Circular Linked List

1] Circular Linked List in Python:

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

def printCircular(head):
    if head==None:
        return

    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next



head=Node(10)
head.next=Node(5)
head.next.next=Node(20)
head.next.next.next=Node(15)
head.next.next.next.next=head
printCircular(head)
```

OUTPUT :

10 5 20 15

2] Circular Linked List Traversal :

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

def printCircular(head):
    if head==None:
        return
    print(head.key,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.key,end=" ")
        curr=curr.next

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=head

printCircular(head)
```

OUTPUT :

10 20 30

3] Insert at the beginning of Circular linked list in O(n) :

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

def insertBeg(head,x):
    temp=Node(x)
    if head==None:
        temp.next=temp
        return temp

    curr=head
    while curr.next!=head:
        curr=curr.next

    curr.next=temp
    temp.next=head
    return temp

def printCircular(head):
    if head==None:
        return

    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()

head=Node(20)
head.next=Node(30)
head.next.next=head
printCircular(head)

head=insertBeg(head,10)

printCircular(head)
```

OUTPUT : 20 30

10 20 30

4] Insert at the beginning of Circular linked list in constant time :

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

def insertBeg(head,x):
    temp=Node(x)
    if head==None:
        temp.next=temp
        return temp
    else:
        temp.next=head.next
        head.next=temp

        head.data,temp.data=temp.data,head.data   #swapping data
        return head

def printCircular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()


head=Node(20)
head.next=Node(30)
head.next.next=head
printCircular(head)

head=insertBeg(head,10)

printCircular(head)
```

OUTPUT :

20 30

10 20 30

5] Insert At the end of Circular list in O(n) :

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


def insertEnd(head,x):
    temp=Node(x)
    if head==None:
        temp.next=temp
        return temp
    else:
        curr=head
        while curr.next!=head:
            curr=curr.next

        curr.next=temp
        temp.next=head

        return head

def printCircular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()

head=Node(10)
head.next=Node(20)
head.next.next=head
printCircular(head)

head=insertEnd(head,30)

printCircular(head)
```

OUTPUT : 10 20

10 20 30

6] Insert at the end of circular linked list in constant time :

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

def inserEnd(head,x):
    temp=Node(x)

    if head==None:
        temp.next=temp
        return temp
    else:
        temp.next=head.next
        head.next=temp

        temp.data,head.data=head.data,temp.data   #swapping data
        return temp


def printCircular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()

head=Node(10)
head.next=Node(20)
head.next.next=head

printCircular(head)

head=inserEnd(head,30)

printCircular(head)
```

OUTPUT : 10 20

10 20 30

7] Delete Head of Circular list in O(n):

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

def deleteHead(head):
    if head==None:
        return None
    elif head==head.next:
        return None

    curr=head
    while curr.next!=head:
        curr=curr.next
    curr.next=head.next
    return curr.next

def printCicular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=head

printCicular(head)

head=deleteHead(head)

printCicular(head)
```

OUTPUT :

10 20 30

20 30

8] Delete Head of Circular Linked List in constant time :

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

def deleteHead(head):
    if head==None:
        return None
    elif head.next==head:
        return None
    else:
        head.data=head.next.data
        head.next=head.next.next
        return head

def printCircular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next

    print()


head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=head

printCircular(head)

head=deleteHead(head)

printCircular(head)
```

OUTPUT :

10 20 30

20 30

9] delete Kth Node of Circular Linked List :

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

def delHead(head):
    if head==None:
        return None
    elif head.next==head:
        return None

    else:
        head.data=head.next.data
        head.next=head.next.next
        return head

def delKth(head,k):
    if head==None:
        return head

    elif k==1:
        return delHead(head)

    else:
        curr=head

        for i in range(k-2):
            curr=curr.next

        curr.next=curr.next.next
        return head

def printCircular(head):
    if head==None:
        return
    print(head.data,end=" ")
    curr=head.next
    while curr!=head:
        print(curr.data,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
```

```
head.next.next=Node(15)
head.next.next.next=Node(30)
head.next.next.next.next=head

printCircular(head)

head=delKth(head,3)

printCircular(head)
```

OUTPUT :

10 20 15 30

10 20 30