

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <conio.h>
4
5 struct Node{
6     int data;
7     struct Node *link;
8 };
9
10
11
12 //Add some more nodes
13 struct Node* add_some_nodes(struct Node *head, int data){
14     struct Node *ptr = NULL;
15     ptr = malloc(sizeof(struct Node));
16     ptr->data = data;
17     ptr->link = NULL;
18
19     head->link = ptr;
20     head = ptr;
21     return head;
22 }
23
24 //Traverse the list
25 void traverse_list(struct Node *head){
26     struct Node *ptr = head;
27
28     if(ptr == NULL){
29         printf("\nList is already empty\n");
30         return;
31     }else{
32         printf("\nData inside the list is\n");
33         while(ptr != NULL){
34             printf("%d\t", ptr->data);
35             ptr = ptr->link;
36         }
37         printf("\n");
38     }
39 }
40
41
42 //Add new nodes at the end
43 void add_at_end(struct Node *head, struct Node *last){
44     struct Node *ptr = head;
45
46     if(ptr == NULL){
47         printf("List is already empty\n");
48         return;
49     }else{
50         while(ptr->link != NULL){
51             ptr = ptr->link;
52         }
53         ptr->link = last;
54     }
55 }
56
57
58 //Add new node at beg
```

```
59 void add_new_at_beg(struct Node **head, struct Node *first){
60     struct Node *ptr = *head;
61
62     first->link = *head;
63     *head = first;
64 }
65
66
67 //Count number of nodes
68 int count_no_of_nodes(struct Node *head){
69     struct Node *ptr = head;
70     int count = 0;
71
72     if(ptr == NULL){
73         return count;
74     }else{
75         while(ptr != NULL){
76             count++;
77             ptr = ptr->link;
78         }
79         return count;
80     }
81 }
82
83
84 //Add new node at any pos
85 void add_at_pos(struct Node **head, struct Node *new, int pos){
86     struct Node *ptr = *head;
87
88     if(pos == 1){
89         add_new_at_beg(head, new);
90         return;
91     }else{
92         pos--;
93         while(--pos){
94             ptr = ptr->link;
95         }
96         new->link = ptr->link;
97         ptr->link = new;
98     }
99 }
100
101
102 //Delete first node
103 void del_first_node(struct Node **head){
104     struct Node *ptr = *head;
105
106     if(ptr == NULL){
107         printf("List is already empty\n");
108         return;
109     }else{
110         *head = (*head)->link;
111         free(ptr);
112         ptr = NULL;
113     }
114 }
115
116 //Delete last node
117 void del_last_node(struct Node **head){
118     struct Node *temp1 = *head;
```

```
119 struct Node *temp2 = *head;
120
121 if(*head == NULL){
122     printf("List is already empty\n");
123     return;
124 }else if((*head)->link == NULL){
125     free(*head);
126     *head = NULL;
127 }else{
128     while(temp1->link != NULL){
129         temp2 = temp1;
130         temp1 = temp1->link;
131     }
132     temp2->link = NULL;
133     free(temp1);
134     temp1 = NULL;
135 }
136
137 }
138
139
140
141 //Delete a node at some position
142 void del_at_pos(struct Node **head, int pos){
143     struct Node *ptr = *head;
144     struct Node *temp1 = *head;
145     struct Node *temp2 = *head;
146
147     if(ptr == NULL){
148         printf("List is already empty\n");
149         return;
150 }else if((*head)->link == NULL){
151     free(*head);
152     *head = NULL;
153 }else{
154     while(--pos){
155         temp1 = temp2;
156         temp2 = temp1->link;
157     }
158     temp1->link = temp2->link;
159     free(temp2);
160     temp2 = temp1 = NULL;
161 }
162 }
163
164
165
166 //Delete all nodes (forward deletion)
167 void del_all_nodes(struct Node **head){
168     struct Node *ptr = NULL;
169
170     if(*head == NULL){
171         printf("List is already empty\n");
172         return;
173     }else{
174         while((*head) != NULL){
175             ptr = *head;
176             *head = (*head)->link;
177             free(ptr);
178             ptr = NULL;
```

```
179     }
180 }
181 }
182
183
184
185
186 //Testing
187 int main(){
188     struct Node *head = NULL;
189     head = malloc(sizeof(struct Node));
190     head->data = 100;
191     head->link = NULL;
192
193     struct Node *ptr = head;
194
195     int yourChoice;
196     do{
197         system("CLS");
198         printf("\n\n");
199         printf("\t <Insertion Section>\n");
200         printf("<1>. Add some new nodes\n");
201         printf("<2>. Traverse the list\n");
202         printf("<3>. Add new node at the end\n");
203         printf("<4>. Add new node at beg\n");
204         printf("<5>. Count number of nodes\n");
205         printf("<6>. Add new node at any position\n");
206         printf("\t <Deletion Section>\n");
207         printf("<7>. Delete first Node\n");
208         printf("<8>. Delete last Node\n");
209         printf("<9>. Delete node at any position\n");
210         printf("<10>. Delete all nodes in one short\n");
211         printf("<11>. Exit\n");
212         printf("Enter you choice: ");
213         scanf("%d", &yourChoice);
214
215         switch(yourChoice){
216             case 1:
217                 printf("\nEnter number of new nodes to add: ");
218                 int newNodes, newData;
219                 scanf("%d", &newNodes);
220                 for(int i = 1; i <= newNodes; i++){
221                     printf("\tEnter data of new node: ");
222                     scanf("%d", &newData);
223                     ptr = add_some_nodes(ptr, newData);
224                 }
225                 printf("\n\tHit enter to continue:-...");
226                 getch();
227                 break;
228
229             case 2:
230                 traverse_list(head);
231                 printf("\n\tHit enter to continue:-...");
232                 getch();
233                 break;
234
235             case 3:
236                 printf("\n\tEnter data of new node: ");
237                 scanf("%d", &newData);
238                 struct Node *last = malloc(sizeof(struct Node));
```

```
239         last->data = newData;
240         last->link = NULL;
241
242         add_at_end(head, last);
243         break;
244
245     case 4:
246         printf("\n\tEnter data of new node: ");
247         scanf("%d", &newData);
248         struct Node *first = malloc(sizeof(struct Node));
249         first->data = newData;
250         first->link = NULL;
251
252         add_new_at_beg(&head, first);
253         break;
254
255     case 5:
256         printf("\n\t");
257         int noOfNodes = 0;
258         noOfNodes = count_no_of_nodes(head);
259
260         printf("No of nodes are: %d", noOfNodes);
261         printf("\n\tHit enter to continue:-...");
262         getch();
263         break;
264
265     case 6:
266         printf("\n\tEnter position of new node to insert: ");
267         int pos;
268         scanf("%d", &pos);
269
270         printf("\n\tEnter data of new node: ");
271         scanf("%d", &newData);
272         struct Node *newNode = malloc(sizeof(struct Node));
273
274         newNode->data = newData;
275         newNode->link = NULL;
276
277         add_at_pos(&head, newNode, pos);
278         break;
279
280     case 7:
281         del_first_node(&head);
282         break;
283
284     case 8:
285         del_last_node(&head);
286         break;
287
288     case 9:
289         printf("\tEnter position no to delete: ");
290         scanf("%d", &pos);
291
292         del_at_pos(&head, pos);
293         break;
294
295     case 10:
296         del_all_nodes(&head);
297         break;
298
```

```
299         case 11:
300             exit(1);
301
302         default:
303             printf("Invalid Choice\n");
304     }
305 }while(1);
306
307
308 return 0;
309 }
310
311
```