CP2410 Practical 05

Sihan Chen, jcu ID: 14187662

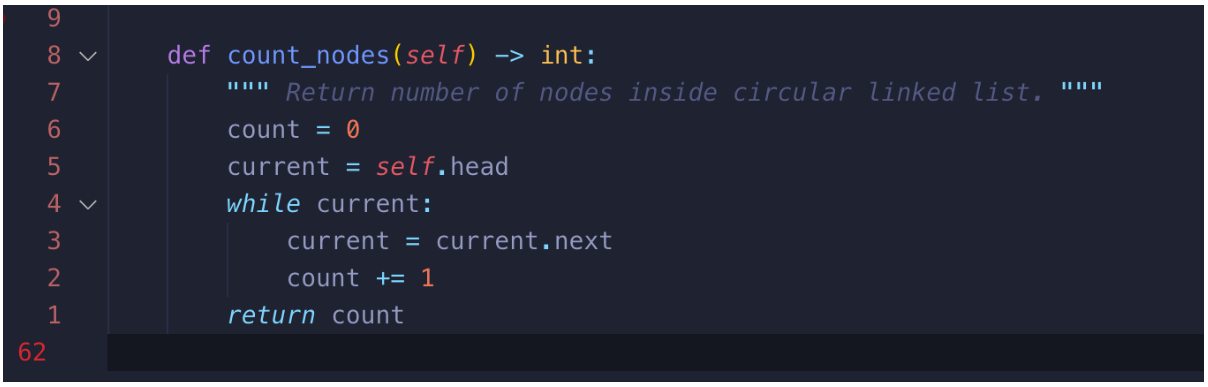
Question 1

```python
def find_second_to_last(head):
    cur, prev = head, head
    while cur._next:
        prev = cur
        cur = cur._next
    return prev
```

Assuming we are given the head node of the singly linked list. We will use `prev` and `cur`, two pointers to iterate through singly linked list. 'prev' will be the node before 'cur', and when the while loop finishes, 'cur' will point at the last node and 'prev' will be the second to last node.

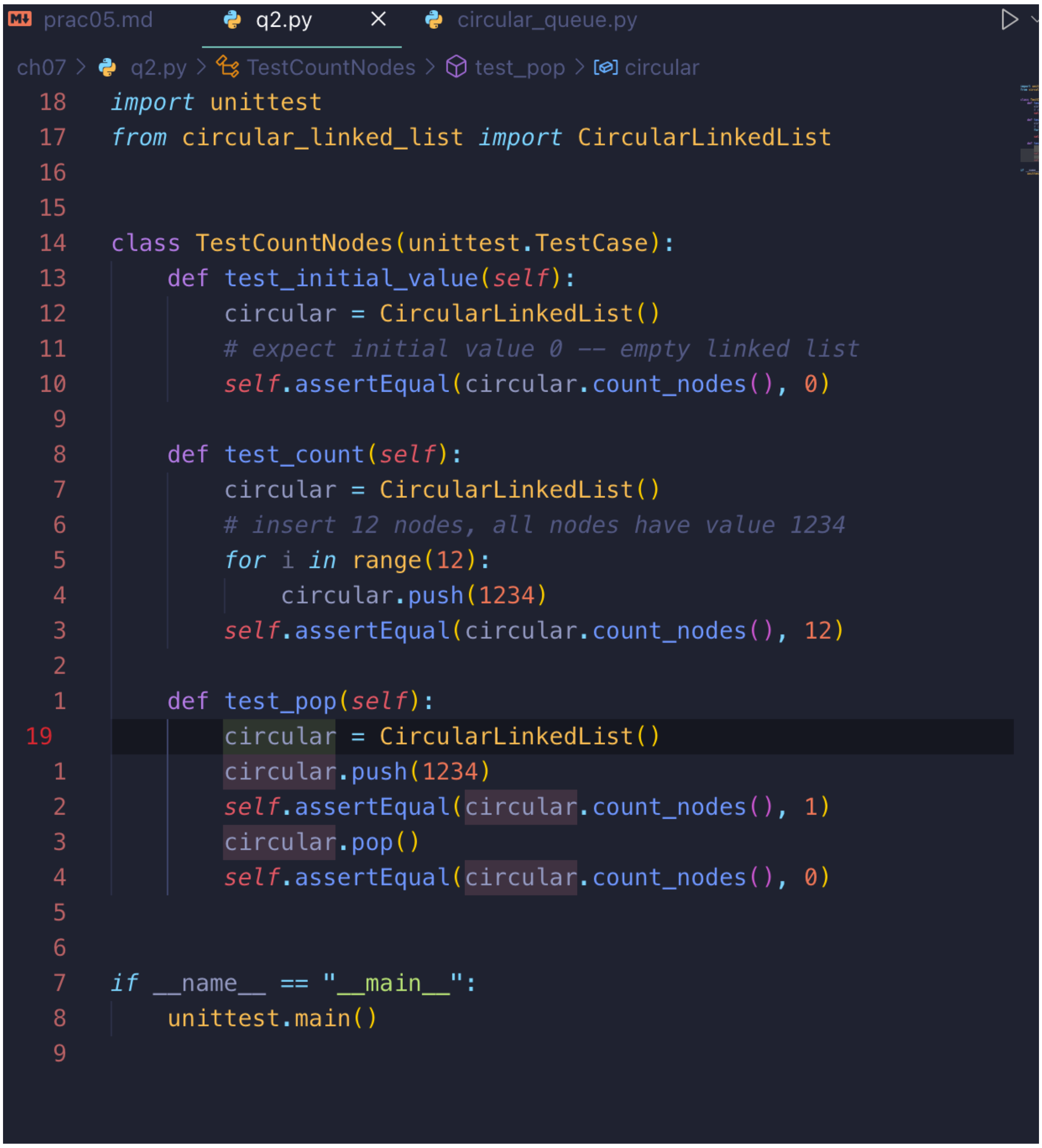
Question 2

To count number of nodes in a circular linked list, here is the code implementation:



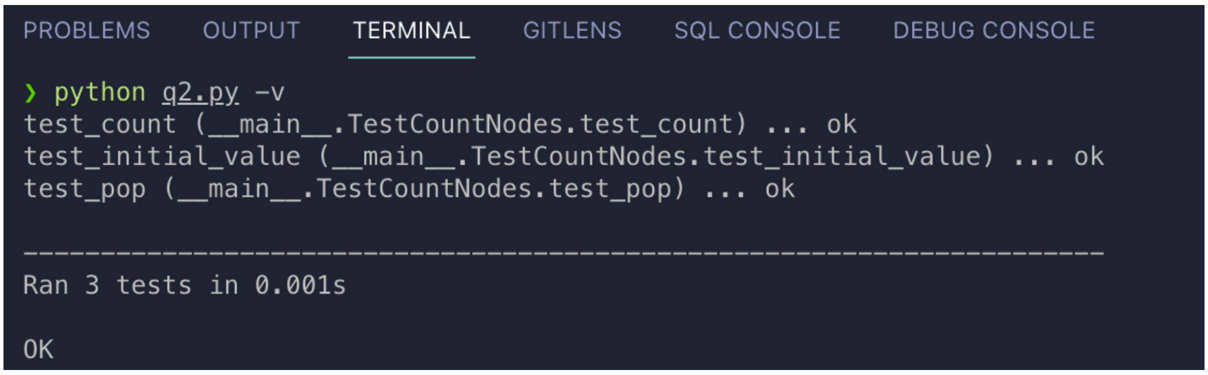
```python
    def count_nodes(self) -> int:
        """ Return number of nodes inside circular linked list. """
        count = 0
        current = self.head
        while current:
            current = current.next
            count += 1
        return count
```

Here is the unit test:



```python
import unittest
from circular_linked_list import CircularLinkedList


class TestCountNodes(unittest.TestCase):
    def test_initial_value(self):
        circular = CircularLinkedList()
        # expect initial value 0 -- empty linked list
        self.assertEqual(circular.count_nodes(), 0)

    def test_count(self):
        circular = CircularLinkedList()
        # insert 12 nodes, all nodes have value 1234
        for i in range(12):
            circular.push(1234)
        self.assertEqual(circular.count_nodes(), 12)

    def test_pop(self):
        circular = CircularLinkedList()
        circular.push(1234)
        self.assertEqual(circular.count_nodes(), 1)
        circular.pop()
        self.assertEqual(circular.count_nodes(), 0)


if __name__ == "__main__":
    unittest.main()
```

Running result of unit test:



```
❯ python q2.py -v
test_count (__main__.TestCountNodes.test_count) ... ok
test_initial_value (__main__.TestCountNodes.test_initial_value) ... ok
test_pop (__main__.TestCountNodes.test_pop) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

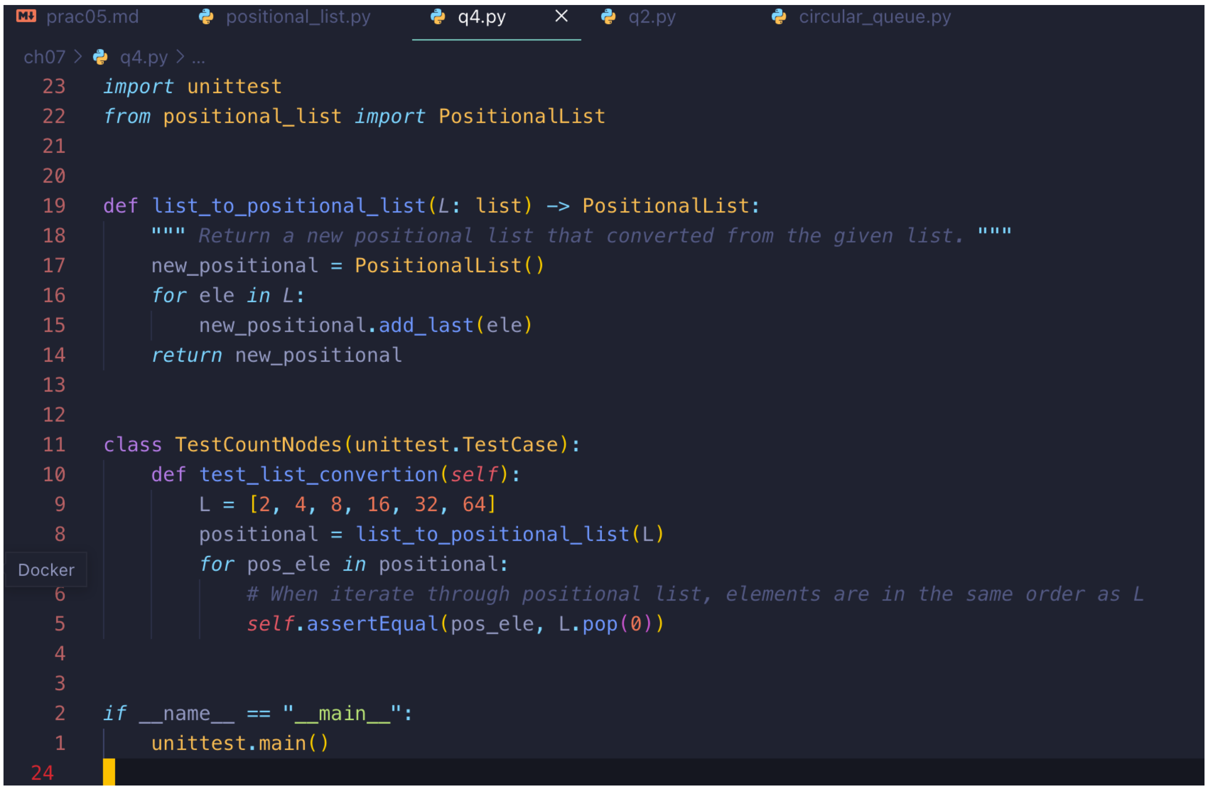
Question 3

```
def checkSameCircular(x, y) -> bool:
    """ Return True if x and y belongs to the same circular linked list.
"""
    current = x
    while current.next != x:
        if current == y:
            return True
        current = current.next
    return False
```

We start from x, and loop through the entire circular linked list by calling x.next(). During iteration, if we found y, means x and y are in the same list, return True. Else, since x is a node in circular linked list, if we come back to x, and never find y, means x and y are not in the same list, return False.

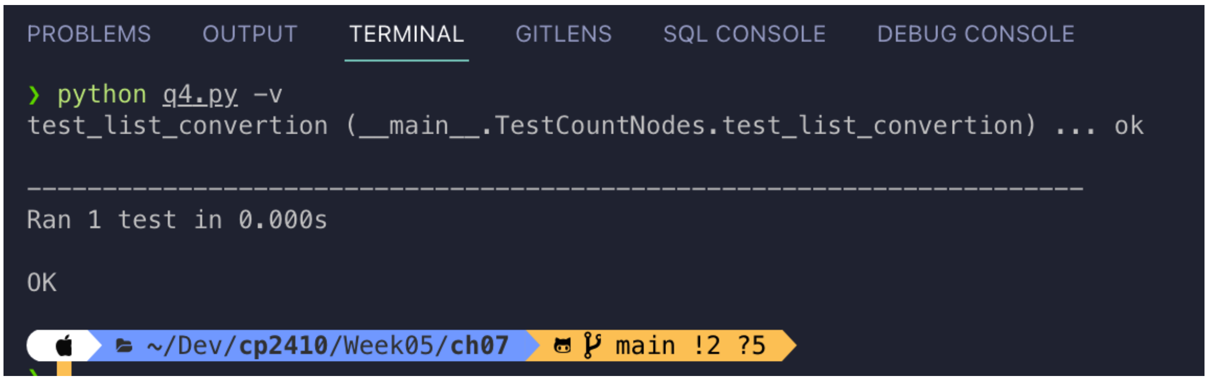
Question 4

Here is the implementated code for function `list_to_positional_list(L)` and its unittest:



```
import unittest
from positional_list import PositionalList


def list_to_positional_list(L: list) -> PositionalList:
    """ Return a new positional list that converted from the given list. """
    new_positional = PositionalList()
    for ele in L:
        new_positional.add_last(ele)
    return new_positional


class TestCountNodes(unittest.TestCase):
    def test_list_convertion(self):
        L = [2, 4, 8, 16, 32, 64]
        positional = list_to_positional_list(L)
        for pos_ele in positional:
            # When iterate through positional list, elements are in the same order as L
            self.assertEqual(pos_ele, L.pop(0))


if __name__ == "__main__":
    unittest.main()
```

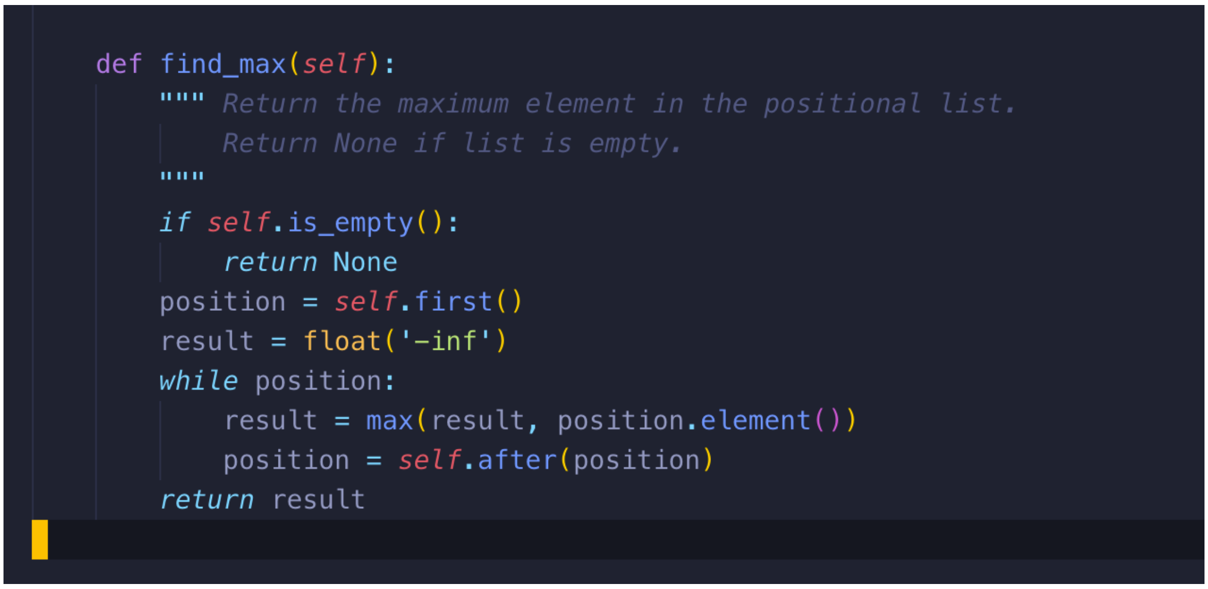
Running result of unit test:



```
❯ python q4.py -v
test_list_convertion (__main__.TestCountNodes.test_list_convertion) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

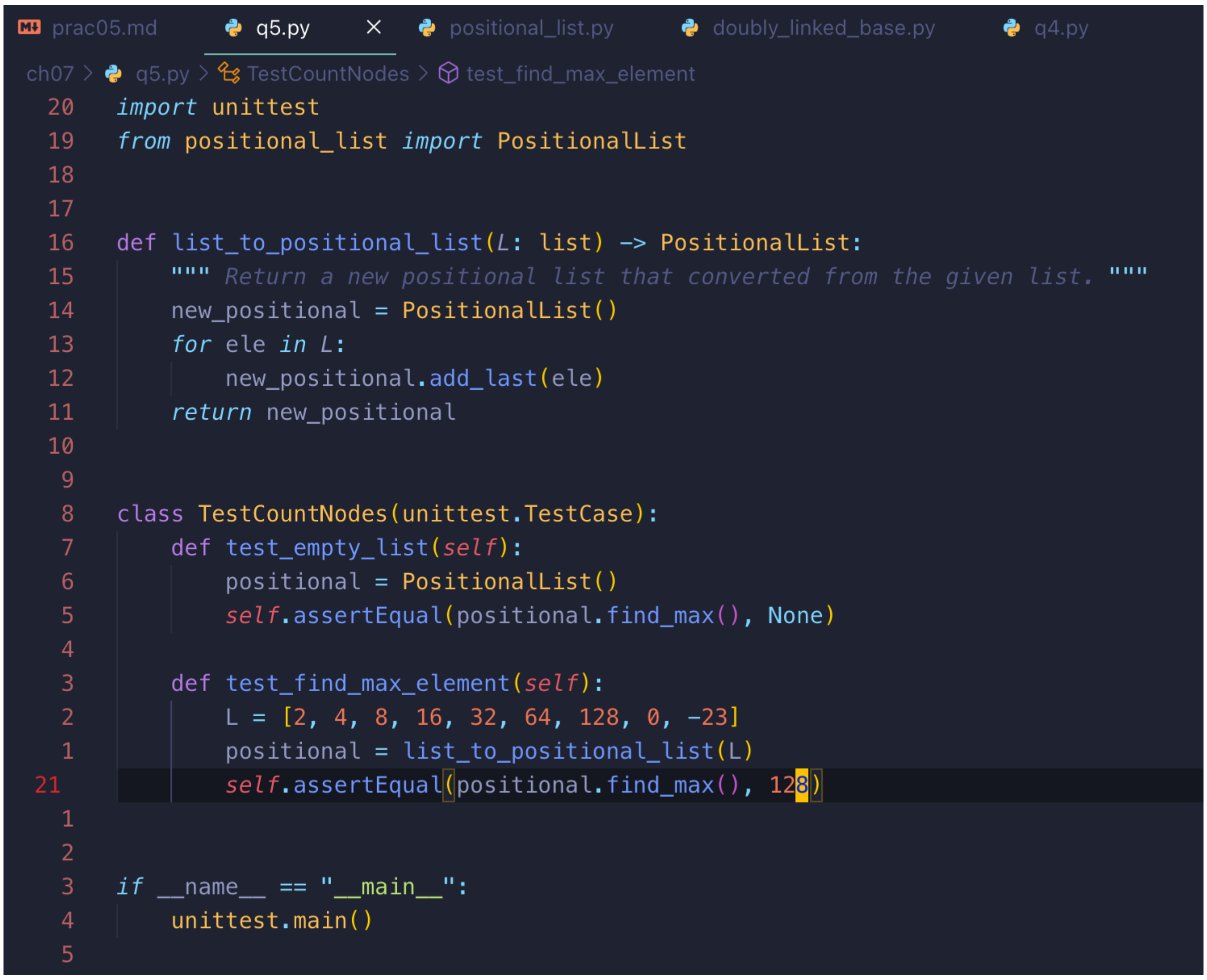
Question 5

To find the maximum element in a positional list, here is the code implementation:



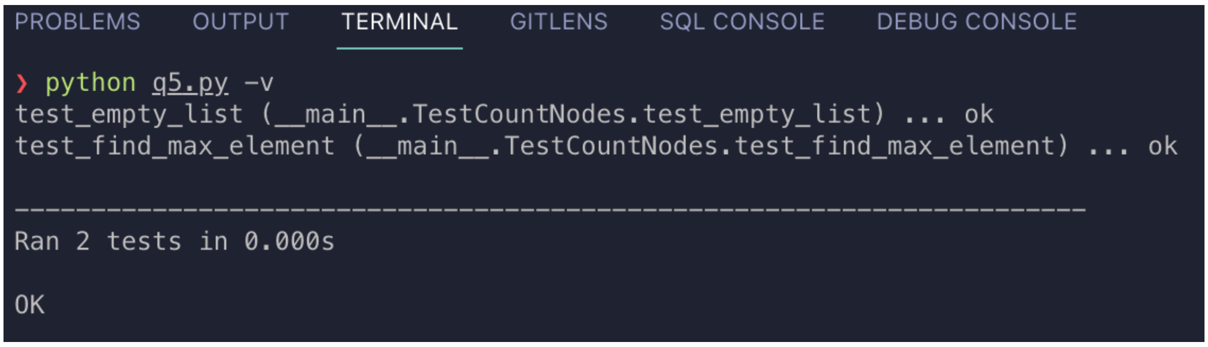
```python
    def find_max(self):
        """ Return the maximum element in the positional list.
            Return None if list is empty.
        """
        if self.is_empty():
            return None
        position = self.first()
        result = float('-inf')
        while position:
            result = max(result, position.element())
            position = self.after(position)
        return result
```

Here is the unit test:



```python
import unittest
from positional_list import PositionalList


def list_to_positional_list(L: list) -> PositionalList:
    """ Return a new positional list that converted from the given list. """
    new_positional = PositionalList()
    for ele in L:
        new_positional.add_last(ele)
    return new_positional


class TestCountNodes(unittest.TestCase):
    def test_empty_list(self):
        positional = PositionalList()
        self.assertEqual(positional.find_max(), None)

    def test_find_max_element(self):
        L = [2, 4, 8, 16, 32, 64, 128, 0, -23]
        positional = list_to_positional_list(L)
        self.assertEqual(positional.find_max(), 128)


if __name__ == "__main__":
    unittest.main()
```

Running result of unit test:



```
❯ python q5.py -v
test_empty_list (__main__.TestCountNodes.test_empty_list) ... ok
test_find_max_element (__main__.TestCountNodes.test_find_max_element) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```