# CSC148 - Choosing Test Cases

```
def insert_after(lst: List[int], n1: int, n2: int) -> None:
    """After each occurrence of <n1> in <lst>, insert <n2>.

    >>> lst = [5, 1, 2, 1, 6]
    >>> insert_after(lst, 1, 99)
    >>> lst
    [5, 1, 99, 2, 1, 99, 6]
    """
```

1. We can only test a tiny fraction of all possible calls to this function. Some properties are likely not relevant, such as whether the values in `lst` are positive or negative. In the table below, we have named one property that *is* relevant. Add more.

| **Relevant Property** | **Values to Try** |
|---|---|
| position of `n1` in `lst` | front, back, somewhere else |
| | |
| | |
| | |

2. Use the above table to help you define specific test cases. Again, we have started. Add more. Continue on the reverse if needed.

| **lst** | **n1** | **n2** | **Purpose** |
|---|---|---|---|
| `[0, 1, 2, 3]` | `0` | `99` | `n1` at the front |
| `[0, 1, 2, 3]` | `3` | `99` | `n1` at the back |
| `[0, 1, 2, 3]` | `1` | `99` | `n1` somewhere else |
| | | | |
| | | | |
| | | | |
| | | | |

Are you combining the properties? You may need to use judgment to choose among the many combinations, or you could end up with *many* test cases.

| lst | n1 | n2 | Purpose |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

3. Here is an example showing how `pytest` can be used to implement the first test case shown in Question 2:

```
def test_insert_after_at_front() -> None:
    """Test insert_after with one occurrence of n1 at the front of lst.
    """
    input_list = [0, 1, 2, 3]
    insert_after(input_list, 0, 99)
    expected = [0, 99, 1, 2, 3]
    assert input_list == expected
```

Choose one of your own test cases from Question 2 and implement it in `pytest`.