The goal of this Lab Exercise is to practice writing programs using Circular Queue and Double Linked List.

Q1. Write a program to simulate two queues at a grocery store. One queue is for normal customers and the other is for VIP customers. Normal customers can only wait in the Normal customers' queue, and VIP customers can only wait in the VIP customers' queue.
Your program should start by asking the user if they want to add a customer to the queue, serve the next customer in the queue, or exit.

If the user chooses to add a customer, the program asks the user to enter the name of the customer followed by whether or not the customer is a VIP. If the customer is a VIP then the customer's name is added to the VIP queue else the customer's name is added to the Normal Queue. After EACH Add operation, the program prints the customer's name and the queue (Normal or VIP) the customer has been added to. The program also prints both queues after EACH Add operation. After the add operation is completed, the user is asked to enter Add,Serve or Exit again. **Your program will continue to prompt the user repeatedly with these options and perform the task associated to these operations until the user enters Exit.** Here is a partial sample output of the program that shows the above description.

```
Add, Serve, or Exit:Add
Enter the name of the person to add:Fred
Is the customer VIP?True
Add Fred to VIP line
Normal customers queue: ]]
VIP customers queue: ]Fred]

Add, Serve, or Exit:Add
Enter the name of the person to add:Bob
Is the customer VIP?False
Add Bob to the line.
Normal customers queue: ]Bob]
VIP customers queue: ]Fred]
```

If the user chooses to serve a customer, the program will check the VIP Queue first. If the VIP Queue is not empty, a customer from that queue will be served first. If the VIP Queue is empty, a customer from the Normal Queue is served. After the serve operation, the program will print the name of the customer who has been served and the content of each queue. Here is a partial output from the program that shows what is displayed after the serve operation:

```
Add, Serve, or Exit:Add
Enter the name of the person to add:Barney
Is the customer VIP?True
add Barney to VIP line
Normal customers queue: ]]
VIP customers queue: ]Barney]

Add, Serve, or Exit:Add
Enter the name of the person to add:Wilma
Is the customer VIP?False
add Wilma to the line.
Normal customers queue: ]Wilma]
VIP customers queue: ]Barney]

Add, Serve, or Exit:Serve
Barney has been served
Normal customers queue: ]Wilma]
VIP customers queue: ]]
```

If the user wants to add a customer to a queue (Normal or VIP) that is full, the program should print one of the following error message:

```
Error: Normal customers queue is full
```

or

```
Error: VIP customers queue is full
```

The program however should continue to ask the user if they wish to Add, Serve or Exit.

If both Normal and VIP Queues are empty and if the user requests a serve from the queue, then the program should print the following error message:

```
Error: Queues are empty
```

However, in this case also the program continues to ask the user if they wish to Add, Serve or Exit. If the user enters the word Exit, the program should end by printing the word Quitting.

Please note the following:

- You may restrict the capacity of each queue to 3 for testing simplicity.

- `The CircularQueue` class is given to you. Please import the class in your program by writing the following:

  ```
  from CircularQueue import CircularQueue
  ```

Here is the **complete sample output** from the program:

```
Add, Serve, or Exit:Add
Enter the name of the person to add:Fred
Is the customer VIP?True
add Fred to VIP line
Normal customers queue: ]]
VIP customers queue: ]Fred]

Add, Serve, or Exit:Add
Enter the name of the person to add:Barney
Is the customer VIP?False
add Barney to the line.
Normal customers queue: ]Barney]
VIP customers queue: ]Fred]

Add, Serve, or Exit:Add
Enter the name of the person to add:Wilma
Is the customer VIP?False
add Wilma to the line.
Normal customers queue: ]Barney,Wilma]
VIP customers queue: ]Fred]

Add, Serve, or Exit:Add
Enter the name of the person to add:Pebbles
Is the customer VIP?False
add Pebbles to the line.
Normal customers queue: ]Barney,Wilma,Pebbles]
VIP customers queue: ]Fred]
```

```
Add, Serve, or Exit:Add
Enter the name of the person to add:Flint
Is the customer VIP?False
Error: Normal customers queue is full
Normal customers queue: ]Barney,Wilma,Pebbles]
VIP customers queue: ]Fred]

Add, Serve, or Exit:Serve
Fred has been served
Normal customers queue: ]Barney,Wilma,Pebbles]
VIP customers queue: ]]

Add, Serve, or Exit:Serve
Barney has been served
Normal customers queue: ]Wilma,Pebbles]
VIP customers queue: ]]

Add, Serve, or Exit:Serve
Wilma has been served
Normal customers queue: ]Pebbles]
VIP customers queue: ]]

Add, Serve, or Exit:Serve
Pebbles has been served
Normal customers queue: ]]
VIP customers queue: ]]

Add, Serve, or Exit:Serve
Error: Queues are empty
Normal customers queue: ]]
VIP customers queue: ]]

Add, Serve, or Exit:Exit
Quitting
```

Q2 You are given the file **DoubleLinkedList.py**. There are two classes in the file – the **DLinkedListNode class and the DLinkedList class**. Some of the methods in the DLinkedList class have been completed for you. Your task is to complete the implementation of the **append** and **insert method** in DLinkedList class. Once you have completed the methods, you can use the code in the main function to test the **append and insert method**. Here is the output that you should get if the methods have been implemented correctly.

```
List after append and add
Z<->2<->4<->6<->77<->A<->

After inserting the word start at position 0
start<->Z<->2<->4<->6<->77<->A<->

After inserting the word end at position 7
start<->Z<->2<->4<->6<->77<->A<->end<->

After inserting middle at position 4
start<->Z<->2<->4<->middle<->77<->A<->end<->
```