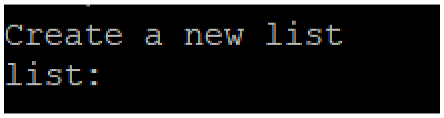
Doubly Linked List

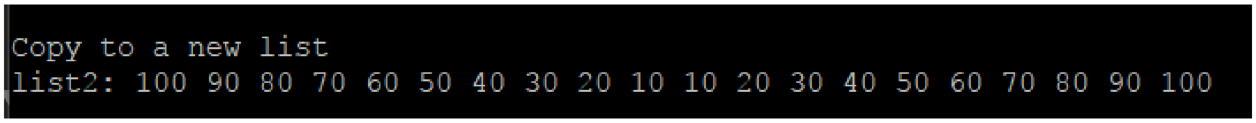
Linked list contains a Node class as private member. This node class helps us get accessed to element before and element after the class. It also has the content(obj) of the item. Linked List has default constructor, copy constructor, copy assignment, and destructor to create and destroy an object. Linked List has the following functions:

1) Default constructor: initialize the linked list. It takes O(1) time.



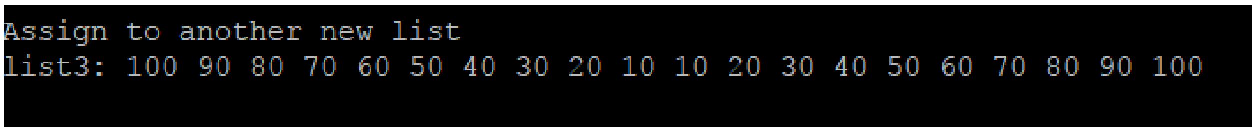
```
Create a new list
list:
```

2) Copy constructor: initialize a list and copy new list into the current list. It takes O(N) time because it iterates through the whole new list.



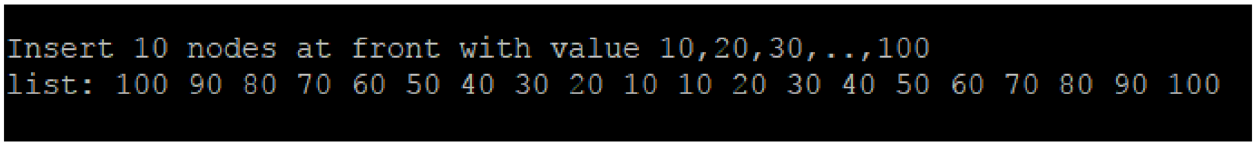
```
Copy to a new list
list2: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
```

3) Copy assignment: copy new list into the current list. It takes O(N) time because it iterates through the whole new list.



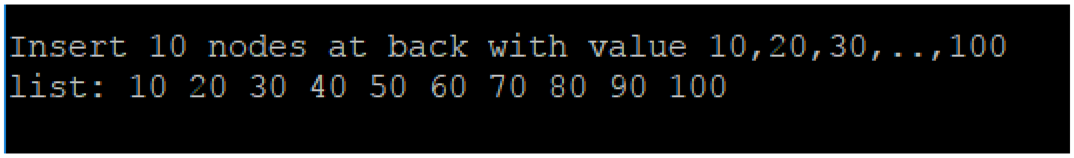
```
Assign to another new list
list3: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
```

4) Destructor: Destroy a list. It takes O(N) time because it deletes one by one.
5) getFirst(): get the first element. It takes O(1) time because it only iterates to the first element.
6) getAfterLast(); get the trailer address of the linked list. It takes O(1) time.
7) isEmpty(): checks if the list is empty. It takes O(1) time.
8) First(): return the object of the first element. It takes O(1) time because it only iterates to the first element.
9) Last(): return the object of the last element. It takes O(1) time because it accesses from the trailer.
10) insertFirst(): insert an element in front of the list. It takes O(1) time because it accesses the header and put the element after header.



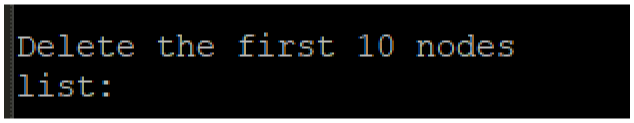
```
Insert 10 nodes at front with value 10,20,30,..,100
list: 100 90 80 70 60 50 40 30 20 10 10 20 30 40 50 60 70 80 90 100
```

11) InsertLast(): insert an element in back of the list. It takes O(1) time because it accesses the trailer and put the element before trailer.



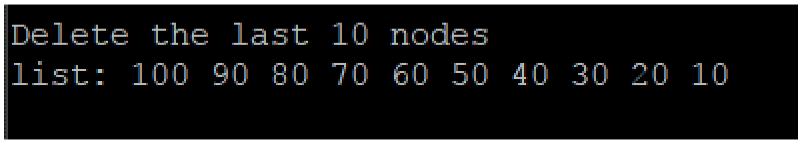
```
Insert 10 nodes at back with value 10,20,30,..,100
list: 10 20 30 40 50 60 70 80 90 100
```

12) removeFirst(): remove an element in front of the list. It takes O(1) time because it accesses the header and remove the element after header.



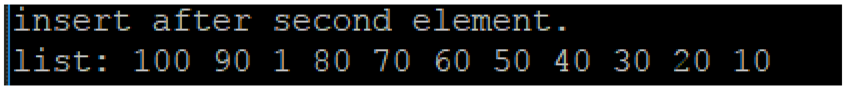
```
Delete the first 10 nodes
list:
```

13) removeLast(): remove an element in back of the list. It takes O(1) time because it accesses the trailer and remove the element before trailer.



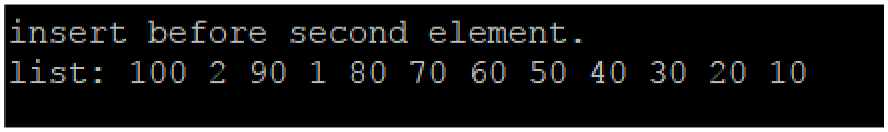
```
Delete the last 10 nodes
list: 100 90 80 70 60 50 40 30 20 10
```

14) insertAfter(): insert an element after node p. it takes O(1) time because we already know the address of node p.



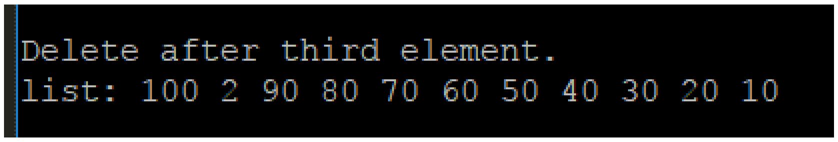
```
insert after second element.
list: 100 90 1 80 70 60 50 40 30 20 10
```

15) insertBefore(): insert an element before node p. it takes O(N) time because we already know address of node p.



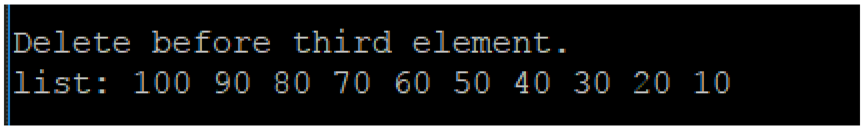
```
insert before second element.
list: 100 2 90 1 80 70 60 50 40 30 20 10
```

16) removeAfter(): remove an element after node p. it takes O(1) time because we already know address of node p.



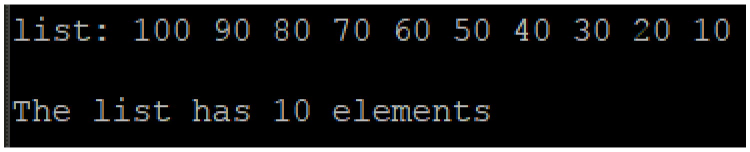
```
Delete after third element.
list: 100 2 90 80 70 60 50 40 30 20 10
```

17) removeBefore(): remove an element before node p. it takes O(1) time because we already know address of node p.



```
Delete before third element.
list: 100 90 80 70 60 50 40 30 20 10
```

18) DoublyLinkedListLength(): find the length of linked list. It takes O(N) time because it has to iterate through all nodes of list to count the total.



```
list: 100 90 80 70 60 50 40 30 20 10

The list has 10 elements
```

19) Operator <<: print out all of the nodes. It takes O(N) time because it has to iterate through all nodes of list to print.