MH1403 Algorithms and Computing
Lab 2 Linked List
(Week 7, 20.02.2023 – 24.02.2023)

Submission Instructions:

1. This lab is 5% of the final grade of this course.
2. The submission deadline is 11:59PM, 27 February (Monday).
3. You need to submit the codes of Task 1 (task1.py) and Task 2 (task2.py) through NTULearn.

Task 1. (2 marks) Add a method reverse() to the linked list class (the code of linked list class is given in the file linkedlist.py). When we call this method of the linked list class, this method reverses the order of the nodes in the linked list. To have an efficient reverse() method, you are required not to create any new node. To figure out how to reverse the linked list, you can draw on paper to find out when we visit the nodes of the linked list one by one, how to update the .next instance variable of each node, and finally update the .head variable of the linked list to point to the original last node.

After defining the method reverse(), execute the following code:

```
linkedlistA = LinkedList()
for i in range(1,6):
    linkedlistA.addNode(i)

linkedlistA.printNode()
linkedlistA.reverse()
print("After reversing the linked list, the linked list becomes:")
linkedlistA.printNode()
```

Write your code in the file task1.py . You can copy the code of linked list class into the file task1.py, then modify it.

Task 2. (2 marks) In this task, we will use the code of linked list class given in the file linkedlist.py. You are required to write a function copylinkedlist(). When we call this function with an input linked list, it returns a new linked list which is a copy of the input linked list. After copying, modifying the original linked list does not affect the new copy of the linked list and vice versa.

In this task, we assume that the data in each node of the linked list is a numerical number. After defining the function copylinkedlist, execute the following code:

```
linkedlistA = LinkedList()
for i in range(1,6):
    linkedlistA.addNode(i)

linkedlistB = copylinkedlist(linkedlistA)
# modify linkedlistA
curr = linkedlistA.head
for i in range(linkedlistA.size):
    curr.data += 10
    curr = curr.next
linkedlistA.printNode()
linkedlistB.printNode()
```

Write your code in the file task2.py. Note that you should NOT use the Python built-in function copy to copy the linked list in this task.

Remarks. In Python, the assignment operation between two variables does not create a copy of an object. For example:

```
listA = [2,3,4]
listB = listA         # listB and listA are the same list
listA[0] = 1
print(listA)          # [1,3,4]
print(listB)          # [1,3,4]
```

To create a copy of an object, we can use the Python built-in deepcopy function. For example:

```
import copy

listA = [2,3,4]
listB = copy.deepcopy(listA)
```

```
listA[0] = 1
print(listA)          # [1,3,4]
print(listB)          # [2,3,4]
```

To copy a linked list, after creating a linked list linkedlistA, the following code is used to create a copy of a linkedlistA:

```
import copy

linkedlistB = copy.deepcopy(linkedlistA)
```

In Python, there are two ways to create a copy: shallow copy and deep copy. Deep copy is a process in which the copying process occurs recursively. When we use copy.copy(), it is shallow copy; deep copy is copy.deepcopy(). You may refer to the following webpage for more information:

`https://www.geeksforgeeks.org/copy-python-deep-copy-shallow-copy/`

To make it simple, if you want to create a copy of an object so that the copy is completely independent of the original object, you can simply use copy.deepcopy().