Assignment 3.1: Linked List (4 points)

Implement the linked list data structure in Python. Create a class `Node` which stores the data (integer) and link to another Node object in the list. Create an additional class `LinkedList` which maintains the linked list created by Node objects.

You can implement your linked list with or without header and trailer nodes.

Class `LinkedList` must contain following methods:

- `print()`: prints the content of linked list (format: `node1 -> node2 -> node3`).
- `append(data: object)`: inserts `data` to the end of the linked list
- `insert(data: object, i: int)`: inserts `data` to the index i.
- `delete(i: int)`: removes a node at the index i and returns the value (`data`) of the deleted node.

Grading: points are awarded based on the method that are correctly implemented. The print and append methods have to be correctly implemented to obtain any points. The insert and delete operations award 1 point each.

A code template with an example program:

```
class Node:
    # TODO


class LinkedList:
    # TODO


if __name__ == "__main__":
    L = LinkedList()
    L.append(1)
    L.append(3)
    L.print()
    L.insert(10, 1)
    L.insert(15, 0)
    L.print()
    L.delete(0)
    L.print()
```

```
$ python linked.py
1 -> 3
15 -> 1 -> 10 -> 3
1 -> 10 -> 3
```

Submit your solution in CodeGrade as `linked.py`.

Assignment 3.2: Indices and Swapping (2 points)

Implement two new methods to your `LinkedList` class:

- `index(data: object)` (**1 pt**): returns the index where `data` is stored in linked list, returns -1 if not found.
- `swap(i: int, j: int)` (**1 pt**): swaps two nodes in locations i and j, returns without changes if indices are invalid.

A code template with an example program:

```
if __name__ == "__main__":
    L = LinkedList()
    for num in (3, 5, 2, 7, 8, 10, 6):
        L.append(num)
    L.print()
    print(L.index(7))
    print(L.index(9))
    L.swap(1, 4)
    L.print()
    L.swap(2, 0)
    L.print()
```

Output

```
$ python linked.py
3 -> 5 -> 2 -> 7 -> 8 -> 10 -> 6
3
-1
3 -> 8 -> 2 -> 7 -> 5 -> 10 -> 6
2 -> 8 -> 3 -> 7 -> 5 -> 10 -> 6
```

Submit your solution in CodeGrade as `linked.py`.

Assignment 3.3: Insertion Sort (2 points)

Consider week's 1 assignment 1.1: Insertion Sort.

Implement a new method `isort` to you LinkedList class. The method sorts the linked list in ascending order using insertion sort.

A code template with an example program:

```
if __name__ == "__main__":
    L = LinkedList()
    for num in (3, 5, 2, 7, 8, 10, 6):
        L.append(num)
    L.print()
    L.isort()
    L.print()
```

```
$ python linked.py
3 -> 5 -> 2 -> 7 -> 8 -> 10 -> 6
2 -> 3 -> 5 -> 6 -> 7 -> 8 -> 10
```

Submit your solution in CodeGrade as `linked.py`.

Last modified: Sunday, 22 September 2024, 8:56 PM