10. Circular linked list

Theory

A circular linked list is a sequence of elements in which every element has a link to its next element in the sequence and the last element has a link to the first element.

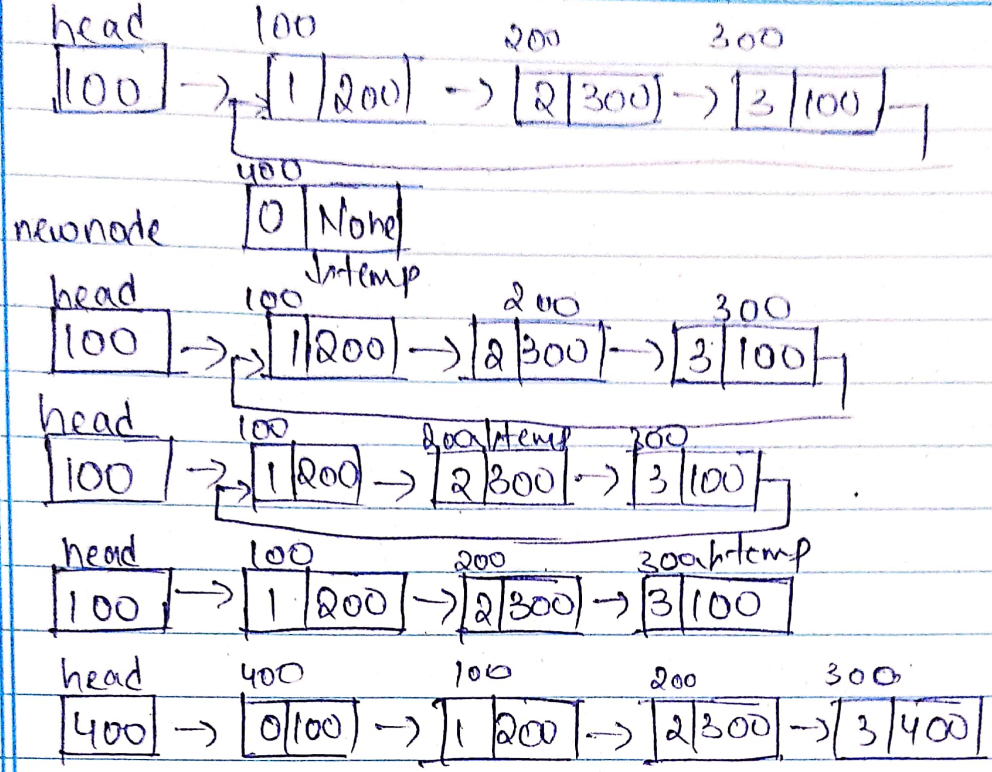
There are two types of circular linked list
* Circular single linked list
* Circular Double linked list

Circular single linked list—

Operations
* Insertion
* Deletion

Insertion at Beginning

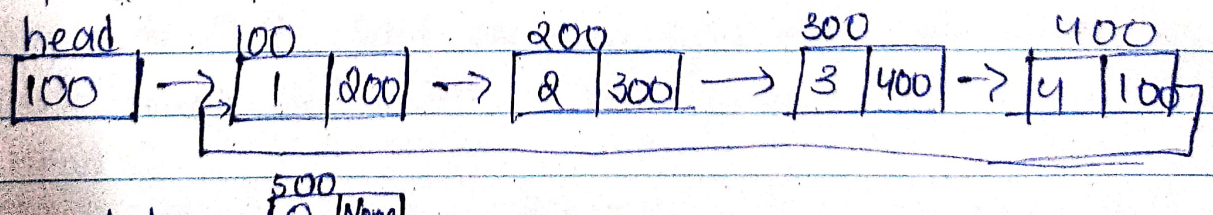
Step 1 - Create a newNode with given value
Step 2 - Check whether list is Empty (head == NULL)
Step 3 - If it is Empty then, 'set head = newNode and newNode -> next = head'.
Step 4 - If it is Not empty then, define a Node pointer 'temp' and initialize with 'head'.
Step 5 - Keep moving the 'temp' to its next node until it reaches to the last node (until 'temp -> next = head').
Step 6 - Set 'newNode -> next = head', 'head = newNode' and 'temp -> next = head'.



Inserting At End

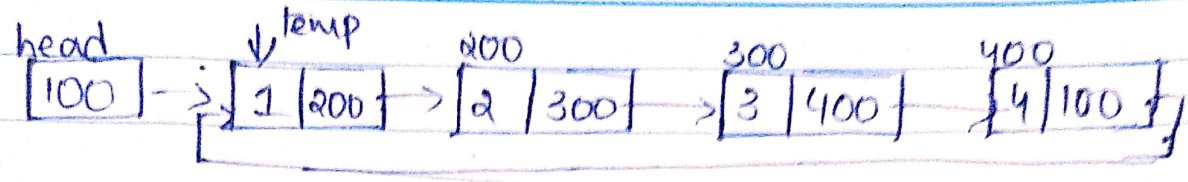
Algorithm:

Step 1 - Create a newNode with given value
Step 2 - Check whether list is Empty (head == NULL)
Step 3 - If it is Empty then, set head = newNode and newNode->next = head
Step 4 - If it is Not Empty then, define a node pointer temp and initialize with head.
Step 5 - Keep moving the temp to its next node until it reaches to the last node in the list (until temp->next == head)
Step 6 - Set temp->next = newNode and newNode->next = head.

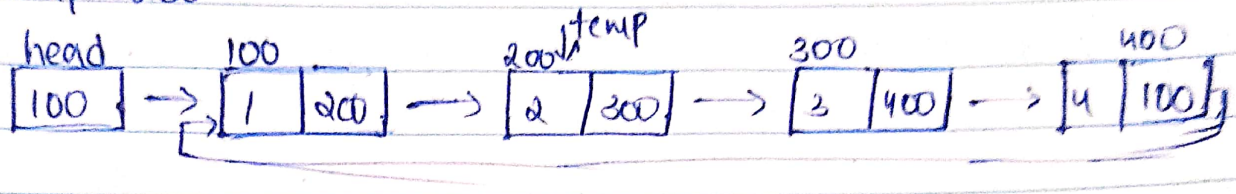


newNode =

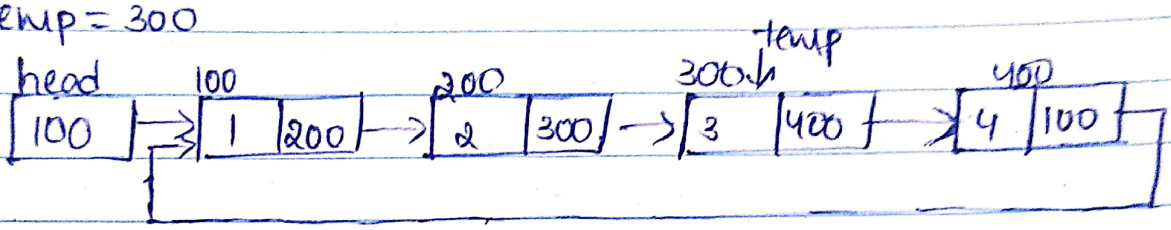
temp = 100



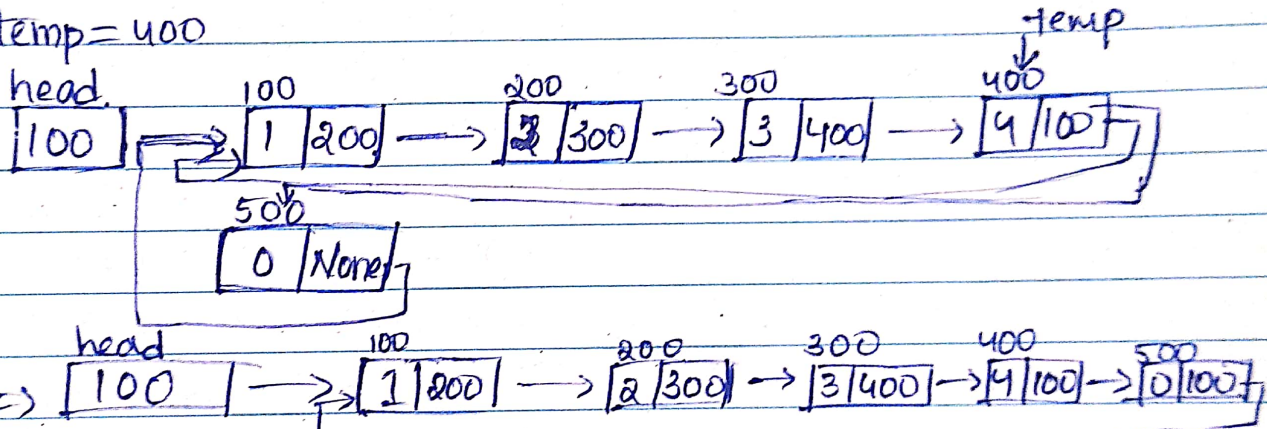
temp = 200



temp = 300



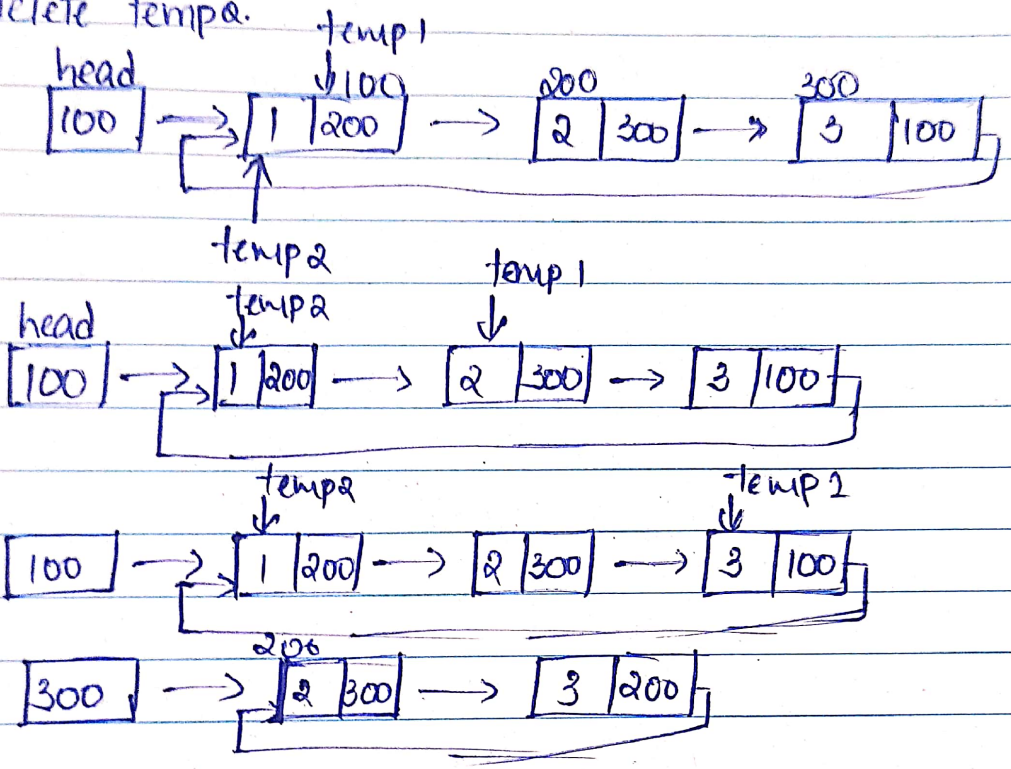
temp = 400



Delete from Beginning

Step 1- Check whether list is Empty (head == NULL)
Step 2- If it is empty then, display 'List is empty!!! Deletion is not possible' and terminate the function
Step 3- If it is Not Empty then, define two Node pointers 'temp1' and 'temp2' and initialize both 'temp1' and 'temp2' with head.
Step 4- Check whether list is having only one node (temp1->next == head)

Step 5 - If it is TRUE then set head = NULL and delete temp 1 (setting empty list conditions)
Step 6 - If it is FALSE move the temp1 until it reaches to the last node. (until temp 1 -> next == head)
Step 7 - Then set head = temp 2 -> next, temp 1 -> next = head and delete temp 2.



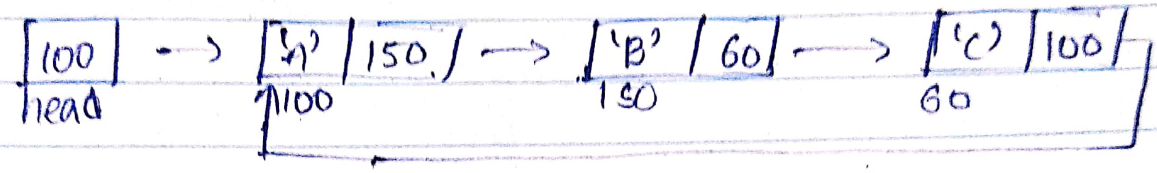
Delete from End

Step 1 - Check whether list is Empty (head == NULL)
Step 2 - If it is Empty then, display 'list is Empty!!! Deletion is not possible' and terminate the function.
Step 3 - If it is Not Empty then, define two Node pointers 'temp 1' and 'temp 2' and initialize 'temp 1' with head.
Step 4 - Check whether list has only one Node (temp 1 -> next == head)

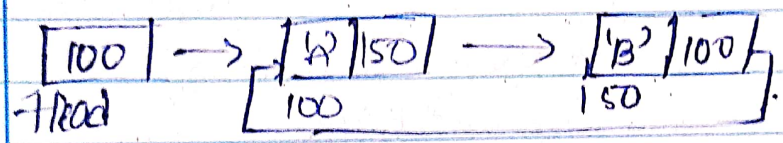
Step 5 - If it is TRUE. Then, set head = NULL and delete temp 1. And terminate from the function.

Step 6: If it is FALSE then set 'temp2 = temp1' and move temp1 to its next node. Repeat the same until temp1 reaches to the last node in the list.

Existing CSLL:



After deletion at end:



Code:-

```
while(1):
    print("\n\n Choose an option:
        1. Insert at beginning
        2. Insert at End
        3. Delete at beginning
        4. Delete at End
        5. Display
        choice = int(input())
class Node:
        def __init__(self,data):
            self.data = data
            self.next = None

class CSLL:
        def __init__(self):
            self.head = None
        def insert_begin(self,data):
            new_node = Node(data)
            if self.head is None:
                self.head = new_node
                new_node.next = self.head
            else:
                temp = self.head
                while temp.next != self.head:
                    temp = temp.next
                new_node.next = self.head
                temp.next = new_node
                self.head = new_node
```

```
def insert_end(self,data):
    new_node = Node(data)
    if self.head is None:
        self.head = new_node
        new_node.next = self.head
    else:
        temp = self.head
        while temp.next != self.head:
            temp = temp.next
        new_node.next = self.head
        temp.next = new_node
def delete_begin(self):
    if self.head is None:
        print("elements not present")
    else:
        temp = self.head
        while temp.next != self.head:
            temp = temp.next
        print(self.head.data,"deleted succesfully")
        temp.next = self.head.next
        self.head = self.head.next
def delete_end(self):
    if self.head is None:
        print("No elements present")
    else:
        temp = self.head
        while temp.next.next != self.head:
            temp = temp.next
        print(temp.next.data, "delete succesfully")
        temp.next = self.head
```

```
def display(self):
        temp = self.head
        while temp.next!=self.head:
            print(temp.data, end="--->")
            temp=temp.next
        print(temp.data)
if choice==1:
    data=input("enter data")
    insert_begin(data)
if choice==2:
    data=input("enter data")
    insert_begin(data)
if choice==3:
    data=input("enter data")
    insert_begin(data)
if choice==4:
    data=input("enter data")
    insert_begin(data)
if choice==5:
    data=input("enter data")
    insert_begin(data)
```

Output :-

User choice = 1 Data = 60

User choice = 1 Data = 30

User choice = 2 Data = 40

User choice = 3

User choice = 3

60 --> 40

Circular Double linked list:

Menu:

1: Insert Begin

2: Insert End

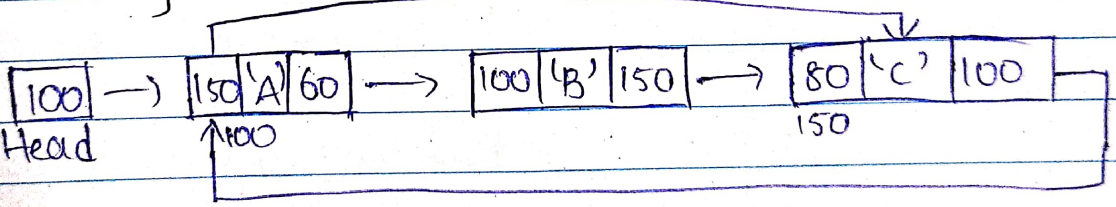
3: Delete Begin

4: Delete End

5: Display

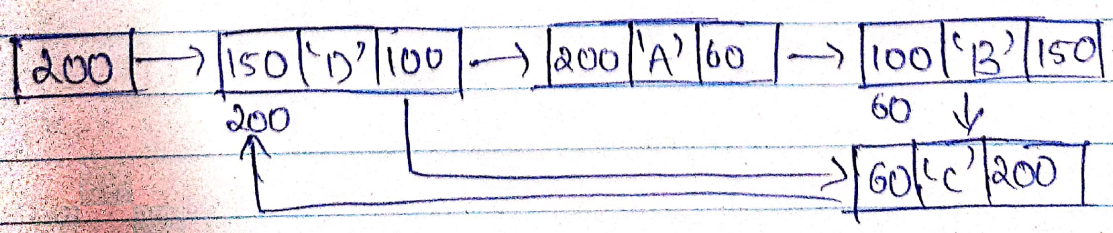
Process with example:-

(i) Existing CDLL:

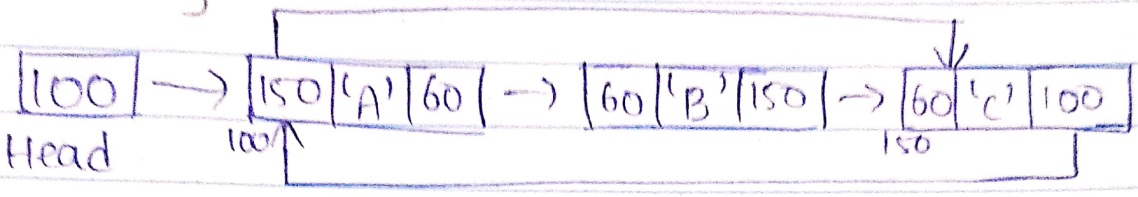


user choice = 1 Data = 'D'

After insertion at begin:

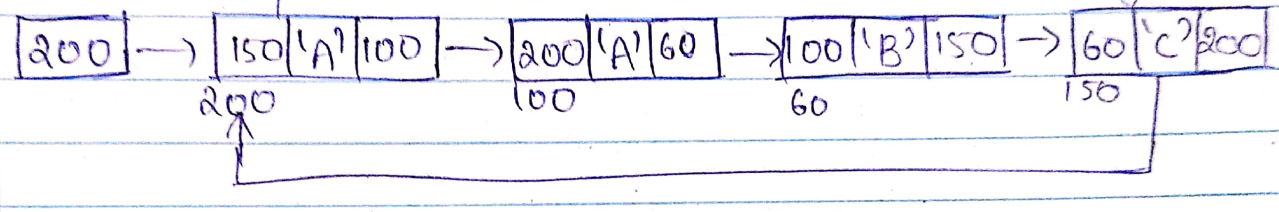


(ii) Existing CDLL:-

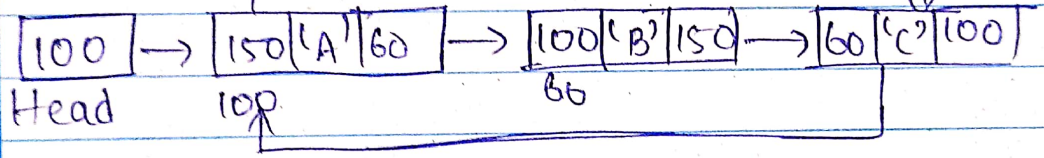


User choice = 2 Data = 'D'

After insertion at beginning:

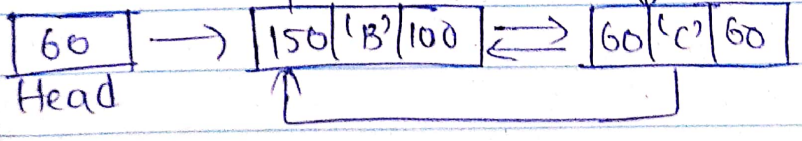


(iii) Existing CDLL:-

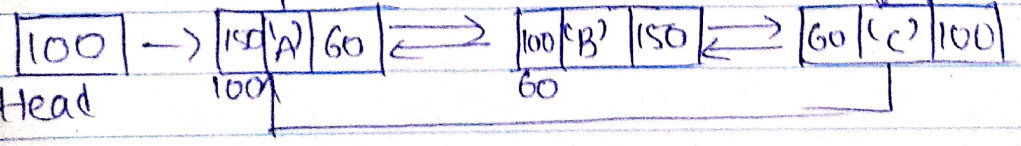


User choice = 3

After deletion at end

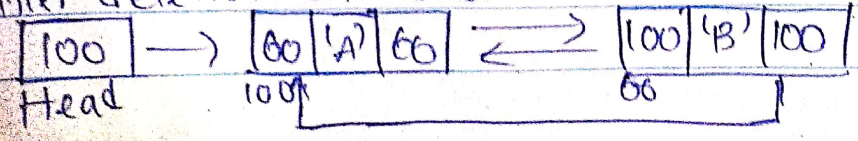


(iv) Existing CDLL:-

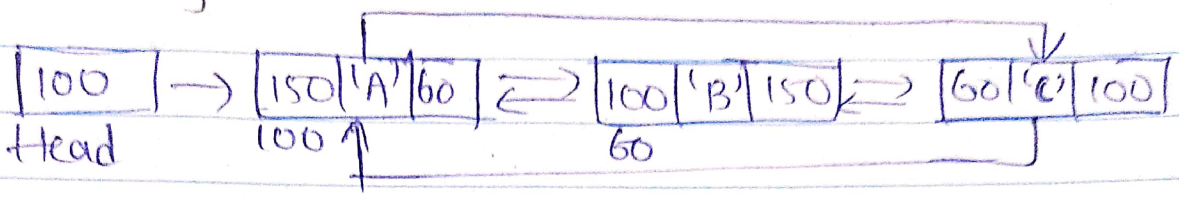


User choice = 4

After deletion at end:

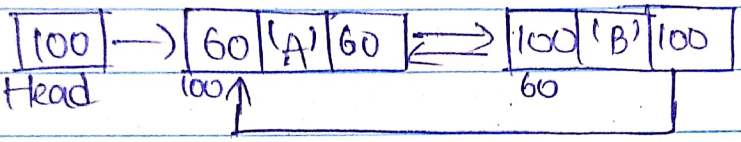


(iv) Existing CDLL:



user choice = 4

After deletion at end:



Code:-

```
while(1):
    print("""choose an option:
        1. Insert at beginning
        2. Insert at end
        3. Delete begin
        4. Delete end
        5. Display
choice = int(input())

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

Class CDLL:
    def __init__(self):
            self.head =None
    def insert_begin(self,data):
            new_node = Node(data)
            if self.head is None:
                self.head = new_node
                new_node.next = self.head
                new_node.prev = self.head
            else:
            temp = self.head
            while temp.next != self.head
                temp = temp.next
```

```
        temp.next = new_node
        new_node.next = self.head
        self.head = new_node
        new_node.prev = temp
    def insert_end(self,data):
        new_node = Node(data)
        temp = self.head
        if self.head is None:
            self.head = new_Node
            new_node.next = self.head
            new_node.prev = self.head
        Else:
            while temp.next != self.head:
                temp = temp.next
            temp.next = new_node
            new_node.next = self.head
            new_node.prev = temp
            self.head.prev = new_node
    def delete_begin(self):
        if self.head is None:
            print("No elements found")
        else:
            temp = self.head
            while temp.next != self.head
                temp = temp.next
            temp.next = self.head.next
            self.head = temp.next
            self.head.prev = temp
    def delete_end(self):
        if self.head is None:
```

```
                print("No elements found")
        else:
            temp=self.head
            while temp.next.next!=self.head:
                temp=temp.next
            temp.next=self.head
            self.head.prev=temp
    def display(self):

            temp=self.head
            while temp.next!=self.head
                print(temp.data, end="-->")
                temp=temp.next
            print(temp.data)
    if choice==1:
        data=input("enter data")
        insert_begin(data)
    if choice==2:
        data=input("enter data")
        insert_end(data)
    if choice==3:
        delete_begin()
    if choice==4:
        delete_end()
    if choice==5:
        display()
```

Output:-

User Choice = 1
User Choice = 2
User Choice = 2
User Choice = 5

Data = A
Data = B
Data = C

A ---> B ---> C