Started on	Thursday, 17 July 2025, 3:14 PM
State	Finished
Completed on	Thursday, 17 July 2025, 3:51 PM
Time taken	37 mins 1 sec
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python class program to store all odd numbers between 500 and 600 in a reverse order using list comprehension

For example:

Input	Result
500 -2 600	[599, 597, 595, 593, 591, 589, 587, 585, 583, 581, 579, 577, 575, 573, 571, 569, 567, 565, 563, 561, 559, 557, 555, 553, 551, 549, 547, 545, 543, 541, 539, 537, 535, 533, 531, 529, 527, 525, 523, 521, 519, 517, 515, 513, 511, 509, 507, 505, 503, 501]

Answer: (penalty regime: 0 %)

```
class Generate:
    def __init__(self, first,d,last):
        self.first = first
        self.d = d
        self.last=last
    def Ap_generate(self):
        L=[i for i in range(self.last-1,self.first,self.d)]
        return L
Series = Generate(500,-2,600)
print(Series.Ap_generate())
```

	Input	Expected	Got	
✔	500 -2 600	[599, 597, 595, 593, 591, 589, 587, 585, 583, 581, 579, 577, 575, 573, 571, 569, 567, 565, 563, 561, 559, 557, 555, 553, 551, 549, 547, 545, 543, 541, 539, 537, 535, 533, 531, 529, 527, 525, 523, 521, 519, 517, 515, 513, 511, 509, 507, 505, 503, 501]	[599, 597, 595, 593, 591, 589, 587, 585, 583, 581, 579, 577, 575, 573, 571, 569, 567, 565, 563, 561, 559, 557, 555, 553, 551, 549, 547, 545, 543, 541, 539, 537, 535, 533, 531, 529, 527, 525, 523, 521, 519, 517, 515, 513, 511, 509, 507, 505, 503, 501]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Define a function to delete the first element in the given linked list.

Answer: (penalty regime: 0 %)

Reset answer

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class delete_front:
    def __init__(self):
        self.head = None
    def removeFirstNode(self):
        if self.head is None:
            print("List is already empty.")
        else:
            self.head = self.head.next
    def push(self, data):
        if self.head is None:
            self.head = Node(data)
            return
        temp = Node(data)
        temp.next = self.head
        self.head = temp

    def display(self):
        temp1 = self.head
```

	Input	Expected	Got	
✔	5 2 6 4 8 9	Enter the number of elements to push: 8 4 6 2	Enter the number of elements to push: 8 4 6 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Type a python function to insert elements at the beginning of the doubly linked list.

Answer: (penalty regime: 0 %)

Reset answer

```
class Node:
    def __init__(self, data):
        self.item = data
        self.nref = None
        self.pref = None

class DoublyLinkedList:
    def __init__(self):
        self.start_node = None

    def insert_in_emptylist(self, data):
        if self.start_node is None:
            new_node = Node(data)
            self.start_node = new_node
        else:
            print("list is not empty")

    def insert_at_start(self, data):
        new_node = Node(data)
        if self.start_node is None:
            self.start_node = new_node
        else:
```

	Expected	Got	
✔	10 20 30 40	10 20 30 40	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to traverse the elements in forward and reverse direction in doubly linked list.

Answer: (penalty regime: 0 %)

Reset answer

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        if self.head is not None:
            self.head.prev = new_node
        self.head = new_node

    def append(self, new_data):
        new_node = Node(new_data)
        if self.head is None:
            self.head = new_node
            return
```

	Input	Expected	Got	
✔	50 10 20 100	Insert the element to add at the end Insert the element to add at the beginning Insert the element to add at the beginning Insert the element to add at the end Created DLL is: Traversal in forward direction 20 10 50 100 Traversal in reverse direction 100 50 10 20	Insert the element to add at the end Insert the element to add at the beginning Insert the element to add at the beginning Insert the element to add at the end Created DLL is: Traversal in forward direction 20 10 50 100 Traversal in reverse direction 100 50 10 20	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to insert an element in the specified position in singly linked list.

Answer: (penalty regime: 0 %)

Reset answer

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def traverse_list(self):
        if self.head is None:
            print("List has no element")
            return
        else:
            n = self.head
            while n is not None:
                print(n.data , " ")
                n = n.next

    def insert_at_start(self, data):
        new_node = Node(data)
        new_node.next = self.head
```

	Expected	Got	
✔	After inserting elements at the end 25 35 45 After inserting elements at the beginning 15 25 35 45 Inserting elements at the specific position 15 40 25 35 45	After inserting elements at the end 25 35 45 After inserting elements at the beginning 15 25 35 45 Inserting elements at the specific position 15 40 25 35 45	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.