Report 4

Question 1:

1. This question is saved in q1.py.
2. this code can create a class named SinglyLinkedList. In this class, you can use insert() function to insert elements. Besides, you can use recursively_count() function to count how many elements are there in the singly linked list. The base case is when n (the current node) is pointing to the end. For each iteration, it will count once. (a local variable count will +1)
3. the user can input (in the code area) the node they want to start with. It will automatically count how many elements between the chosen node and the tail.
4. sample run: (the above is the code representing the node. The below is the output)

```
28  SL = SinglyLinkedList()
29  node1=SL.insert(1)
30  node2=SL.insert(2)
31  node3=SL.insert(7)
32  node4=SL.insert(8)
33  node5=SL.insert(9)
34  print('The number of nodes is: ',SL.recursive_count(node5

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Microsoft Windows [版本 10.0.19041.985]
(c) Microsoft Corporation。保留所有权利。

C:\Users\surface\Desktop\CSC1001A4>c:/users/surface/appdata/loc
The number of nodes is:  5
```

Question 2:

1. this code is saved in q2.py
2. this code can will create a class named singly linked list. In this list, you may use insert() method to insert elements. Besides, you may use method quick_sort() to quick sort the singly linked list.

There are some supplimentary methods in this question:

Output(): to print out the linked list

Delete(): to delete a node

Add_tail(): to add a node in the tail

Delete_head(): to delete the head of the singly linked list

Concat(): to connect two part of singly linked list together

3. sample run:

```
93
94   SL = SinglyLinkedList()
95   node1=SL.insert(3)
96   node2=SL.insert(14)
97   node3=SL.insert(5)
98   node4=SL.insert(2)
99   node5=SL.insert(4)
100  node6=SL.insert(7)
101  node7=SL.insert(6)
102  Sll=SL.quick_sort(SL.head)
103  print('the first node\'s reference: ',Sll.head)
104
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Microsoft Windows [版本 10.0.19041.985]
(c) Microsoft Corporation。保留所有权利。

C:\Users\surface\Desktop\CSC1001A4>c:/users/surface/appdata/local/programs/python/python38/python.exe c:/
the first node's reference:  <__main__.Node object at 0x0000016F57352A00>
```

Question 3:

1. this code is saved in q3.py
2. this code will print the Hanoi tower with the given number of n. it uses no recursion. Firstly, it will determine whether the number of elements is even or not. if it is even, name the help nod='B' while the goal nod = 'C', otherwise, name the two rods reversely. this is to transform the "odd" into the "even" situation. For the smallest plane, if n is even, it will move A-B-C-A……。else, it will move A-C-B-A……. Based on this, the code will start move. In total, there will move 2^n-1 times. The time is corresponding to the number of the plates. For given time, according to its value (%3=0,%3=1,%3=2), there are three different ways to go. After finishing all of the 2^n-1 steps, the code will stop.
3. the code will require the user to input a number n, representing the number of the plates. It must be an integer and positive.
4. sample run:

```
C:\Users\surface\Desktop\CSC1001A4>c:/users/
please enter an integer: 4
A --> B
A --> C
B --> C
A --> B
C --> A
C --> B
A --> B
A --> C
B --> C
B --> A
C --> A
B --> C
A --> B
A --> C
B --> C
```