# INSERTION AND DELETION OF DOUBLY LINKLIST

```
    #include<stdio.h>
1.  #include<stdlib.h>
2.  struct node
3.  {
4.      struct node *prev;
5.      struct node *next;
6.      int data;
7.  };
8.  struct node *head;
9.  void insertion_beginning()
10. {
11.    struct node *ptr;
12.    int item;
13.    ptr = (struct node *)malloc(sizeof(struct node));
14.    if(ptr == NULL)
15.    {
16.        printf("\nOVERFLOW");
17.    }
18.    else
19.    {
20.     printf("\nEnter Item value");
21.     scanf("%d",&item);
22.
23.    if(head==NULL)
24.    {
25.        ptr->next = NULL;
26.        ptr->prev=NULL;
27.        ptr->data=item;
28.        head=ptr;
29.    }
30.    else
```

```
31.   {
32.       ptr->data=item;
33.       ptr->prev=NULL;
34.       ptr->next = head;
35.       head->prev=ptr;
36.       head=ptr;
37.   }
38.   printf("\nNode inserted\n");
39. }
40.
41. }
42. void insertion_last()
43. {
44.   struct node *ptr,*temp;
45.   int item;
46.   ptr = (struct node *) malloc(sizeof(struct node));
47.   if(ptr == NULL)
48.   {
49.       printf("\nOVERFLOW");
50.   }
51.   else
52.   {
53.       printf("\nEnter value");
54.       scanf("%d",&item);
55.        ptr->data=item;
56.       if(head == NULL)
57.       {
58.           ptr->next = NULL;
59.           ptr->prev = NULL;
60.           head = ptr;
61.       }
62.       else
63.       {
64.         temp = head;
65.         while(temp->next!=NULL)
66.         {
67.             temp = temp->next;
```

```
        }
        temp->next = ptr;
        ptr ->prev=temp;
        ptr->next = NULL;
        }

      }
     printf("\nnode inserted\n");
    }
void insertion_specified()
{
   struct node *ptr,*temp;
   int item,loc,i;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\n OVERFLOW");
   }
   else
   {
      temp=head;
      printf("Enter the location");
      scanf("%d",&loc);
      for(i=0;i<loc;i++)
      {
         temp = temp->next;
         if(temp == NULL)
         {
            printf("\n There are less than %d elements", loc);
            return;
         }
      }
            printf("Enter value");
            scanf("%d",&item);
            ptr->data = item;
            ptr->next = temp->next;
            ptr -> prev = temp;
```

```
105.              temp->next = ptr;
106.              temp->next->prev=ptr;
107.              printf("\nnode inserted\n");
108.           }
109.       }
110.       void deletion_beginning()
111.       {
112.          struct node *ptr;
113.          if(head == NULL)
114.          {
115.             printf("\n UNDERFLOW");
116.          }
117.          else if(head->next == NULL)
118.          {
119.             head = NULL;
120.             free(head);
121.             printf("\nnode deleted\n");
122.          }
123.          else
124.          {
125.             ptr = head;
126.             head = head -> next;
127.             head -> prev = NULL;
128.             free(ptr);
129.             printf("\nnode deleted\n");
130.          }
131.
132.       }
133.       void deletion_last()
134.       {
135.          struct node *ptr;
136.          if(head == NULL)
137.          {
138.             printf("\n UNDERFLOW");
139.          }
140.          else if(head->next == NULL)
141.          {
```

```
142.             head = NULL;
143.             free(head);
144.             printf("\nnode deleted\n");
145.         }
146.         else
147.         {
148.             ptr = head;
149.             if(ptr->next != NULL)
150.             {
151.                 ptr = ptr -> next;
152.             }
153.     0       ptr -> prev -> next = NULL;
154.             free(ptr);
155.             printf("\nnode deleted\n");
156.         }
157.     }
158.     void deletion_specified()
159.     {
160.         struct node *ptr, *temp;
161.         int val;
162.         printf("\n Enter the data after which the node is to be deleted : ");
163.         scanf("%d", &val);
164.         ptr = head;
165.         while(ptr -> data != val)
166.         ptr = ptr -> next;
167.         if(ptr -> next == NULL)
168.         {
169.             printf("\nCan't delete\n");
170.         }
171.         else if(ptr -> next -> next == NULL)
172.         {
173.             ptr ->next = NULL;
174.         }
175.         else
176.         {
177.             temp = ptr -> next;
178.             ptr -> next = temp -> next;
```

```
179.            temp -> next -> prev = ptr;
180.            free(temp);
181.            printf("\nnode deleted\n");
182.        }
183.    }
184.    void display()
185.    {
186.        struct node *ptr;
187.        printf("\n printing values...\n");
188.        ptr = head;
189.        while(ptr != NULL)
190.        {
191.            printf("%d\n",ptr->data);
192.            ptr=ptr->next;
193.        }
194.    }
195.    void main ()
196.    {
197.
198.            insertion_beginning();
199.            insertion_beginning();
200.            insertion_beginning();
201.                    insertion_last();
202.            insertion_specified();
203.            deletion_beginning();
204.            deletion_last();
205.            display();
206.    }
207.
```