

```

1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct node
5  {
6      struct node *prev;
7      int data;
8      struct node *next;
9  };
10 struct node *head=NULL;
11 void insertbegindoubly();
12 void displayforward();
13 void insertposition(int,int,struct node *);
14 void insertend(int);
15 void displaybackward();
16 void deletedobly();
17
18 int main()
19 {
20     int item,n;
21     insertbegindoubly();
22     insertbegindoubly();
23     insertbegindoubly();
24     insertbegindoubly();
25     displayforward();
26     displaybackward();
27     insertposition(n,item,head);
28     displayforward();
29     printf("\nin backwards\n");
30     displaybackward();
31     insertend(item);
32     displayforward();
33
34
35
36     return 0;
37 }
38
39 void insertbegindoubly()
40 {
41     struct node *newnode;
42     newnode=(struct node *)malloc(sizeof(struct node));
43     printf("enter data : ");
44     scanf("%d",&newnode->data);
45     newnode->next=NULL;
46     if(head==NULL){
47         head=newnode;
48     }
49
50     else{
51         newnode->next=head;
52         head->prev=newnode;
53         head=newnode;
54     }
55 }
56
57 void displayforward()
58 {
59     struct node *temp;
60     if(head==NULL){
61         printf("List is empty\n");
62     }
63     else{
64         temp=head;
65         printf("list elements in doubly linked list are:\n");
66         while(temp!=NULL){

```

```

67         printf("%d\t",temp->data);
68         temp=temp->next;
69     }
70 }
71 }
72
73 void insertposition(int pos,int data,struct node *head)
74 {
75     struct node *temp;
76     struct node *newnode=(struct node *)malloc(sizeof(struct node));
77     newnode->next=NULL;
78     printf("\nenter at position:");
79
80     scanf("%d",&pos);
81     if(pos==1){
82         insertbegindoubly();
83     }
84     else{
85         pos=pos-1;
86         printf("enter data:");
87         scanf("%d",&newnode->data);
88         newnode->next=NULL;
89         if(pos==0)
90         {
91             head=newnode;
92         }
93         else{
94             temp=head;
95             while(--pos)
96             {
97                 temp=temp->next;
98             }
99             newnode->next = temp->next;
100             temp->next = newnode;
101         }
102     }
103 }
104 }
105
106 }
107
108 void insertend(int item)
109 {
110     struct node *temp;
111     struct node *newnode=(struct node *)malloc(sizeof(struct node ));
112     printf("\nenter data for end node ");
113     scanf("%d",&newnode->data);
114     if(head==NULL){
115         newnode->next=NULL;
116         head=newnode;
117     }
118     else{
119         temp=head;
120         while(temp->next!=NULL){
121             temp=temp->next;
122         }
123         temp->next=newnode;
124         newnode->next=NULL;
125     }
126 }
127
128 void displaybackward()
129 {
130     struct node *templ=head;
131
132     while(templ->next!=NULL){

```

```

133         templ=templ->next;
134     }
135     while(templ!=head){
136         printf("%d\t",templ->data);
137         templ=templ->prev;
138     }
139     printf("%d\t",templ->data);
140
141
142
143 }
144 void deletedobly();
145 {
146     struct node*temp;
147     if(head==NULL)
148     {
149         printf("list is empty")
150     }
151     else if(head->next==NULL)
152     {
153         head=NULL;
154         free(head);
155         printf("\n begin node deleted");
156     }
157     else
158     {
159         temp=head;
160         head=head->next;
161         head->prev=NULL;
162         free(temp);
163         print("\n begin node deleted");
164     }
165 }
166

```