Insert at Begin of Circular Linked List

Insertion at the beginning of the circular list

Naive Method:

C++ Java

```
 import java.util.*;
import java.io.*;
import java.lang.*;

class Node{
        int data;
        Node next;
        Node(int d){
            data=d;
            next=null;
        }
    }

class Test {

    public static void main(String args[])
    {
        Node head=new Node(10);
        head.next=new Node(20);
        head.next.next=new Node(30);
        head.next.next.next=head;
        head=insertBegin(head,15);
        printlist(head);

    }

    public static void printlist(Node head){
        if(head==null)return;
```

```
            System.out.print(r.data+" ");
```

```
    static Node insertBegin(Node head,int x){
        Node temp=new Node(x);
        if(head==null)
            temp.next=temp;
        else{
            Node curr=head;
            while(curr.next!=head)
                curr=curr.next;
            curr.next=temp;
            temp.next=head;
        }
        return temp;
    }
}
```

Output:

```
15 10 20 30
```

Time Complexity :-

O(n) - We are traversing to the last node and inserting the new node after the last node, then the next of the new node is made to point the head of the link list.

Efficient Method:

C++ | Java

```
import java.util.*;
import java.io.*;
import java.lang.*;
```

```
        int data;
        next=null;
    }
}

class Test {

    public static void main(String args[])
    {
        Node head=new Node(10);
        head.next=new Node(20);
        head.next.next=new Node(30);
        head.next.next.next=head;
        head=insertBegin(head,15);
        printlist(head);

    }

    public static void printlist(Node head){
        if(head==null)return;
        Node r=head;
        do{
            System.out.print(r.data+" ");
            r=r.next;
        }while(r!=head);
    }

    static Node insertBegin(Node head,int x){
        Node temp=new Node(x);
        if(head==null){
            temp.next=temp;
            return temp;
        }
        else{
            temp.next=head.next;
            head.next=temp;
            int t=head.data;
            head.data=temp.data;
```

```
}
```


Output:

```
15 10 20 30
```


Time Complexity :-

O(1) - We are not traversing to the last node instead we attach the new node after head and interchange the data of head and the new node.