

```

1 //Linked List 5
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct node{
6     int data;
7     struct node *next;
8 };
9 void insertfirst(struct node **headptr){
10     struct node *newnode;
11     int value;
12     printf("Enter value: ");
13     scanf("%d",&value);
14     newnode = (struct node*)malloc(sizeof(struct node));
15     newnode->data = value;
16     newnode->next = NULL;
17     if(*headptr == NULL)
18         (*headptr) = newnode;
19     else{
20         newnode->next = (*headptr);
21         (*headptr) = newnode;
22     }
23 }
24 void insertpos(struct node **headptr){
25     struct node *newnode,*temp;
26     int count=0,currpos=1,value,pos;
27     printf("Enter value: ");
28     scanf("%d",&value);
29     newnode = (struct node*)malloc(sizeof(struct node));
30     newnode->data = value;
31     newnode->next = NULL;
32     if((*headptr) == NULL){
33         (*headptr) = newnode;
34     }
35     else{
36         temp = (*headptr);
37         while(temp != NULL){
38             count++;
39             temp = temp->next;
40         }
41         printf("There are %d elements in the list.Enter the position where you want to insert the value: ",count);
42         scanf("%d",&pos);
43         if(pos > (count+1)){
44             printf("no such position\n");

```

```

45         return;
46     }
47     if(pos == count+1){
48         temp = (*headptr);
49         while(temp->next != NULL)
50             temp = temp->next;
51         temp->next = newnode;
52     }
53     else{
54         temp = (*headptr);
55         while(temp->next != NULL){
56             if(currpos == (pos-1)){
57                 newnode->next = temp->next;
58                 temp->next = newnode;
59                 break;
60             }
61             currpos++;
62             temp = temp->next;
63         }
64     }
65 }
66 }
67 void insertlast(struct node **headptr){
68     struct node *newnode,*temp;
69     int value;
70     printf("Enter value: ");
71     scanf("%d",&value);
72     newnode = (struct node*)malloc(sizeof(struct node));
73     newnode->data = value;
74     newnode->next = NULL;
75     temp = (*headptr);
76     if(*headptr == NULL)
77         (*headptr) = newnode;
78     else{
79         while(temp->next != NULL)
80             temp = temp->next;
81         temp->next = newnode;
82     }
83 }
84 void deletelast(struct node **headptr){
85     struct node *temp;
86     temp = (*headptr);
87     if((*headptr) == NULL)
88         printf("The list is empty\n");

```

```

89     else if((*headptr)->next == NULL)
90         (*headptr) = NULL;
91     else{
92         temp = *headptr;
93         while((temp->next)->next != NULL)
94             temp = temp->next;
95         temp->next = NULL;
96     }
97 }
98 void display(struct node *temp){
99     if(temp == NULL){
100         printf("The list is empty\n");
101         return;
102     }
103     else{
104         while(temp != NULL){
105             printf("%d\t",temp->data);
106             temp = temp->next;
107         }
108         printf("\n");
109     }
110 }
111
112 int main(int argc,char **argv){
113     int choice;
114     struct node *head = NULL;
115     while(choice != 6){
116         printf("Enter choice 1)insertfirst 2)insertpos 3)insertlast 4)deletelast 5)display 6)exit : ");
117         scanf("%d",&choice);
118         switch(choice){
119             case 1:insertfirst(&head);break;
120             case 2:insertpos(&head);break;
121             case 3:insertlast(&head);break;
122             case 4:deletelast(&head);break;
123             case 5:display(head);break;
124             case 6:
125                 default:exit(0);
126         }
127     }
128     return 0;
129 }
130

```