

1. //A Complete Source Code for the Implementation of Double Linked List:

```
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <conio.h>
5. struct dlinklist
6. {
7.     struct dlinklist *left;
8.     int data;
9.     struct dlinklist *right;
10. };
11. typedef struct dlinklist node;
12. node *start = NULL;
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->left = NULL;
20.     newnode->right = NULL;
21.     return newnode;
22. }
23. int countnode(node *start)
24. {
25.     if(start == NULL)
26.         return 0;
27.     else
28.         return 1 + countnode(start->right);
29. }
30. int menu()
31. {
32.     int ch;
33.     clrscr();
34.     printf("\n 1.Create");
35.     printf("\n-----");
36.     printf("\n 2. Insert a node at beginning ");
37.     printf("\n 3. Insert a node at end");
38.     printf("\n 4. Insert a node at middle");
39.     printf("\n-----");
40.     printf("\n 5. Delete a node from beginning");
41.     printf("\n 6. Delete a node from Last");
42.     printf("\n 7. Delete a node from Middle");
43.     printf("\n-----");
44.     printf("\n 8. Traverse the list from Left to Right ");
45.     printf("\n 9. Traverse the list from Right to Left ");
46.     printf("\n-----");
47.     printf("\n 10.Count the Number of nodes in the list");
48.     printf("\n 11.Exit ");
49.     printf("\n\n Enter your choice: ");
50.     scanf("%d", &ch);
51.     return ch;
52. }
```

```

53. void createlist(int n)
54. {
55. int i;
56. node *newnode;
57. node *temp;
58. for(i = 0; i < n; i++)
59. {
60. newnode = getnode();
61. if(start == NULL)
62. start = newnode;
63. else
64. {
65. temp = start;
66. while(temp -> right)
67. temp = temp -> right;
68. temp -> right = newnode;
69. newnode -> left = temp;
70. }
71. }
72. }
73.
74. void traverse_left_to_right()
75. {
76. node *temp;
77. temp = start;
78. printf("\n The contents of List:
79. "); if(start == NULL )
80. printf("\n Empty List");
81. else
82. {
83. while(temp != NULL)
84. {
85. printf("\t %d ", temp -> data);
86. temp = temp -> right;
87. }
88. }
89. }
90. void traverse_right_to_left()
91. {
92. node *temp;
93. temp = start;
94. printf("\n The contents of List:
95. "); if(start == NULL)
96. printf("\n Empty List");
97. else
98. {
99. while(temp -> right != NULL)
100.     temp = temp -> right;
101.     }
102.     while(temp != NULL)
103.     {
104.         printf("\t%d", temp ->

```

```

105.     data); temp = temp -> left;
106.     }
107.     }
108.     void dll_insert_beg()
109.     {
110.         node *newnode;
111.         newnode = getnode();
112.         if(start == NULL)
113.             start = newnode;
114.         else
115.         {
116.             newnode -> right = start;
117.             start -> left = newnode;
118.             start = newnode;
119.         }
120.     }
121.     void dll_insert_end()
122.     {
123.         node *newnode, *temp;
124.         newnode = getnode();
125.         if(start == NULL)
126.             else
127.             {
128.             }
129.         }
130.         start = newnode;
131.         temp = start;
132.         while(temp -> right != NULL)
133.             temp = temp -> right;
134.             temp -> right = newnode;
135.             newnode -> left = temp;
136.
137.         void dll_insert_mid()
138.         {
139.             node *newnode,*temp;
140.             int pos, nodectr, ctr = 1;
141.             newnode = getnode();
142.             printf("\n Enter the position: ");
143.             scanf("%d", &pos);
144.             nodectr = countnode(start);
145.             if(pos - nodectr >= 2)
146.             {
147.                 printf("\n Position is out of range..");
148.                 return;
149.             }
150.             if(pos > 1 && pos < nodectr)
151.             {
152.                 temp = start;
153.                 while(ctr < pos - 1)
154.                 {
155.                     temp = temp -> right;
156.                     ctr++;

```

```

157.     }
158.     newnode -> left = temp; newnode
159.     -> right = temp -> right; temp ->
160.     right -> left = newnode; temp ->
161.     right = newnode;
162.     }
163.     else
164.     printf("position %d of list is not a middle position ", pos);
165.     }
166.     void dll_delete_beg()
167.     {
168.     node *temp;
169.     if(start == NULL)
170.     {
171.     printf("\n Empty
172.     list"); getch();
173.     return ;
174.     }
175.     else
176.     {
177.     temp = start;
178.     start = start -> right;
179.     start -> left = NULL;
180.     free(temp);
181.     }
182.     }
183.     void dll_delete_last()
184.     {
185.     node *temp;
186.     if(start == NULL)
187.     {
188.     printf("\n Empty
189.     list"); getch();
190.     return ;
191.     }
192.     else
193.     {
194.     temp = start;
195.     while(temp -> right != NULL)
196.
197.     temp = temp -> right;
198.     temp -> left -> right = NULL;
199.     free(temp);
200.     temp = NULL;
201.     }
202.     }
203.     void dll_delete_mid()
204.     {
205.     int i = 0, pos, nodectr;
206.     node *temp;
207.     if(start == NULL)
208.     {

```

```

209.     printf("\n Empty List");
210.     getch();
211.     return;
212. }
213. else
214. {
215.     printf("\n Enter the position of the node to delete: ");
216.     scanf("%d", &pos);
217.     nodectr = countnode(start);
218.     if(pos > nodectr)
219.     {
220.         printf("\nthis node does not
221.         exist"); getch();
222.         return;
223.     }
224.     if(pos > 1 && pos < nodectr)
225.     {
226.         temp =
227.         start; i = 1;
228.         while(i < pos)
229.         {
230.             temp = temp -> right;
231.             i++;
232.         }
233.         temp -> right -> left = temp -> left;
234.         temp -> left -> right = temp -> right;
235.         free(temp);
236.         printf("\n node deleted..");
237.     }
238.     else
239.     {
240.         printf("\n It is not a middle position..");
241.         getch();
242.     }
243. }
244. }
245. void main(void)
246. {
247.     int ch, n;
248.     clrscr();
249.     while(1)
250.     {
251.         ch = menu();
252.         switch( ch)
253.         {
254.             case 1 :
255.                 printf("\n Enter Number of nodes to create: ");
256.                 scanf("%d", &n);
257.                 createlist(n);
258.
259.                 printf("\n List
260.                 created.."); break;

```

```
261.      :
262.      dll_insert_beg();
263.      break;
264.      :
265.      dll_insert_end();
266.      break;
267.      :
268.      dll_insert_mid();
269.      break;
270.      :
271.      dll_delete_beg();
272.      break;
273.      case 6 : dll_delete_last();
274.      break;
275.      case 7 :
276.      dll_delete_mid();
277.      break;
278.      case 8 :
279.      traverse_left_to_right();
280.      break;
281.      case 9 :
282.      traverse_right_to_left();
283.      break;
284.      case 10 :
285.      printf("\n Number of nodes: %d", countnode(start));
286.      break;
287.      case 11:
288.      exit(0);
289.      }
290.      getch();
291.      }
292.      }
293.
294.
```