```
1. #include<stdio.h>
2. #include<stdlib.h>
3. struct node
4. {
5.     struct node *prev;
6.     struct node *next;
7.     int data;
8. };
9. struct node *head;
10.void insertion_beginning();
11.void insertion_last();
12.void insertion_specified();
13.void deletion_beginning();
14.void deletion_last();
15.void deletion_specified();
16.void display();
17.void search();
18.void main ()
19.{
20.int choice =0;
21.    while(choice != 9)
22.    {
23.        printf("\n*********Main Menu*********\n");
24.        printf("\nChoose one option from the following list ...\n");
25.        printf("\
   n==========================================\
   =\n");
26.        printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any rando
   m location\n4.Delete from Beginning\n
27.        5.Delete from last\n6.Delete the node after the given data\
   n7.Search\n8.Show\n9.Exit\n");
28.        printf("\nEnter your choice?\n");
29.        scanf("\n%d",&choice);
30.        switch(choice)
31.        {
32.            case 1:
```

```
33.             insertion_beginning();
34.             break;
35.             case 2:
36.                     insertion_last();
37.             break;
38.             case 3:
39.             insertion_specified();
40.             break;
41.             case 4:
42.             deletion_beginning();
43.             break;
44.             case 5:
45.             deletion_last();
46.             break;
47.             case 6:
48.             deletion_specified();
49.             break;
50.             case 7:
51.             search();
52.             break;
53.             case 8:
54.             display();
55.             break;
56.             case 9:
57.             exit(0);
58.             break;
59.             default:
60.             printf("Please enter valid choice..");
61.         }
62.    }
63.}
64.void insertion_beginning()
65.{
66.   struct node *ptr;
67.   int item;
68.   ptr = (struct node *)malloc(sizeof(struct node));
```

```
69.  if(ptr == NULL)
70.  {
71.      printf("\nOVERFLOW");
72.  }
73.  else
74.  {
75.   printf("\nEnter Item value");
76.   scanf("%d",&item);
77.
78.  if(head==NULL)
79.  {
80.      ptr->next = NULL;
81.      ptr->prev=NULL;
82.      ptr->data=item;
83.      head=ptr;
84.  }
85.  else
86.  {
87.      ptr->data=item;
88.      ptr->prev=NULL;
89.      ptr->next = head;
90.      head->prev=ptr;
91.      head=ptr;
92.  }
93.  printf("\nNode inserted\n");
94.}
95.
96.}
97.void insertion_last()
98.{
99.  struct node *ptr,*temp;
100.   int item;
101.   ptr = (struct node *) malloc(sizeof(struct node));
102.   if(ptr == NULL)
103.   {
104.       printf("\nOVERFLOW");
```

```
105.  }
106.  else
107.  {
108.     printf("\nEnter value");
109.     scanf("%d",&item);
110.      ptr->data=item;
111.     if(head == NULL)
112.     {
113.        ptr->next = NULL;
114.        ptr->prev = NULL;
115.        head = ptr;
116.     }
117.     else
118.     {
119.       temp = head;
120.       while(temp->next!=NULL)
121.       {
122.          temp = temp->next;
123.       }
124.       temp->next = ptr;
125.       ptr ->prev=temp;
126.       ptr->next = NULL;
127.       }
128.
129.     }
130.    printf("\nnode inserted\n");
131.   }
132.void insertion_specified()
133.{
134.  struct node *ptr,*temp;
135.  int item,loc,i;
136.  ptr = (struct node *)malloc(sizeof(struct node));
137.  if(ptr == NULL)
138.  {
139.     printf("\n OVERFLOW");
140.  }
```

```
141.  else
142.  {
143.     temp=head;
144.     printf("Enter the location");
145.     scanf("%d",&loc);
146.     for(i=0;i<loc;i++)
147.     {
148.        temp = temp->next;
149.        if(temp == NULL)
150.        {
151.           printf("\n There are less than %d elements", loc);
152.           return;
153.        }
154.     }
155.     printf("Enter value");
156.     scanf("%d",&item);
157.     ptr->data = item;
158.     ptr->next = temp->next;
159.     ptr -> prev = temp;
160.     temp->next = ptr;
161.     temp->next->prev=ptr;
162.     printf("\nnode inserted\n");
163.  }
164.}
165.void deletion_beginning()
166.{
167.   struct node *ptr;
168.   if(head == NULL)
169.   {
170.      printf("\n UNDERFLOW");
171.   }
172.   else if(head->next == NULL)
173.   {
174.      head = NULL;
175.      free(head);
176.      printf("\nnode deleted\n");
```

```
177.    }
178.    else
179.    {
180.        ptr = head;
181.        head = head -> next;
182.        head -> prev = NULL;
183.        free(ptr);
184.        printf("\nnode deleted\n");
185.    }
186.
187.}
188.void deletion_last()
189.{
190.    struct node *ptr;
191.    if(head == NULL)
192.    {
193.        printf("\n UNDERFLOW");
194.    }
195.    else if(head->next == NULL)
196.    {
197.        head = NULL;
198.        free(head);
199.        printf("\nnode deleted\n");
200.    }
201.    else
202.    {
203.        ptr = head;
204.        if(ptr->next != NULL)
205.        {
206.            ptr = ptr -> next;
207.        }
208.        ptr -> prev -> next = NULL;
209.        free(ptr);
210.        printf("\nnode deleted\n");
211.    }
212.}
```

```
213.void deletion_specified()
214.{
215.    struct node *ptr, *temp;
216.    int val;
217.    printf("\n Enter the data after which the node is to be deleted : ");
218.    scanf("%d", &val);
219.    ptr = head;
220.    while(ptr -> data != val)
221.    ptr = ptr -> next;
222.    if(ptr -> next == NULL)
223.    {
224.        printf("\nCan't delete\n");
225.    }
226.    else if(ptr -> next -> next == NULL)
227.    {
228.        ptr ->next = NULL;
229.    }
230.    else
231.    {
232.        temp = ptr -> next;
233.        ptr -> next = temp -> next;
234.        temp -> next -> prev = ptr;
235.        free(temp);
236.        printf("\nnode deleted\n");
237.    }
238.}
239.void display()
240.{
241.    struct node *ptr;
242.    printf("\n printing values...\n");
243.    ptr = head;
244.    while(ptr != NULL)
245.    {
246.        printf("%d\n",ptr->data);
247.        ptr=ptr->next;
248.    }
```

```
}
void search()
{
    struct node *ptr;
    int item,i=0,flag;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        while (ptr!=NULL)
        {
            if(ptr->data == item)
            {
                printf("\nitem found at location %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        if(flag==1)
        {
            printf("\nItem not found\n");
        }
    }

}
```