

CP2410 Practical 05

Sihan Chen, jcu ID: 14187662

Question 1

```
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:

```
9  
8  ✓ def count_nodes(self) -> int:  
7      """ Return number of nodes inside circular linked list. """  
6      count = 0  
5      current = self.head  
4  ✓ while current:  
3          current = current.next  
2          count += 1  
1      return count  
62
```

Here is the unit test:

```

prac05.md  q2.py  x  circular_queue.py
ch07 > q2.py > TestCountNodes > test_pop > circular
18  import unittest
17  from circular_linked_list import CircularLinkedList
16
15
14  class TestCountNodes(unittest.TestCase):
13      def test_initial_value(self):
12          circular = CircularLinkedList()
11          # expect initial value 0 -- empty linked list
10          self.assertEqual(circular.count_nodes(), 0)
9
8      def test_count(self):
7          circular = CircularLinkedList()
6          # insert 12 nodes, all nodes have value 1234
5          for i in range(12):
4              circular.push(1234)
3              self.assertEqual(circular.count_nodes(), 12)
2
1      def test_pop(self):
19          circular = CircularLinkedList()
1              circular.push(1234)
2              self.assertEqual(circular.count_nodes(), 1)
3              circular.pop()
4              self.assertEqual(circular.count_nodes(), 0)
5
6
7  if __name__ == "__main__":
8      unittest.main()
9

```

Running result of unit test:

```

PROBLEMS  OUTPUT  TERMINAL  GITLENS  SQL CONSOLE  DEBUG CONSOLE
> 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

```

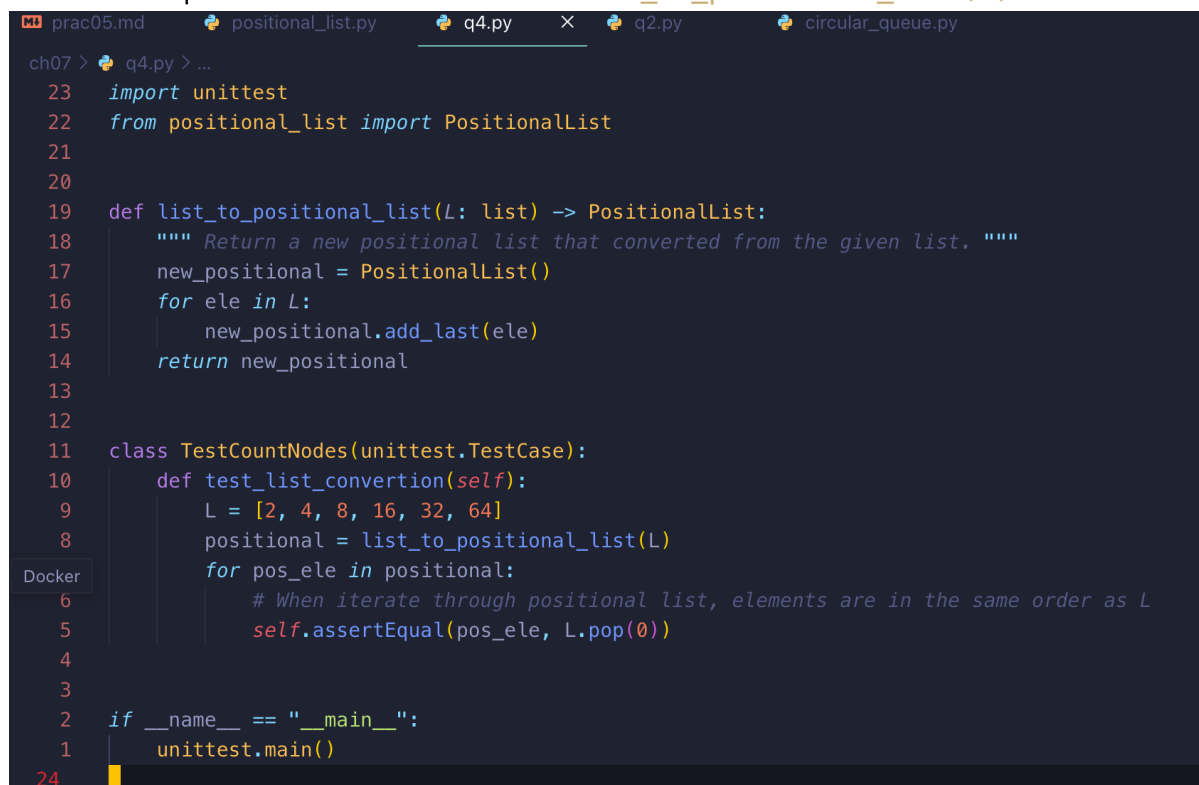
Question 3

```
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.

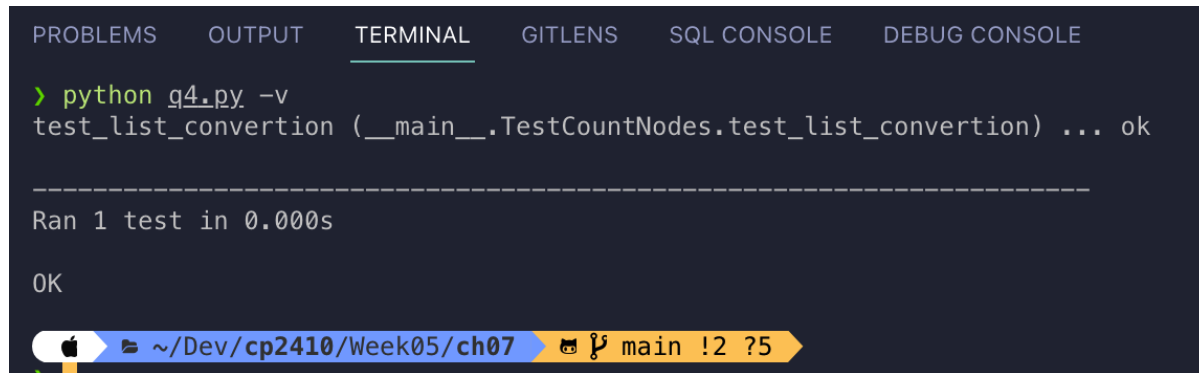
Question 4

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



```
prac05.md positional_list.py q4.py x q2.py circular_queue.py
ch07 > q4.py > ...
23 import unittest
22 from positional_list import PositionalList
21
20
19 def list_to_positional_list(L: list) -> PositionalList:
18     """ Return a new positional list that converted from the given list. """
17     new_positional = PositionalList()
16     for ele in L:
15         new_positional.add_last(ele)
14     return new_positional
13
12
11 class TestCountNodes(unittest.TestCase):
10     def test_list_conversion(self):
9         L = [2, 4, 8, 16, 32, 64]
8         positional = list_to_positional_list(L)
7         for pos_ele in positional:
6             # When iterate through positional list, elements are in the same order as L
5             self.assertEqual(pos_ele, L.pop(0))
4
3
2 if __name__ == "__main__":
1     unittest.main()
24
```

Running result of unit test:



The screenshot shows a terminal window with a dark background. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL' (which is active), 'GITLENS', 'SQL CONSOLE', and 'DEBUG CONSOLE'. The terminal output shows a command to run a Python script with verbose output, followed by the test result 'ok'. A separator line is shown, then the execution time 'Ran 1 test in 0.000s', and finally 'OK'. The bottom status bar shows the file path '~/Dev/cp2410/Week05/ch07', the branch 'main', and the line/character count '!2 ?5'.

```
PROBLEMS  OUTPUT  TERMINAL  GITLENS  SQL CONSOLE  DEBUG CONSOLE

> python g4.py -v
test_list_conversion (__main__.TestCountNodes.test_list_conversion) ... ok

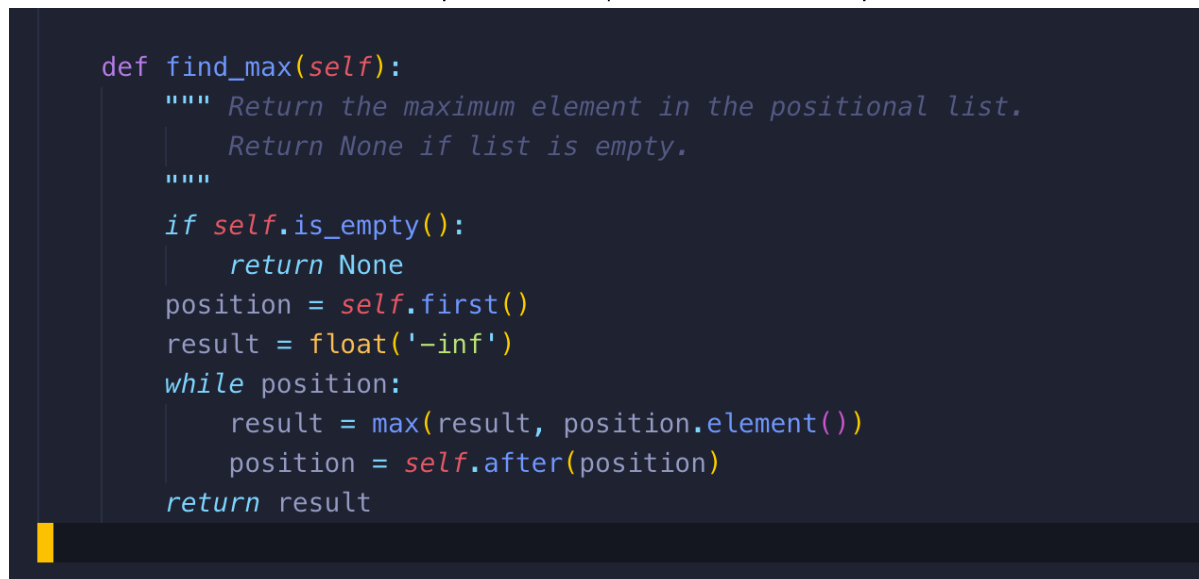
-----
Ran 1 test in 0.000s

OK

~/Dev/cp2410/Week05/ch07  main !2 ?5
```

Question 5

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



The screenshot shows a code editor with a dark background. It contains a Python method definition for 'find_max' on a 'self' object. The method has a docstring that says 'Return the maximum element in the positional list. Return None if list is empty.' The code checks if the list is empty, and if not, it iterates through the list to find the maximum element by comparing each element to the current maximum. The method returns the maximum value.

```
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:

```
prc05.md  q5.py  ×  positional_list.py  doubly_linked_base.py  q4.py
ch07 > q5.py > TestCountNodes > test_find_max_element
20 import unittest
19 from positional_list import PositionalList
18
17
16 def list_to_positional_list(L: list) -> PositionalList:
15     """ Return a new positional list that converted from the given list. """
14     new_positional = PositionalList()
13     for ele in L:
12         new_positional.add_last(ele)
11     return new_positional
10
9
8 class TestCountNodes(unittest.TestCase):
7     def test_empty_list(self):
6         positional = PositionalList()
5         self.assertEqual(positional.find_max(), None)
4
3     def test_find_max_element(self):
2         L = [2, 4, 8, 16, 32, 64, 128, 0, -23]
1         positional = list_to_positional_list(L)
21         self.assertEqual(positional.find_max(), 128)
1
2
3 if __name__ == "__main__":
4     unittest.main()
5
```

Running result of unit test:

```
PROBLEMS  OUTPUT  TERMINAL  GITLENS  SQL CONSOLE  DEBUG CONSOLE
> 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
```