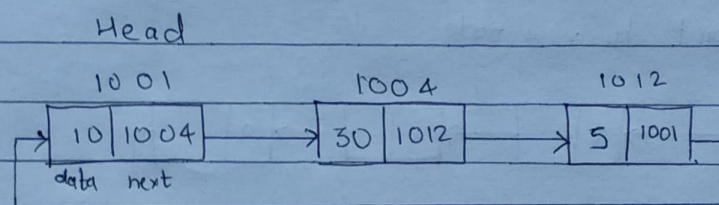


AIM: Implement Circular Linked list using ADT

THEORY:

1. Circular linked list is a variation of linked list in which the first element points to the last element
2. 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.

For Example,



• Operations on Circular Linked List:

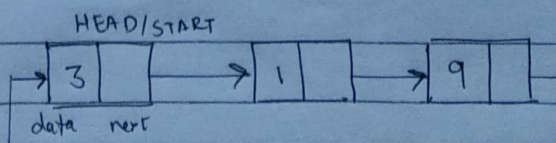
1) Traversing

- a) Traversing a linked list means accessing the nodes of the list in order to perform some processing on them.
- b) A circular linked list contains a pointer variable START which stores address of first node of list.

2) Insertion:

a) Insertion of node at beginning of circular linked list:

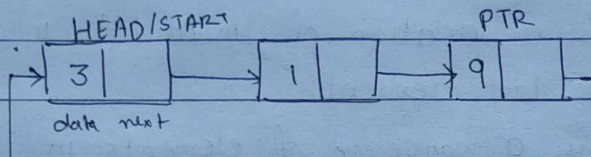
Consider the linked list shown below. Suppose we want to add a new node with data 11 as first node of list.



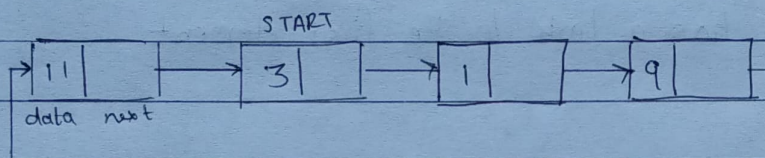


Allocate memory for new node and set its data part to 11.

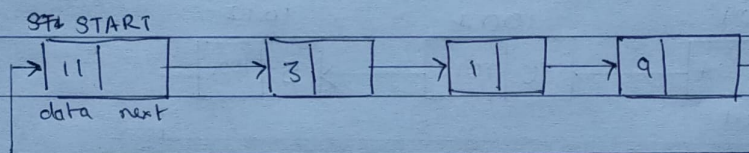
Then take a pointer variable PTR that points to the start node of the list. Move PTR so that it now points to last node of list.



Add a new node between PTR and START

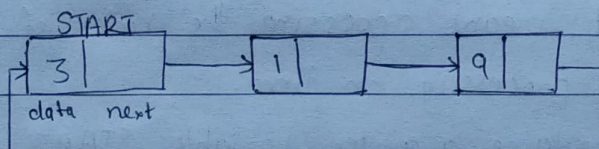


Make START point to new node



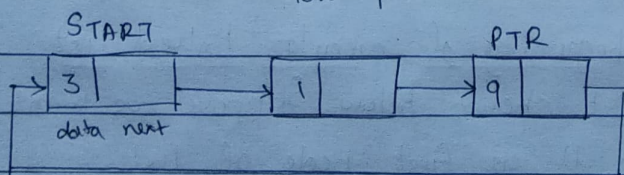
## b) Insertion of Node at end of the circular linked list.

Consider the linked list shown below. Suppose we want to add a new node with data 11 at the last node of the list



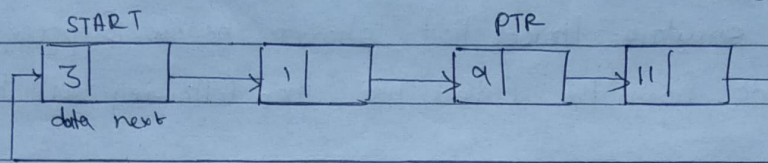
Allocate memory for new node and set its data part to 11.

Take a pointer variable PTR which will initially point to START. Move PTR so that it now point to last node of list





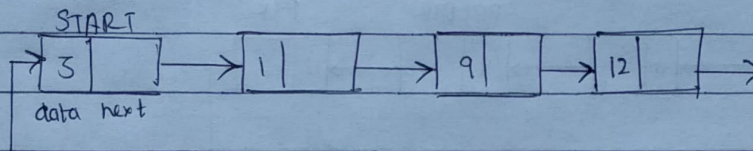
Add the new node after the node pointed by PTR



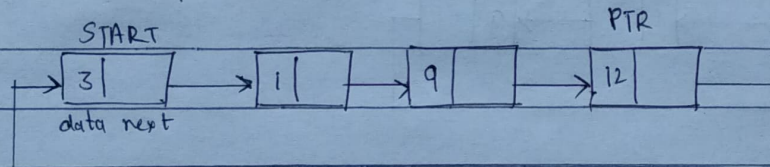
3) Deletion:

a) Deleting the first node from Circular linked list:

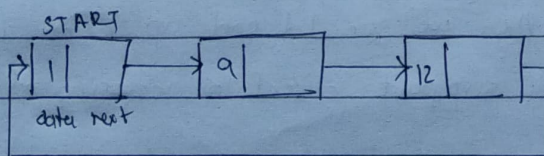
Consider the circular linked list shown below. Suppose we want to delete a node from beginning of the list.



Take a variable PTR and make it point to START. Move PTR further so that it points to last node of list



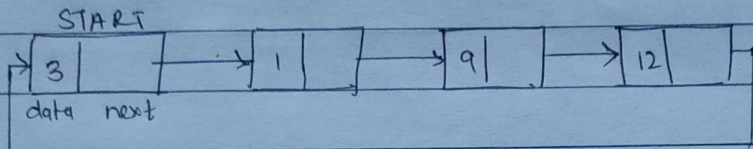
The next part of PTR is made to point to second node of list and memory of first node is freed. The second node becomes the START of the list



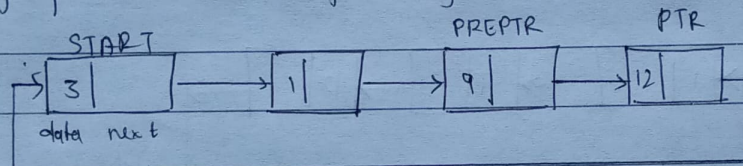


b) Deleting the last node from circular linked list

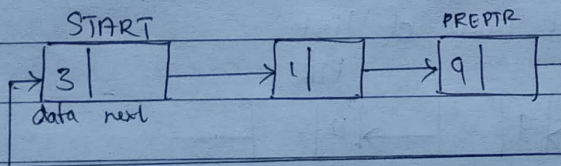
Consider the circular linked list shown below. Suppose we want to delete last node from the linked list, then following will be done:



Take 2 pointer's PREPTR and PTR will initially point to START. Move PTR so that it points to the last node of list. PREPTR will always point to node preceding PTR.



Make PREPTR's next part START and free PTR. Now PREPTR is last node of list.



• Limitations of Circular Linked List:

- They are complex as compared to linked list.
- Reversing of the list is complex as compared to single linked list.
- If not traversed carefully, then we would end up in an infinite loop.
- Circular linked list doesn't support direct accessing of elements.



## CONCLUSION:

### Errors encountered:

- 1) variable 'choice' declared inside switch, variable not defined.

Solution Declare the variable outside the switch() and take input from user.

- 2) Using assignment operator '=' instead of '==' in if statement.

Solution Using the relation operator '==' solves the error.

```

1 //SHREYAS SAWANT D7A 55
2 //Implement Circular Linked List using ADT
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 struct node
7 {
8     int data;
9     struct Node *next;
10 };
11 struct node *head,*ptr,*temp;
12
13 void beginsert()
14 {
15     struct node *ptr,*temp;
16     int item;
17     ptr = (struct node *)malloc(sizeof(struct node));
18     if(ptr == NULL)
19     {
20         printf("\nOVERFLOW\n");
21     }
22     else
23     {
24         printf("\nEnter the node data\n");
25         scanf("%d",&item);
26         ptr->data = item;
27         if(head == NULL)
28         {
29             head = ptr;
30             ptr->next = head;
31         }
32         else
33         {
34             temp = head;
35             while(temp->next != head)
36                 temp = temp->next;
37             ptr->next = head;
38             temp->next = ptr;
39             head = ptr;
40         }
41         printf("\nNode inserted\n");
42     }
43 }
44
45 void lastinsert()
46 {
47     struct node *ptr,*temp;
48     int item;
49     ptr = (struct node *)malloc(sizeof(struct node));
50     if(ptr == NULL)
51     {
52         printf("\nOVERFLOW\n");
53     }
54     else
55     {
56         printf("\nEnter Data\n");
57         scanf("%d",&item);
58         ptr->data = item;
59         if(head == NULL)
60         {
61             head = ptr;
62             ptr->next = head;
63         }
64         else
65         {
66             temp = head;
67             while(temp->next != head)
68             {
69                 temp = temp->next;
70             }
71             temp->next = ptr;
72             ptr->next = head;
73         }
74         printf("\nNode inserted\n");
75     }
76 }
77
78
79 void insAfter(int a,int b)
80 {
81     int k=0;
82     ptr =(struct Node *)malloc(sizeof(struct Node *));
83     temp =head;
84     if(temp==NULL)
85     {

```

```

85     printf("\nEMPTY\n");return;
86 }
87 while(temp->next!=NULL)
88 {
89     if(temp->data==b)
90     {
91         k=1;break;
92     }
93     temp=temp->next;
94 }
95
96 if(k)
97 {
98     ptr->next=temp->next;
99     ptr->data=a;
100    temp->next=ptr;
101    printf("\nNode inserted\n");
102 }
103 else
104     printf("\nNOT FOUND\n");
105 }
106
107 void begin_delete()
108 {
109     struct node *ptr;
110     if(head == NULL)
111     {
112         printf("\nUNDERFLOW\n");
113     }
114     else if(head->next == head)
115     {
116         head = NULL;
117         free(head);
118         printf("\nNode deleted\n");
119     }
120
121     else
122     {
123         ptr = head;
124         while(ptr -> next != head)
125             ptr = ptr -> next;
126         ptr->next = head->next;
127         free(head);
128         head = ptr->next;
129         printf("\nNode deleted\n");
130     }
131 }
132 void last_delete()
133 {
134     struct node *ptr, *preptr;
135     if(head==NULL)
136     {
137         printf("\nUNDERFLOW\n");
138     }
139     else if (head ->next == head)
140     {
141         head = NULL;
142         free(head);
143         printf("\nNode deleted\n");
144     }
145
146     else
147     {
148         ptr = head;
149         while(ptr ->next != head)
150         {
151             preptr=ptr;
152             ptr = ptr->next;
153         }
154         preptr->next = ptr -> next;
155         free(ptr);
156         printf("\nNode deleted\n");
157     }
158 }
159 }
160
161 void search()
162 {
163     struct node *ptr;
164     int item,i=0,flag=1;
165     ptr = head;temp=ptr->next;
166     if(ptr == NULL)
167     {
168         printf("\nEmpty List\n");

```

```

169     }
170     else
171     {
172         printf("\nEnter item which you want to search\n");
173         scanf("%d",&item);
174         if(head->data == item)
175         {
176             printf("\nItem found at location %d\n",i+1);
177             flag=0;
178         }
179         else
180         {
181             while (ptr->next != head)
182             {
183                 if(ptr->data == item)
184                 {
185                     printf("\nItem found at location %d\n ",i+1);
186                     flag=0;
187                     break;
188                 }
189                 else
190                 {
191                     flag=1;
192                 }
193                 i++;
194                 ptr = ptr -> next;
195             }
196         }
197     }
198     if(ptr->data==item)
199     {
200         printf("\nItem found at location %d\n ",i+1);flag=0;
201     }
202     if(flag != 0)
203     {
204         printf("Item not found\n");
205     }
206 }
207
208 }
209
210 void display()
211 {
212     struct node *ptr;
213     ptr=head;
214     if(head == NULL)
215     {
216         printf("\nEMPTY");return;
217     }
218     else
219     {
220         printf("\nElements of list \n");
221
222         while(ptr->next!= head)
223         {
224
225             printf("%d ", ptr -> data);
226             ptr = ptr -> next;
227         }
228         printf("%d ", ptr -> data);
229     }printf("\n");
230 }
231
232
233
234 void main ()
235 {
236     int choice =0,item;
237     while(choice != 7)
238     {
239
240         printf("\n1.Insert in beginning or Create List\n2.Insert at last\n3.Insert after a
Node\n4.Delete from Beginning\n5.Delete from last\n6.Search for an element\n7.Show\n8.Exit\n");
241         printf("\nEnter your choice?\n");
242         scanf("%d",&choice);
243         switch(choice)
244         {
245             case 1:
246                 begininsert();
247                 break;
248             case 2:
249                 lastinsert();
250                 break;
251             case 3:

```



```

252     {
253
254     int n;
255     printf("\nEnter the item which you want to insert?\n");
256     scanf("%d",&item);
257     printf("\nEnter the Node after which it is to be inserted\n");
258     scanf("%d",&n);
259     insAfter(item,n);break;
260     }
261     case 4:
262     begin_delete();
263     break;
264     case 5:
265     last_delete();
266     break;
267     case 6:
268     search();
269     break;
270     case 7:
271     {display();
272     break;}
273     case 8:
274     exit(0);
275     break;
276     default:
277     printf("Please enter valid choice..");
278     }
279     }
280 }
281

```

"C:\Users\user\Desktop\SHREYAS\SEM II\Circular.exe"

```
1.Insert in beginning or Create List
2.Insert at last
3.Insert after a Node
4.Delete from Beginning
5.Delete from last
6.Search for an element
7.Show
8.Exit
```

Enter your choice?

4

UNDERFLOW

```
1.Insert in beginning or Create List
2.Insert at last
3.Insert after a Node
4.Delete from Beginning
5.Delete from last
6.Search for an element
7.Show
8.Exit
```

Enter your choice?

1

Enter the node data

52

Node inserted

```
1.Insert in beginning or Create List
2.Insert at last
3.Insert after a Node
4.Delete from Beginning
5.Delete from last
6.Search for an element
7.Show
8.Exit
```

Enter your choice?

2



"C:\Users\user\Desktop\SHREYAS\SEM II\Circular.exe"

2

Enter Data

65

Node inserted

1.Insert in beginning or Create List

2.Insert at last

3.Insert after a Node

4.Delete from Beginning

5.Delete from last

6.Search for an element

7.Show

8.Exit

Enter your choice?

3

Enter the item which you want to insert?

78

Enter the Node after which it is to be inserted

52

Node inserted

1.Insert in beginning or Create List

2.Insert at last

3.Insert after a Node

4.Delete from Beginning

5.Delete from last

6.Search for an element

7.Show

8.Exit

Enter your choice?

6

Enter item which you want to search

65

Item found at location 3

"C:\Users\user\Desktop\SHREYAS\SEM II\Circular.exe"

Enter item which you want to search

65

Item found at location 3

1.Insert in beginning or Create List

2.Insert at last

3.Insert after a Node

4.Delete from Beginning

5.Delete from last

6.Search for an element

7.Show

8.Exit

Enter your choice?

7

Elements of list

52 78 65

Process returned 7 (0x7) execution time : 45.330 s

Press any key to continue.