

Single Linked Lists And Circular Linked Lists

1. Create a single linked list and insert a node at the beginning, end and any position in between.

Source Code

```
1 //  
2 // main.c  
3 // Lab-2-1  
4 //  
5 // Created by Aaron R on 17/08/21.  
6 // Create a single linked list and insert a node at the beginning, end and any position in between.  
7  
8 #include <stdio.h>  
9 #include <stdlib.h>  
10  
11 struct node  
12 {  
13     int data;  
14     struct node * link;  
15 };  
16  
17 void display (struct node *temp)  
18 {  
19     // Function to display the data in the linked list  
20  
21     // If head is NULL, the linked list is empty  
22     if (temp==NULL)  
23         printf("The linked list is empty\n");  
24     else  
25         // Else, display it's elements  
26         while ((temp)!=NULL)  
27         {  
28             printf("%d\n", (temp)->data);  
29             temp = (temp)->link;  
30         }  
31 }  
32  
33 void insert_head(struct node ** current, struct node ** head,int * count)  
34 {  
35     // Function to insert node at the head  
36     *current = (struct node *) malloc(sizeof(struct node));  
37  
38     // Input the node's data  
39     printf("Enter the data: ");  
40     scanf("%d",&(*current)->data);  
41  
42     // The address part of the new node points to the previous head  
43     (*current)->link = *head;  
44  
45     // The head variable is updated to the current one  
46     *head = *current;  
47  
48     *count+=1;  
49 }  
50
```

```
51 void insert_end(struct node ** current, struct node ** head,int * count)
52 {
53     // Function to insert node at the end
54     *current = (struct node *) malloc(sizeof(struct node));
55     // Input the node's data
56     printf("Enter the data: ");
57     scanf("%d",&(*current)->data);
58
59     // The link of the new node is NULL
60     (*current)->link = NULL;
61
62     // A temporary node is created
63     struct node * temp;
64     temp = *head;
65
66     // It is used to traverse to the end of the existing linked list
67     while (temp->link!=NULL)
68     {
69         temp = temp->link;
70     }
71
72     // The link of the node at the end of the existing linked list is updated to the new node
73     temp->link = *current;
74
75     *count+=1;
76 }
77
78 void insert_position(struct node ** current, struct node ** head,int * count,int p)
79 {
80     // Function to insert node at the head
81     *current = (struct node *) malloc(sizeof(struct node));
82
83     // Input the node's data
84     printf("Enter the data: ");
85     scanf("%d",&(*current)->data);
86
87     // A temporary node is created
88     struct node * temp;
89
90     // The temporary node is used to traverse and get to the pth position
91     temp = *head;
92     int i=1;
93     while (i<p)
94     {
95         temp = temp->link;
96         i+=1;
97     }
98
99     // The link of the new node is updated to the link of the node at (p+1)th position
100    (*current)->link = temp->link;
101
102    // The link of the node at the pth position is updated to the new node
103    temp->link = (*current);
104
105    *count+=1;
106 }
```

```
107
108 int main(int argc, const char * argv[])
109 {
110     // Initialising the linked list
111     printf("Initialising a linked list...\n");
112     struct node *head, *current;
113     int count=0;
114     head = NULL;
115     current = NULL;
116
117     // Displaying the initial elements of the linked list
118     printf("Elements of the linked list:\n");
119     display(head);
120
121     int c=1;
122     while (c==1)
123     {
124         int p;
125         printf("\n~##### Insertion of data into linked list #####~\n");
126         printf("1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are
127             currently %d nodes in the linked list\nEnter the position: ",count,count);
128         scanf("%d",&p);
129
130         if (p==0)
131             insert_head(&current,&head,&count);
132
133         else if (p==count||p==-1)
134             insert_end(&current,&head,&count);
135
136         else if((p>0)&&(p<=count))
137             insert_position(&current, &head, &count,p);
138
139         else
140             printf("Invalid position\n");
141
142         printf("Elements of the linked list:\n");
143         display(head);
144
145         printf("To insert another node, press 1: ");
146         scanf("%d",&c);
147     }
148     printf("\n---#####---\n                                ~End.\n\n");
149
150 }
```

Output(s):

```
Initialising a linked list...
Elements of the linked list:
The linked list is empty

~##### Insertion of data into linked list #####
1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position: 0
Enter the data: 10
Elements of the linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into linked list #####
1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position: 1
Enter the data: 20
Elements of the linked list:
10
20
To insert another node, press 1: 1

~##### Insertion of data into linked list #####
1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position: -1
Enter the data: 30
Elements of the linked list:
10
20
30
To insert another node, press 1: 1

~##### Insertion of data into linked list #####
1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position: 2
Enter the data: 40
Elements of the linked list:
10
20
40
30
To insert another node, press 1: 1

~##### Insertion of data into linked list #####
1) 0 to insert a new node at the head
2) 4 or -1 to insert a new node at the end
There are currently 4 nodes in the linked list
Enter the position: 5
Invalid position
Elements of the linked list:
10
20
40
30
To insert another node, press 1: 1
```

```
~##### Insertion of data into linked list #####~  
1) 0 to insert a new node at the head  
2) 4 or -1 to insert a new node at the end  
There are currently 4 nodes in the linked list  
Enter the position: 1  
Enter the data: 50  
Elements of the linked list:  
10  
50  
20  
40  
30  
To insert another node, press 1: 0  
----#####----  
~End.
```

Program ended with exit code: 0|

All Output ◊

Filter



2. Create a single linked list and delete a node at the beginning, end and any position in between.

Source Code

```
1 //  
2 // main.c  
3 // Lab-2-2  
4 //  
5 // Created by Aaron R on 18/08/21.  
6 // Create a single linked list and delete a node at the beginning, end and any position in between.  
7  
8 #include <stdio.h>  
9 #include <stdlib.h>  
10  
11 struct node  
12 {  
13     int data;  
14     struct node * link;  
15 };  
16  
17 void display (struct node *temp)  
18 {  
19     // Function to display the data in the linked list  
20  
21     // If head is NULL, the linked list is empty  
22     if (temp==NULL)  
23         printf("The single linked list is empty\n");  
24     else  
25         // Else, display it's elements  
26         while ((temp)!=NULL)  
27         {  
28             printf("%d\n", (temp)->data);  
29             temp = (temp)->link;  
30         }  
31 }  
32  
33 // ##### Functions to insert nodes #####  
34  
35 void insert_head(struct node ** current, struct node ** head,int * count)  
36 {  
37     // ##### Function to insert node at the head #####  
38     *current = (struct node *) malloc(sizeof(struct node));  
39  
40     // Input the node's data  
41     printf("Enter the data: ");  
42     scanf("%d",&(*current)->data);  
43  
44     // The address part of the new node points to the previous head  
45     (*current)->link = *head;  
46  
47     // The head variable is updated to the current one  
48     *head = *current;  
49  
50     *count+=1;  
51 }  
52
```



```
119 // ##### Functions to delete nodes #####
120 void delete_head(struct node **head, int * count)
121 {
122     // #### Function to delete the node at the beginning ####
123     struct node * temp;
124     temp=*head;
125     // The address of the head is updated to the next node
126     *head = (*head)->link;
127
128     // free the memory of the previous tail
129     free(temp);
130     *count-=1;
131 }
132
133 void delete_end(struct node **head, int * count)
134 {
135     // #### If there is only one node, delete head #####
136     if ((*head)->link==NULL)
137         delete_head(head, count);
138
139     // if there is more than one node
140     else
141     {
142         struct node * temp;
143         struct node * prev_temp;
144         temp = *head;
145         prev_temp=NULL;
146         // Traverse through the single linked list and store the last node in temp
147         // and the node preceding it in prev_temp
148         while (temp->link!=NULL)
149         {
150             prev_temp = temp;
151             temp = temp->link;
152         }
153
154         // Update the link of the node preceding the previous tail to NULL
155         prev_temp->link = NULL;
156
157         // Free the memory of the previous tail
158         free(temp);
159         *count-=1;
160     }
161 }
162 void delete_pos(struct node **head, int p, int * count)
163 {
164     // ##### Function to delete node at pth position #####
165
166     struct node * temp;
167     struct node * next_temp;
168     temp = *head;
169
170     // Traverse till the p-1th element and store it in temp
171     int i=1;
172     while (i<p-1)
173     {
174         temp = temp->link;
175         i+=1;
176     }
177     // The pth node is stored in next_temp
178     next_temp = temp->link;
179
180     // The link of the p-1th node is updated to the p+1th node
181     temp->link = next_temp->link;
182
183     // free the memory of the previous tail
184     free(next_temp);
185     *count-=1;
186
187 }
```

```
188 int main(int argc, const char * argv[])
189 {
190     // Initialising the linked list
191     printf("Initialising a single linked list...\n");
192     struct node *head, *current;
193     int count=0;
194     head = NULL;
195     current = NULL;
196
197     // Displaying the initial elements of the linked list
198     printf("Elements of the single linked list:\n");
199     display(head);
200
201     int c=1;
202     while (c==1)
203     {
204         int p;
205         printf("\n~##### Insertion of data into single linked list #####~\n");
206         printf("\n1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are
207             currently %d nodes in the linked list\nEnter the position: ",count,count);
208         scanf("%d",&p);
209
210         if (p==0)
211             insert_head(&current,&head,&count);
212
213         else if (p==count||p==-1)
214             insert_end(&current,&head,&count);
215
216         else if((p>=0)&&(p<=count))
217             insert_position(&current, &head, &count,p);
218
219         else
220             printf("Invalid position\n");
221
222         printf("Elements of the linked list:\n");
223         display(head);
224
225         printf("To insert another node, press 1: ");
226         scanf("%d",&c);
227     }
228
229     c=1;
230     while (c==1)
231     {
232         if (head==NULL)
233         {
234             printf("The linked list is empty. There are no more nodes to delete.\n");
235             break;
236         }
237         else
238         {
239             int p;
240             printf("\n~##### Deletion of data in linked list #####~\n");
241             printf("\n1) 0 to delete a new node at the head\n2) %d or -1 to delete a new node at the end\nThere
242                 are currently %d nodes in the linked list\nEnter the position in which to delete the new node:
243                 ",count,count);
244             scanf("%d",&p);
245             if (p==0)
246                 delete_head(&head,&count);
247
248             else if ((p==count)|| (p==-1))
249                 delete_end(&head,&count);
250
251             else if((p>=0)&&(p<=count))
252                 delete_pos(&head, p,&count);
253
254             else
255                 printf("Invalid position");
```

```
255     printf("Elements of the circular linked list:\n");
256     display(head);
257     printf("To delete another node, press 1: ");
258     scanf("%d",&c);
259
260 }
261 }
262 }
263
264 printf("\n---#####\n"); ~End.\n\n");
265 return 0;
266 }
```

Outputs:

```
Initialising a single linked list...
Elements of the single linked list:
The single linked list is empty

~##### Insertion of data into single linked list ####~

1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position: -1
Enter the data: 10
Elements of the linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into single linked list ####~

1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position: -1
Enter the data: 20
Elements of the linked list:
10
20
To insert another node, press 1: 1

~##### Insertion of data into single linked list ####~

1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position: -1
Enter the data: 30
Elements of the linked list:
10
20
30
To insert another node, press 1: 1
```

```
~##### Insertion of data into single linked list #####~
```

```
1) 0 to insert a new node at the head  
2) 3 or -1 to insert a new node at the end  
There are currently 3 nodes in the linked list  
Enter the position: -1  
Enter the data: 40  
Elements of the linked list:  
10  
20  
30  
40  
To insert another node, press 1: 1
```

```
~##### Insertion of data into single linked list #####~
```

```
1) 0 to insert a new node at the head  
2) 4 or -1 to insert a new node at the end  
There are currently 4 nodes in the linked list  
Enter the position: -1  
Enter the data: 50  
Elements of the linked list:  
10  
20  
30  
40  
50  
To insert another node, press 1: 2
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 5 or -1 to delete a new node at the end  
There are currently 5 nodes in the linked list  
Enter the position in which to delete the new node: 0  
Elements of the circular linked list:  
20  
30  
40  
50  
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 4 or -1 to delete a new node at the end  
There are currently 4 nodes in the linked list  
Enter the position in which to delete the new node: 2  
Elements of the circular linked list:  
20  
40  
50  
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 3 or -1 to delete a new node at the end  
There are currently 3 nodes in the linked list  
Enter the position in which to delete the new node: -1  
Elements of the circular linked list:  
20  
40  
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 2 or -1 to delete a new node at the end  
There are currently 2 nodes in the linked list  
Enter the position in which to delete the new node: 2  
Elements of the circular linked list:  
20  
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 1 or -1 to delete a new node at the end  
There are currently 1 nodes in the linked list  
Enter the position in which to delete the new node: 5  
Invalid position  
Elements of the circular linked list:  
20  
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~
```

```
1) 0 to delete a new node at the head  
2) 1 or -1 to delete a new node at the end  
There are currently 1 nodes in the linked list  
Enter the position in which to delete the new node: 0  
Elements of the circular linked list:  
The single linked list is empty  
To delete another node, press 1: 1  
The linked list is empty. There are no more nodes to delete.
```

```
----#####----
```

```
~End.
```

```
Program ended with exit code: 0
```

```
All Output ▾
```

```
Filter
```

```
✖ | ⌂ ⌂
```

3. Create a circular single linked list and insert a node at the beginning, end and any position in between.

Source code

```
1 //  
2 //  main.c  
3 //  Lab-2-3  
4 //  
5 //  Created by Aaron R on 18/08/21.  
6 //  Create a circular single linked list and insert a node at the beginning, end and any position in between  
7  
8 #include <stdio.h>  
9 #include <stdlib.h>  
10  
11 struct node  
12 {  
13     int data;  
14     struct node * link;  
15 }*tail;  
16  
17 void display(void)  
18 {  
19     // ##### To display the elements of the circular linked list #####  
20     struct node * temp;  
21     temp = tail->link;  
22     if (tail==NULL)  
23     {  
24         printf("The circular linked list is empty");  
25     }  
26     else  
27     {  
28         // traversing from the first node  
29         while (temp->link!=tail->link)  
30         {  
31             printf("%d\n",temp->data);  
32             temp = temp->link;  
33         }  
34         // printing the last node's data  
35         printf("%d\n",temp->data);  
36     }  
37 }  
38 void insert_head(int * count)  
39 {  
40     // ##### Function to insert node at the head #####  
41     struct node * current;  
42     current = (struct node *) malloc(sizeof(struct node));  
43     // Inputting data to the node  
44     printf("Enter data: ");  
45     scanf("%d",&current->data);  
46     current->link=NULL;  
47  
48     // If the circular linked list is empty, assign the new node as the tail  
49     if (tail == NULL)  
50     {  
51         tail = current;  
52         tail->link = current;  
53     }  
54     // If the circular linked list isn't empty  
55     else  
56     {  
57         // The new node is linked to the head  
58         current->link = tail->link;  
59  
60         // The tail is linked to the new node  
61         tail->link = current;  
62     }  
63  
64     *count +=1;  
65 }
```

```
67 void insert_end(int * count)
68 {
69     // ##### Function to insert node at the end #####
70     struct node * current;
71     current = (struct node *) malloc(sizeof(struct node));
72     // Inputting data to the node
73     printf("Enter the data: ");
74     scanf("%d",&current->data);
75     current->link = NULL;
76
77     // If the circular linked list is empty, assign the new node as the tail
78     if (tail==NULL)
79     {
80         tail = current;
81         tail->link = current;
82     }
83
84     // If the circular linked list isn't empty
85     else
86     {
87         // The new node's link is assigned the address of the head
88         current->link = tail->link;
89
90         // The previous tail's link is assigned the address of the new node
91         tail->link = current;
92
93         // The tail variable is updated to the new node
94         tail = current;
95     }
96     *count+=1;
97 }
98
99 void insert_pos(int * count, int p)
100 {
101     // ##### Function to insert node in position p #####
102     struct node * current;
103     struct node * temp;
104     current = (struct node *) malloc(sizeof(struct node));
105
106     // Inputting data to the node
107     printf("Enter data: ");
108     scanf("%d",&current->data);
109     current->link=NULL;
110
111     // The temp variable is assigned the address of the head
112     temp = tail->link;
113     int i=0;
114
115     // The temp variable traverses untill it reaches the p-1th position
116     while (i<p-1)
117     {
118         temp = temp->link;
119         i+=1;
120     }
121     // The new node's link is given the address of the p+1th element
122     current->link = temp->link;
123
124     // The p-1th element's link is given to be the address of the new node
125     temp->link = current;
126
127     *count+=1;
128 }
```

```
130 int main(int argc, const char * argv[])
131 {
132
133     int count=0;
134     int c=1;
135     while (c==1)
136     {
137         int p;
138         printf("\n~##### Insertion of data into circular linked list #####~\n");
139         printf("\n1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are
140             currently %d nodes in the linked list\nEnter the position in which to insert the new node:
141             ",count,count);
142         scanf("%d",&p);
143         if (p==0)
144         {
145             insert_head(&count);
146         }
147         else if (p==count||p== -1)
148         {
149             insert_end(&count);
150         }
151         else if((p>=0)&&(p<=count))
152         {
153             insert_pos(&count,p);
154         }
155         else
156             printf("Invalid position\n");
157
158         printf("Elements of the circular linked list:\n");
159         display();
160
161         printf("To insert another node, press 1: ");
162         scanf("%d",&c);
163     }
164     printf("\n---#####---\n~End.\n\n");
165     return 0;
166 }
```

Outputs:

```
~##### Insertion of data into circular linked list #####~  
1) 0 to insert a new node at the head  
2) 0 or -1 to insert a new node at the end  
There are currently 0 nodes in the linked list  
Enter the position in which to insert the new node: 0  
Enter data: 10  
Elements of the circular linked list:  
10  
To insert another node, press 1: 1  
  
~##### Insertion of data into circular linked list #####~  
1) 0 to insert a new node at the head  
2) 1 or -1 to insert a new node at the end  
There are currently 1 nodes in the linked list  
Enter the position in which to insert the new node: 1  
Enter the data: 20  
Elements of the circular linked list:  
10  
20  
To insert another node, press 1: 1  
  
~##### Insertion of data into circular linked list #####~  
1) 0 to insert a new node at the head  
2) 2 or -1 to insert a new node at the end  
There are currently 2 nodes in the linked list  
Enter the position in which to insert the new node: -1  
Enter the data: 30  
Elements of the circular linked list:  
10  
20  
30  
To insert another node, press 1: 1  
  
~##### Insertion of data into circular linked list #####~  
1) 0 to insert a new node at the head  
2) 3 or -1 to insert a new node at the end  
There are currently 3 nodes in the linked list  
Enter the position in which to insert the new node: 2  
Enter data: 40  
Elements of the circular linked list:  
10  
20  
40  
30  
To insert another node, press 1: 1  
  
~##### Insertion of data into circular linked list #####~  
1) 0 to insert a new node at the head  
2) 4 or -1 to insert a new node at the end  
There are currently 4 nodes in the linked list  
Enter the position in which to insert the new node: 6  
Invalid position  
Elements of the circular linked list:  
10  
20  
40  
30  
To insert another node, press 1: 0  
  
----#####----  
~End.
```

Program ended with exit code: 0

All Output 

 Filter

  

4. Create a circular single linked list and delete a node at the beginning, end and any position in between.

Source Code

```
1 //  
2 // main.c  
3 // Lab-2-4  
4 //  
5 // Created by Aaron R on 19/08/21.  
6 // Create a circular single linked list  
7 // and delete a node at the beginning, end and any position in between.  
8  
9 #include <stdio.h>  
10 #include <stdlib.h>  
11  
12 struct node  
13 {  
14     int data;  
15     struct node * link;  
16 }*tail;  
17  
18 void display(void)  
19 {  
20     // ##### To display the elements of the circular linked list #####  
21     struct node * temp;  
22  
23     if (tail==NULL)  
24     {  
25         printf("The circular linked list is empty\n");  
26     }  
27     else  
28     {  
29         temp = tail->link;  
30         // traversing from the first node  
31         while (temp->link!=tail->link)  
32         {  
33             printf("%d\n",temp->data);  
34             temp = temp->link;  
35         }  
36         // printing the last node's data  
37         printf("%d\n",temp->data);  
38     }  
39 }  
40  
41 // ##### Functions to insert nodes #####  
42 void insert_head(int * count)  
43 {  
44     // ##### Function to insert node at the head #####  
45     struct node * current;  
46     current = (struct node *) malloc(sizeof(struct node));  
47     // Inputting data to the node  
48     printf("Enter data: ");  
49     scanf("%d",&current->data);  
50     current->link=NULL;  
51  
52     // If the circular linked list is empty, assign the new node as the tail  
53     if (tail == NULL)  
54     {  
55         tail = current;  
56         tail->link = current;  
57     }  
58     // If the circular linked list isn't empty  
59     else  
60     {  
61         // The new node is linked to the head  
62         current->link = tail->link;  
63  
64         // The tail is linked to the new node  
65         tail->link = current;  
66     }  
67  
68     *count +=1;  
69 }
```

```
70
71 void insert_end(int * count)
72 {
73     // ##### Function to insert node at the end #####
74     struct node * current;
75     current = (struct node *) malloc(sizeof(struct node));
76     // Inputting data to the node
77     printf("Enter the data: ");
78     scanf("%d",&current->data);
79     current->link = NULL;
80
81     // If the circular linked list is empty, assign the new node as the tail
82     if (tail==NULL)
83     {
84         tail = current;
85         tail->link = current;
86     }
87
88     // If the circular linked list isn't empty
89     else
90     {
91         // The new node's link is assigned the address of the head
92         current->link = tail->link;
93
94         // The previous tail's link is assigned the address of the new node
95         tail->link = current;
96
97         // The tail variable is updated to the new node
98         tail = current;
99     }
100    *count+=1;
101 }
102
103
104 void insert_pos(int * count, int p)
105 {
106     // ##### Function to insert node in position p #####
107     struct node * current;
108     struct node * temp;
109     current = (struct node *) malloc(sizeof(struct node));
110
111     // Inputting data to the node
112     printf("Enter data: ");
113     scanf("%d",&current->data);
114     current->link=NULL;
115
116     // The temp variable is assigned the address of the head
117     temp = tail->link;
118     int i=1;
119
120     // The temp variable traverses untill it reaches the p-1th position
121     while (i<p-1)
122     {
123         temp = temp->link;
124         i+=1;
125     }
126     // The new node's link is given the address of the p+1th element
127     current->link = temp->link;
128
129     // The p-1th element's link is given to be the address of the new node
130     temp->link = current;
131
132     *count+=1;
133 }
```

```
134 // ##### Functions to delete nodes #####
135 void delete_head(int * count)
136 {
137     // #### Function to delete a node at the beginning ####
138
139     struct node * temp;
140     // Initialise the temp variable with the head of the circular linked list
141     temp = tail->link;
142
143     // If the tail is empty, display that the circular linked list is empty
144     if (tail==0)
145         printf("The circular linked list is empty");
146
147     // If the tail has only one node, delete the node
148     else if (tail->link==tail)
149         tail = NULL;
150
151
152     // If the tail has more than one node,
153     // update the link of the tail node to the 2nd node of circular linked list
154     else
155     {
156         tail->link = temp->link;
157     }
158
159
160     *count-=1;
161 }
162
163
164 void delete_end(int * count)
165 {
166     // #### Function to delete a node at the end ####
167
168     struct node * current;
169     struct node * prev;
170     prev=NULL;
171     // Initialise the current variable with the head of the circular linked list
172     current = tail->link;
173
174     // If the tail is empty, display that the circular linked list is empty
175     if (tail==NULL)
176         printf("The circular linked list is empty\n");
177
178     // If the tail has only one node, delete the node
179     else if (tail->link==tail)
180         tail = NULL;
181
182     // If the tail has more than one node,
183     else
184     {
185         // Traverse the list using the current variable till the tail is reached
186         while (current->link!=tail->link)
187         {
188             // The element that precedes the tail is stored in the variable prev
189             prev = current;
190             current = current->link;
191         }
192         // The link of the node preceding the tail is updated to the address of the head
193         prev->link = tail->link;
194
195         // The tail variable is updated with the element that preceded the previous tail
196         tail = prev;
197     }
198     *count-=1;
199     // free the memory of the node that was deleted
200     free(current);
201 }
```

```

202
203 void delete_pos(int * count, int p)
204 {
205     // ##### Function to delete a node at the pth position #####
206
207     struct node * current;
208     current = tail->link;
209     struct node * next_current;
210     next_current = NULL;
211     int i=1;
212
213     // Traverse till the p-1th element is reached and stored in the current variable
214     while (i<p-1)
215     {
216         current = current->link;
217         i++;
218     }
219     // The pth element is stored in the next_current variable
220     next_current = current->link;
221
222     // The link of the p-1th element is updated to the address of the p+1th element
223     current->link = next_current->link;
224
225     // free the memory of the node that was deleted
226     free(current);
227     *count-=1;
228 }
229
230 int main(int argc, const char * argv[])
231 {
232
233     int count=0;
234
235     // ##### Menu to insert nodes #####
236     int c=1;
237     while (c==1)
238     {
239         int p;
240         printf("\n~##### Insertion of data into linked list #####~\n");
241         printf("Enter the position from which to delete the node\n1) 0 to insert a new node at the head\n2) -1 to insert a new node at the end\nThere are currently %d nodes in the linked list\nEnter the position to insert: ",count,count);
242         scanf("%d",&p);
243         if (p==0)
244         {
245             insert_head(&count);
246         }
247         else if (p==count||p==-1)
248         {
249             insert_end(&count);
250         }
251         else if((p>0)&&(p<count))
252         {
253             insert_pos(&count,p);
254         }
255
256         printf("Elements of the circular linked list:\n");
257         display();
258
259         printf("To insert another node, press 1: ");
260         scanf("%d",&c);
261     }
262
263     // ##### Menu to delete nodes #####
264     c=1;
265     while (c==1)
266     {
267         if (tail==NULL)
268         {
269             printf("There are no more nodes left to delete.\n\n");
270             break;
271         }

```

```
272
273     else
274     {
275         int p;
276         printf("\n~##### Deletion of data from circular linked list #####-\n");
277         printf("Enter the position in which to insert the new node\n1) 0 to delete a new node at the head\n2)
278             %d or -1 to delete a new node at the end\nThere are currently %d nodes in the linked list\nEnter
279             the position to delete: ",count,count);
280         scanf("%d",&p);
281         if (p==0)
282         {
283             delete_head(&count);
284         }
285         else if (p==count||p== -1)
286         {
287             delete_end(&count);
288         }
289         else if((p>0)&&(p<count))
290         {
291             delete_pos(&count, p);
292         }
293         else
294             printf("Invalid position\n");
295         printf("Elements of the circular linked list:\n");
296         display();
297
298         printf("To delete another node, press 1: ");
299         scanf("%d",&c);
300     }
301 }
302 return 0;
303 }
```

Outputs

```
##### Insertion of data into linked list #####
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position to insert: 0
Enter data: 10
Elements of the circular linked list:
10
To insert another node, press 1: 1

##### Insertion of data into linked list #####
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position to insert: 0
Enter data: 20
Elements of the circular linked list:
20
10
To insert another node, press 1: 1

##### Insertion of data into linked list #####
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position to insert: 0
Enter data: 30
Elements of the circular linked list:
30
20
10
To insert another node, press 1: 1

##### Insertion of data into linked list #####
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position to insert: 0
Enter data: 40
Elements of the circular linked list:
40
30
20
10
To insert another node, press 1: 0
```

```
##### Deletion of data from circular linked list #####
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 4 or -1 to delete a new node at the end
There are currently 4 nodes in the linked list
Enter the position to delete: 2
Elements of the circular linked list:
40
20
10
To delete another node, press 1: 1

##### Deletion of data from circular linked list #####
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 3 or -1 to delete a new node at the end
There are currently 3 nodes in the linked list
Enter the position to delete: -1
Elements of the circular linked list:
40
20
To delete another node, press 1: 1

##### Deletion of data from circular linked list #####
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 2 or -1 to delete a new node at the end
There are currently 2 nodes in the linked list
Enter the position to delete: 0
Elements of the circular linked list:
20
To delete another node, press 1: 1

##### Deletion of data from circular linked list #####
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 1 or -1 to delete a new node at the end
There are currently 1 nodes in the linked list
Enter the position to delete: 1
Elements of the circular linked list:
The circular linked list is empty
To delete another node, press 1: 1
There are no more nodes left to delete.
```

Program ended with exit code: 0

All Output 

 Filter

