

CODE:

```
from DoublyLinkedListBase import _DoublyLinkedListBase
4 usages: A castro_mh

class PositionList(_DoublyLinkedListBase):
    """A sequential container of elements allowing positional access."""
    #--Positional list class
    A castro_mh
    class Position:
        """An abstraction representing the location of a single element."""
        A castro_mh
        def __init__(self, container, node):
            """Constructor should not be invoked by the user."""
            self._container = container
            self._node = node
        A castro_mh
        def element(self):
            """Return the element stored at this Position"""
            return self._node._element
        A castro_mh
        def __eq__(self, other):
            """Return True if other is a Position representing the same location."""
            return type(other) is type(self) and other._node is self._node
        A castro_mh
        def __ne__(self, other):
            """Return True if other does not represent the same location."""
            return not (self == other) #opposite of __eq__
    A castro_mh
    #--utility methods
    6 usages: A castro_mh
    def _validate(self, p):
        """Return position's node or raise appropriate error if invalid"""
        if not isinstance(p, self.Position):
            raise TypeError('p must be proper Position type')
        if p._container is not self:
            raise ValueError('p does not belong to this container')
        if p._node._next is None: #convention for deprecated nodes
            raise ValueError('p is no longer valid')
        return p._node
    A castro_mh
    #--utility method
    5 usages: A castro_mh
    def _make_position(self, node):
        """Return Position instance for given node (or None if sentinel)."""
        if node is None:
            return None
        return self.Position(self, node)

PositionList: delete()
```

```
13
14 """Return Position instance for given node (or None if sentinel)."""
15 if node is self._header or node is self._trailer:
16     return None #boundary violation
17 else:
18     return self.Position(self, node) #legitimate position
19
20 #--Accessors
21 10 usages (8 dynamic): A castro_mh
22 def first(self):
23     """Return the first Position in the list (or None if list is empty)."""
24     return self._make_position(self._header._next)
25 4 usages (4 dynamic): A castro_mh
26 def last(self):
27     """Return the last Position in the list (or None if list is empty)."""
28     return self._make_position(self._trailer._prev)
29 6 usages (6 dynamic): A castro_mh
30 def before(self, p):
31     """Return the Position just before Position p (or None if p is first)."""
32     node = self._validate(p)
33     return self._make_position(node._prev)
34 6 usages (6 dynamic): A castro_mh
35 def after(self, p):
36     """Return the Position just after Position p (or None if p is last)."""
37     node = self._validate(p)
38     return self._make_position(node._next)
39 A castro_mh
40 def __iter__(self):
41     """Generate forward iteration of the elements of the list"""
42     cursor = self.first()
43     while cursor is not None:
44         yield cursor.element()
45     cursor = self.after(cursor)
46
47 #--Mutators
48 4 usages: A castro_mh
49 def _insert_between(self, e, predecessor, successor):
50     """Add element between existing nodes and return new Position"""
51     node = super()._insert_between(e, predecessor, successor)
52     return self._make_position(node)
53 7 usages: A castro_mh
54 def add_first(self, e):
55     """Insert element e at the front of the list and return new Position."""
56     self._insert_between(e, self._header, self._header._next)
57
PositionList: delete()
```

```
DSALG01-IDB2 - main
Activity #2.py PositionalList.py LinkedStack.py main.py Activity #1 (Stack).py
4 usages: A castro.mh
61 def _insert_between(self, e, predecessor, successor):
62     """Add element e between existing nodes and return new Position"""
63     node = super()._insert_between(e, predecessor, successor)
64     return self._make_position(node)
65
66 7 usages: A castro.mh
67 def add_first(self, e):
68     """Insert element e at the front of the list and return new Position"""
69     return self._insert_between(e, self._header, self._header._next)
70
71 7 usages: A castro.mh
68 def add_last(self, e):
69     """Insert element e at the back of the list and return new Position"""
70     return self._insert_between(e, self._trailer._prev, self._trailer)
71
72 4 usages (4 dynamic): A castro.mh
71 def add_before(self, p, e):
72     """Insert element e into list before Position p and return new Position"""
73     original = self._validate(p)
74     return self._insert_between(e, original._prev, original)
75
76 4 usages: A castro.mh
75 def add_after(self, p, e):
76     """Insert element e into list after Position p and return new Position"""
77     original = self._validate(p)
78     return self._insert_between(e, original, original._next)
79
80 4 usages (4 dynamic): A castro.mh
79 def delete(self, p):
80     """Remove and return the element at Position p"""
81     original = self._validate(p)
82     return self._delete_node(original)  # inherited method returns element
83
84 3 usages (2 dynamic): A castro.mh
83 def replace(self, p, e):
84     """Replace the element at Position p with e"""
85     """Return the element formerly at Position P"""
86     original = self._validate(p)
87     old_value = original._element  # temporarily store old element
88     original._element = e  # replace with new element
89     return old_value  # return the old element value
90
PositionalList | delete()
DSALG01-IDB2 > FNALS > LinkedListActivities > PositionalList.py 79:23 (1 char) CRLF UTF-8 4 spaces Python 3.12
```

```
DSALG01-IDB2 - main
Activity #2.py PositionalList.py LinkedStack.py main.py Activity #1 (Stack).py
1 class LinkedStack:
2     """IFO Stack Implementation using a singly linked list for storage"""
3
4     #----- nested _Node class -----
5     A castro.mh
6     class _Node:
7         """lightweight non public class for storing a singly linked node"""
8         __slots__ = 'element', 'next'  # streamline memory usage
9
10        A castro.mh
11        def __init__(self, element, next):
12            self._element = element
13            self._next = next
14
15        #----- stack methods -----
16        A castro.mh
17        def __init__(self):
18            """Create an empty stack"""
19            self._head = None
20            self._size = 0
21
22        A castro.mh
23        def __len__(self):
24            """Return the number of elements in the stack"""
25            return self._size
26
27        A castro.mh
28        def is_empty(self):
29            """Return True if the stack is empty"""
30            return self._size == 0
31
32        9 usages: A castro.mh
33        def push(self, e):
34            """Add element e to the top of the stack"""
35            self._head = self._Node(e, self._head)
36            self._size += 1
37
38        4 usages: A castro.mh
39        def top(self):
40            """Return but do not remove the element at the top of the stack"""
41            """Raise empty exception if the stack is empty"""
42            if self.is_empty():
43                raise Exception('Stack is empty')
44            return self._head._element  # top of the stack is the head of the list
45
LinkedStack | _Node
DSALG01-IDB2 > FNALS > LinkedListActivities > LinkedStack.py 5:17 CRLF UTF-8 4 spaces Python 3.12
```

```
DSALG01-IDB2 - main
Activity #2.py PositionalList.py LinkedStack.py main.py Activity #1 (Stack).py
A castronm
14 def __init__(self):
15     """Create an empty Stack"""
16     self._head = None
17     self._size = 0
18 A castronm
19 def __len__(self):
20     """Return the number of elements in the stack"""
21     return self._size
22 A castronm
23 def is_empty(self):
24     """Return True if the stack is empty"""
25     return self._size == 0
26
27 @usage A castronm
28 def push(self, e):
29     """Add element e to the top of the stack"""
30     self._head = self._Node(e, self._head)
31     self._size += 1
32 4 usage A castronm
33 def top(self):
34     """Return but do not remove the element at the top of the stack"""
35     """Raise empty exception if the stack is empty"""
36     if self.is_empty():
37         raise Exception('Stack is empty')
38     return self._head._element
39 7 usage A castronm
40 def pop(self):
41     """Remove and return the elements for the top of the stack (LIFO)"""
42     """Raise Empty exception if the stack is empty"""
43     if self.is_empty():
44         raise Exception('The stack is empty')
45     answer = self._head._element
46     self._head = self._head._next
47     self._size -= 1
48     return answer
LinkedStack - _Node
DSALG01-IDB2 > FNALS > LinkedStack_Activities > LinkedStack.py 5:17 CRLF UTF-8 4 spaces Python 3.12
```

```
Activity #2.py PositionalList.py LinkedStack.py main.py Activity #1 (Stack).py
1 from LinkedStack import LinkedStack as LinkStack
2 from PositionalList import PositionalList as PositionalList
3 import re
4
5 '''Part 1'''
6 expression = input("Enter an Expression: ")
7
8 precedence = {'+': 1, '-': 1, '*': 2, '/': 2, '^': 3}
9 operators = set(precedence.keys())
10 stack = LinkStack()
11 temp_stack = LinkStack()
12
13 expression = re.sub(pattern=r'([+~*/^()])', repl=r' \1 ', expression)
14 tokens = expression.split()
15
16 for token in tokens:
17     if token.isalnum():
18         temp_stack.push(token)
19     elif token in operators:
20         while not stack.is_empty() and stack.top() != '(' and precedence.get(stack.top(), 0) >= precedence[token]:
21             temp_stack.push(stack.pop())
22         stack.push(token)
23     elif token == '(':
24         stack.push(token)
25     elif token == ')':
26         while not stack.is_empty() and stack.top() != '(':
27             temp_stack.push(stack.pop())
28         stack.pop()
29     else:
30         raise ValueError(f"Unknown token: {token}")
31 while not stack.is_empty():
32     if stack.top() == '(':
33         raise ValueError("Mismatched parentheses")
34     temp_stack.push(stack.pop())
35
36 postfix1_stack = LinkStack()
37 while not temp_stack.is_empty():
38     postfix1_stack.push(temp_stack.pop())
39
40 print("Postfix expression:", end=' ')
41 while not postfix1_stack.is_empty():
```

```

print("Postfix expression:", end=' ')
while not postfix1_stack.is_empty():
    print(postfix1_stack.pop(), end=' ')
print()

```

'''Part 2'''

1 usage

```

def insertion_sort_descending(L):
    '''Sort the Positional List of comparable elements into non decreasing order.'''
    if len(L) > 1: #otherwise, no need to sort it
        marker = L.first()
        while marker != L.last():
            pivot = L.after(marker)#next item to place
            value = pivot.element()
            if value < marker.element():#pivot is already sorted
                marker = pivot#pivot becomes new marker
            else:#must relocate pivot
                walk = marker#find the leftmost value greater than pivot
                while walk != L.first() and L.before(walk).element() < value:
                    walk = L.before(walk)
                L.delete(pivot)#remove pivot
                L.add_before(walk, value)#insert pivot

```

1 usage

```

def insertion_sort(L):
    '''Sort the Positional List of comparable elements into non decreasing order.'''
    if len(L) > 1: #otherwise, no need to sort it
        marker = L.first()
        while marker != L.last():
            pivot = L.after(marker)#next item to place
            value = pivot.element()
            if value > marker.element():#pivot is already sorted
                marker = pivot#pivot becomes new marker
            else:#must relocate pivot
                walk = marker#find the leftmost value greater than pivot
                while walk != L.first() and L.before(walk).element() > value:
                    walk = L.before(walk)
                L.delete(pivot)#remove pivot
                L.add_before(walk, value)#insert pivot

```

```

P = PositionalList()

P.add_first(1)
P.add_last(72)
P.add_last(81)
P.add_last(25)
P.add_last(65)
P.add_last(91)
P.add_last(11)

insertion_sort(P)
print("Insertion Sort Ascending:", end=' ')
for x in P:
    print(x, end=' ')
print()

insertion_sort_descending(P)
print("Insertion Sort Descending:", end=' ')
for x in P:
    print(x, end=' ')

```

OUTPUT FOR PROBLEM:

```

"C:\Program Files\Python312\python.exe" "Z:\DSAL601-IDB2\FINALS\LinkedList_Activites\Activity #2.py"
Enter an Expression: 1 + 2 * 3
Postfix expression: 1 2 3 * +
Insertion Sort Ascending: 1 11 25 65 72 81 91
Insertion Sort Descending: 91 81 72 65 25 11 1
Process finished with exit code 0

```