```c
#include<stdio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};

void linkedListTraversal(struct Node *head){
    struct Node *ptr = head;
    do{
        printf("Element is %d\n", ptr->data);
        ptr = ptr->next;
    }while(ptr!=head);
}

struct Node * insertAtFirst(struct Node *head, int data){
    struct Node * ptr = (struct Node *) malloc(sizeof(struct Node))
    ptr->data = data;

    struct Node * p = head->next;
    while(p->next != head){
        p = p->next;
    }
    // At this point p points to the last node of this circular

    p->next = ptr;
    ptr->next = head;
    head = ptr;
    return head;

}

int main(){

    struct Node *head;
    struct Node *second;
    struct Node *third;
```

```
40:     struct Node *fourth;
41:
42:     // Allocate memory for nodes in the linked list in Heap
43:     head = (struct Node *)malloc(sizeof(struct Node));
44:     second = (struct Node *)malloc(sizeof(struct Node));
45:     third = (struct Node *)malloc(sizeof(struct Node));
46:     fourth = (struct Node *)malloc(sizeof(struct Node));
47:
48:     // Link first and second nodes
49:     head->data = 4;
50:     head->next = second;
51:
52:     // Link second and third nodes
53:     second->data = 3;
54:     second->next = third;
55:
56:     // Link third and fourth nodes
57:     third->data = 6;
58:     third->next = fourth;
59:
60:     // Terminate the list at the third node
61:     fourth->data = 1;
62:     fourth->next = head;
63:
64:       printf("Circular Linked list before insertion\n");
65:     linkedListTraversal(head);
66:     head = insertAtFirst(head, 54);
67:     head = insertAtFirst(head, 58);
68:     head = insertAtFirst(head, 59);
69:     printf("Circular Linked list after insertion\n");
70:     linkedListTraversal(head);
71: //Note that the rest of the operations are similar to that o
72:
73:     return 0;
74: }
75:
```