Insert at the end of Circular Linked List

Insertion at the end of the circular list

Naive Approach :-

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=insertEnd(head,15);
        printlist(head);

    }
```

```
        do{

    }

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

Output:

```
10 20 30 15
```

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 Approach :-

C++ Java

```
import java.util.*;
```

```
        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=insertEnd(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 insertEnd(Node head,int x){

            temp.next=temp;
            return temp;
        }
        else{
            temp.next=head.next;
            head.next=temp;
```

```
        temp.data=t;
    }
```

Output :-

```
10 20 30 15
```

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.