addSplot (Canvas, X, Y):
    def addBlot (Canvas, X, Y):
        addEllipse (Canvas, X, Y, 20, 20, green)
        addEllipse (Canvas, X, Y, 10, 10, yellow)
        return

    def addWidget (Canvas, X, Y):
        addEllipse (Canvas, X, Y, 60, 20, red)
        addEllipse (Canvas, X, Y, 20, 60, cyan)
        addBlot (Canvas, X-40, Y-40)
        addBlot (Canvas, X-40, Y+40)
        addBlot (Canvas, X+40, Y-40)
        addBlot (Canvas, X+40, Y+40)
        return

    addEllipse (Canvas, X, Y, 190, 190, gray)
    addWidget (Canvas, X-70, Y-70)
    addWidget (Canvas, X-70, Y+70)
    addWidget (Canvas, X+70, Y-70)
    addWidget (Canvas, X+70, Y+70)
    return
```

Scoring: I count 41 slots to fill in. Remove ½ point per slot, but do not go below zero.