1. **//Source Code for Circular Single Linked List:**

```
2. include <stdio.h>
3. include <conio.h>
4. include <stdlib.h>
5. struct cslinklist
6. {
7. int data;
8. struct cslinklist *next;
9. };
10. typedef struct cslinklist node;
11. node *start = NULL;
12. int nodectr;
13. node* getnode()
14. {
15. node * newnode;
16. newnode = (node *) malloc(sizeof(node));
17. printf("\n Enter data: ");
18. scanf("%d", &newnode -> data);
19. newnode -> next = NULL; return
20. newnode;
21. }
22.
23. int menu()
24. {
25. int ch;
26. clrscr();
27. printf("\n 1. Create a list ");
28. printf("\n\n-------------------------");
29. printf("\n 2. Insert a node at beginning ");
30. printf("\n 3. Insert a node at end");
31. printf("\n 4. Insert a node at middle");
32. printf("\n\n-------------------------");
33. printf("\n 5. Delete a node from beginning");
34. printf("\n 6. Delete a node from Last");
35. printf("\n 7. Delete a node from Middle");
36. printf("\n\n-------------------------");
37. printf("\n 8. Display the list");
38. printf("\n 9. Exit");
39. printf("\n\n-------------------------");
40. printf("\n Enter your choice:
41. "); scanf("%d", &ch);
42. return ch;
43. }
44. void createlist(int n)
45. {
46. int i;
47. node *newnode;
48. node *temp;
49. nodectr = n;
50. for(i = 0; i < n ; i++)
51. {
52. newnode = getnode();
```

```
53. if(start == NULL)
54. {
55. start = newnode;
56. }
57. else
58. {
59. temp = start;
60. while(temp -> next != NULL)
61. temp = temp -> next;
62. }
63. temp -> next = newnode;
64. }
65. newnode ->next = start;
66. /* last node is pointing to starting node */
67. }
68. void display()
69. {
70. node *temp;
71. temp = start;
72. printf("\n The contents of List (Left to Right): ");
73. if(start == NULL )
74. printf("\n Empty List");
75. else
76. {
77. do
78. {
79. printf("\t %d ", temp -> data);
80. temp = temp -> next;
81. } while(temp !=
82. start); printf(" X ");
83. }
84. }
85.
86. void cll_insert_beg()
87. {
88. node *newnode, *last;
89. newnode = getnode();
90. if(start == NULL)
91. {
92. start = newnode; newnode
93. -> next = start;
94. }
95. else
96. {
97. last = start;
98. while(last -> next != start)
99. last = last -> next;
100.          newnode -> next =
101.          start; start = newnode;
102.          last -> next = start;
103.          }
104.          printf("\n Node inserted at beginning..");
```

```
105.        nodectr++;
106.        }
107.        void cll_insert_end()
108.        {
109.        node *newnode, *temp;
110.        newnode = getnode();
111.        if(start == NULL )
112.        {
113.        start = newnode; newnode
114.        -> next = start;
115.        }
116.        else
117.        {
118.        temp = start;
119.        while(temp -> next != start)
120.        temp = temp -> next;
121.        temp -> next = newnode;
122.        newnode -> next = start;
123.        }
124.        }
125.        printf("\n Node inserted at end..");
126.        nodectr++;
127.        void cll_insert_mid()
128.        {
129.        node *newnode, *temp, *prev;
130.        int i, pos ;
131.        newnode = getnode(); printf("\n
132.        Enter the position: ");
133.        scanf("%d", &pos);
134.        if(pos > 1 && pos < nodectr)
135.        {
136.        temp = start;
137.        prev = temp;
138.        i = 1;
139.        while(i < pos)
140.        {
141.        prev = temp;
142.        temp = temp ->
143.        next; i++;
144.        }
145.        prev -> next = newnode;
146.        newnode -> next = temp;
147.
148.
149.        nodectr++;
150.        printf("\n Node inserted at middle..");
151.        }
152.        else
153.        {
154.        printf("position %d of list is not a middle position ", pos);
155.        }
156.        }
```

```
157.    void cll_delete_beg()
158.    {
159.    node *temp, *last;
160.    if(start == NULL)
161.    {
162.    printf("\n No nodes
163.    exist.."); getch();
164.    return ;
165.    }
166.    else
167.    {
168.    last = temp = start;
169.    while(last -> next != start)
170.    last = last -> next;
171.    start = start -> next;
172.    last -> next = start;
173.    free(temp);
174.    nodectr--;
175.    printf("\n Node deleted..");
176.    if(nodectr == 0)
177.    }
178.    start = NULL;
179.    }
180.    void cll_delete_last()
181.    {
182.    node *temp,*prev;
183.    if(start == NULL)
184.    {
185.    printf("\n No nodes
186.    exist.."); getch();
187.    return ;
188.    }
189.    else
190.    {
191.    temp = start;
192.    prev = start;
193.    while(temp -> next != start)
194.    {
195.    prev = temp;
196.    temp = temp -> next;
197.    }
198.    prev -> next = start;
199.    free(temp); nodectr-
200.    -;
201.    if(nodectr == 0) start
202.    = NULL;
203.    }
204.    }
205.    printf("\n Node deleted
206.
207.
208.
```

```
209.    void cll_delete_mid()
210.    {
211.    int i = 0, pos;
212.    node *temp, *prev;
213.    if(start == NULL)
214.    {
215.    printf("\n No nodes
216.    exist.."); getch();
217.    return ;
218.    }
219.    else
220.    {
221.    printf("\n Which node to delete: ");
222.    scanf("%d", &pos);
223.    if(pos > nodectr)
224.    {
225.    printf("\nThis node does not
226.    exist"); getch();
227.    return;
228.    }
229.    if(pos > 1 && pos < nodectr)
230.    {
231.    temp=start;
232.    prev = start;
233.    i = 0;
234.    while(i < pos - 1)
235.    {
236.    prev = temp;
237.    temp = temp -> next ;
238.    i++;
239.    }
240.    prev -> next = temp -> next;
241.    free(temp);
242.    nodectr--;
243.    printf("\n Node Deleted..");
244.    }
245.    else
246.    {
247.    printf("\n It is not a middle position..");
248.    getch();
249.    }
250.    }
251.    }
252.    void main(void)
253.    {
254.    int result;
255.    int ch, n;
256.    clrscr();
257.    while(1)
258.    {
259.    ch = menu();
260.    switch(ch)
```

```
{
case 1 :
if(start == NULL)
{
printf("\n Enter Number of nodes to create: ");
scanf("%d", &n);
createlist(n);
printf("\nList created..");
}



else
case 2
case 3
case 4
case 5
printf("\n List is already Exist..");
break;
:
cll_insert_beg();
break;
:
cll_insert_end();
break;
:
cll_insert_mid();
break;
:
cll_delete_beg();
break;
case 6 : cll_delete_last();
break;
case 7 :
cll_delete_mid();
break;
case 8 :
display();
break;
case 9 :
exit(0);
}
getch();
}
}

```