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 worth 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:

\[
\begin{align*}
\text{Frog} &= 15 \\
\text{Toad} &= 3.0 \\
\text{Goat} &= "\text{COMPUTER SCIENCE ROCKS}" \\
\text{Newt} &= [2, 7.5, 3L, "\text{Frog}", [2,5,8], \text{True}] \\
\text{Bird} &= (7, 3, 2, "\text{Toad}", 6.0)
\end{align*}
\]

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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Frog + 1</td>
<td>16 (int)</td>
</tr>
<tr>
<td>2. Toad + 1</td>
<td>4.0 (float)</td>
</tr>
<tr>
<td>3. Frog / 2</td>
<td>7 (int)</td>
</tr>
<tr>
<td>4. Toad / 2</td>
<td>1.5 (float)</td>
</tr>
<tr>
<td>5. 2 * Frog + 1</td>
<td>31 (int)</td>
</tr>
<tr>
<td>6. len(Goat)</td>
<td>22 (int)</td>
</tr>
<tr>
<td>7. len(Frog)</td>
<td>ERROR</td>
</tr>
<tr>
<td>8. len(Newt)</td>
<td>6 (int)</td>
</tr>
<tr>
<td>9. len(Newt[3])</td>
<td>4 (int)</td>
</tr>
<tr>
<td>11. Newt[4] + [Toad]</td>
<td>[2,5,8,3.0] (list)</td>
</tr>
<tr>
<td>14. Bird + Newt</td>
<td>ERROR</td>
</tr>
<tr>
<td>15. 5 + Newt[-1]</td>
<td>6 (int)</td>
</tr>
<tr>
<td>16. 5 + (Toad &lt; 1.0)</td>
<td>5 (int)</td>
</tr>
<tr>
<td>17. range(Bird[0])</td>
<td>[0,1,2,3,4,5,6] (list)</td>
</tr>
<tr>
<td>18. range(1,Frog,4)</td>
<td>[1,5,9,13] (list)</td>
</tr>
<tr>
<td>19. range(Newt[0],-1,-1)</td>
<td>[2,1,0] (list)</td>
</tr>
<tr>
<td>20. range(Frog,len(Goat))</td>
<td>[15,16,17,18,19,20,21]</td>
</tr>
<tr>
<td>21. [0 for I in [2,8,1]]</td>
<td>[0,0,0] (list)</td>
</tr>
<tr>
<td>22. [I for I in [2,8,1]]</td>
<td>[2,8,1] (list)</td>
</tr>
<tr>
<td>23. [I for I in Newt[4]]</td>
<td>[2,5,8] (list)</td>
</tr>
<tr>
<td>24. [I*I for I in range(5)]</td>
<td>[0,1,4,9,16] (list)</td>
</tr>
<tr>
<td>25. [2*I+1 for I in range(3)]</td>
<td>[1,3,5] (list)</td>
</tr>
</tbody>
</table>
10 points – Show what is printed by the following code fragment for each given case: (2 points each question, all or nothing)

<table>
<thead>
<tr>
<th>Case #1: N=10</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case #2: N=5</td>
<td>A</td>
</tr>
<tr>
<td>Case #3: N=20</td>
<td>C</td>
</tr>
<tr>
<td>Case #4: N=40</td>
<td>D</td>
</tr>
<tr>
<td>Case #5: N=15</td>
<td>C</td>
</tr>
</tbody>
</table>

<3> 15 Points – I have a list L containing a bunch of strings (such as ["Frog","Toad","Goat",...,"Bird","Fred"]). You don’t know how many items are in the list. Write a while-loop, using I as the loop control variable, that steps through and prints the items in the list in reverse order. Some framework code is provided for you.

**EXPECTED:**

\[
I = \text{len}(L) - 1 \\
\text{while } (I >= 0): \\
    \text{print } L[I] \\
    I = I - 1
\]

**ACCEPTABLE:**

\[
I = 0 \\
\text{while } (I < \text{len}(L)): \\
    \text{print } L[\text{len}(L)-I-1] \\
    I = I + 1
\]

Grading: For each line, remove points according to severity of error. For example, \((I > 0)\) instead of \((I >= 0)\) in the expected answer is -1 point.
20 Points – Show what is printed out as the result from calling `Main()` (four lines total):
(5 points each answer, all or nothing)

```python
def Frog(M, J, Q):
    R = J - Q
    return M + R

def Toad(Z, Q, J):
    return Frog(Q, Z, J)

def Newt(R, Z, Q):
    return Toad(Q, Z+2, R)

def Main():
    print Frog(5, 2, 6)  # Answer #1: 1
    print Toad(1, 6, 3)  # Answer #2: 4
    print Newt(9, 3, 4)  # Answer #3: 0
    print Frog(2, 6, 2)  # Answer #4: 6
    return
```

Answer #1: 1
Answer #2: 4
Answer #3: 0
Answer #4: 6
15 Points – The code below contains syntax errors. Locate each one and indicate what the correction(s) should be. Don’t rewrite any code statements; just correct the mistakes.

```python
def Frog (P,Q):
    print P + Q
    Return

def Main() :
    Z = input("Enter a number --- "):
    for I in range(10):
        for J in [2,8,3,5]:
            if (I+J < Z):
                Frog(I,J)
    return
```

I count 8 distinct errors. Assign +2 for each error found, -1 for each correct item misidentified as an error, but do not go above 15 nor below 0.
20 Points – The following code loads in a graphic picture from a file. Finish the function by doing the same process to each pixel \( \text{PX} \), as follows: Set red to 255 if red was originally greater than 128 but set red to 0 if not; set green to 255 if green was originally greater than 128 but set green to 0 if not; set blue to 255 if blue was originally greater than 128 but set blue to 0 if not. **5 POINT BONUS QUESTION**: what is the maximum number of distinct colors that the resulting image could contain?

```python
def Main():
    Filename = pickAFile()
    Canvas = makePicture(Filename)
    show(Canvas)
    for Y in range(getHeight(Canvas)):
        for X in range(getWidth(Canvas)):
            PX = getPixel(Canvas,X,Y)

            if getRed(PX) > 128:
                setRed(PX,255)
            else:
                setRed(PX,0)

            if getGreen(PX) > 128:
                setGreen(PX,255)
            else:
                setGreen(PX,0)

            if getBlue(PX) > 128:
                setBlue(PX,255)
            else:
                setBlue(PX,0)

    repaint(Canvas)
    return
```

Assign 5 points to each section, and the remaining 5 points for overall syntax issues. In each section, give full credit if the overall structure is basically correct: an if–else that tests a primary color and sets it appropriately. There are acceptable alternatives, such as putting the single statement of each body on the same line as the if or the else:

```python
    if getRed(PX) > 128: setRed(PX,255)
    else: setRed(PX,0)
```

as well as purely computational solutions such as:

```python
    setRed(PX, getRed(PX) / 128 * 255)
```

**BONUS QUESTION +5 POINTS**: 8 distinct colors