This quiz has 3 questions, for a total of 30 points.

1. 9 points  Suppose that \( L \) is a Python list (array) of length \( n \). Categorize the worst-case execution time of the below expressions as either

   1. constant time (takes the same amount of time no matter what \( n \) is).
   2. logarithmic time (takes time proportional to \( \lg n \)).
   3. linear time (takes time proportional to \( n \))
   4. quadratic (takes time proportional to \( n^2 \))

Label each operation with the above item number.

- \( L[n-1] \)
- \( 0 \text{ in } L \)
- \( L + L \)

Solution:

- (1), \( L[n-1] \) is constant time, (3 points)
- (3), \( 0 \text{ in } L \) is linear time, (3 points)
- (3), \( L + L \) is linear time, (3 points)

2. 12 points  Complete the following implementation of the \texttt{insert_before} method in the following class that implements a doubly-linked list.

```python
class DLNode:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def insert_before(self, node, data):
        new_node = DLNode(a)
        if self.head == node:
            self.head = new_node
        ___(b)___ = node.prev
        new_node.next = ___(c)___
        node.prev = ___(d)___
```

2. 12 points  Complete the following implementation of the \texttt{insert_before} method in the following class that implements a doubly-linked list.

```python
class DLNode:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def insert_before(self, node, data):
        new_node = DLNode(a)
        if self.head == node:
            self.head = new_node
        ___(b)___ = node.prev
        new_node.next = ___(c)___
        node.prev = ___(d)___
```
Solution:

(a) data (3 point)
(b) new_node.prev (3 point)
(c) node (3 point)
(d) new_node (3 point)

3. 9 points Let \( f(n) = 2n + 2 \) and \( g(n) = n \). Give the definition of Big-O and prove that \( f(n) \in O(g(n)) \). (Use the other side of this paper for the proof.)

Solution:

\[
O(g(n)) = \{ f(n) \mid \exists n_0. \forall n \geq n_0. \exists c. 0 \leq f(n) \leq c g(n) \}
\]

We need to choose a \( c \) such that \( cn \) becomes greater than \( 2n + 2 \) at some point. We choose \( c = 3 \). Next we need to find out at what point \( 3n \) is equal to or bigger than \( 2n + 2 \), so we chart those out:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( 2n + 2 )</th>
<th>( 3n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

So it looks like \( n_0 = 2 \) is a good choice. We are now ready to give the proof.

To show that \( 2n + 2 \in O(n) \), we need to show that

\[
\exists n_0. \forall n \geq n_0. \exists c. 0 \leq 2n + 2 \leq c n
\]

We choose \( n_0 = 2 \) and \( c = 3 \). So we need to prove that

\[
\forall n \geq 2. 0 \leq 2n + 2 \leq 3 n
\]

We subtract \( 2n \) from both sides to get the following

\[
\forall n \geq 2. 0 \leq 2 \leq n
\]

which is clearly true.