```c
//SHREYAS SAWANT D7A 55
//Implement Doubly Linked List using ADT

#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;

void insertion_beginning()
{
   struct node *ptr;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value\n");
    scanf("%d",&item);

   if(head==NULL)
   {
       ptr->next = NULL;
       ptr->prev=NULL;
       ptr->data=item;
       head=ptr;
   }
   else
   {
       ptr->data=item;
       ptr->prev=NULL;
       ptr->next = head;
       head->prev=ptr;
       head=ptr;
   }
   printf("\nNode inserted\n");
}
}


void insertion_last()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
       printf("\nEnter value\n");
       scanf("%d",&item);
        ptr->data=item;
       if(head == NULL)
       {
           ptr->next = NULL;
           ptr->prev = NULL;
           head = ptr;
       }
       else
       {
          temp = head;
          while(temp->next!=NULL)
          {
              temp = temp->next;
          }
          temp->next = ptr;
          ptr ->prev=temp;
          ptr->next = NULL;
          }

       }
     printf("\nNode inserted\n");
    }
void insertion_specified()
{
```

```
85      struct node *ptr,*temp;
86      int item,loc,i;
87      ptr = (struct node *)malloc(sizeof(struct node));
88      if(ptr == NULL)
89      {
90          printf("\n OVERFLOW");
91      }
92      else
93      {
94          temp=head;
95          printf("\nEnter the location\n");
96          scanf("%d",&loc);
97          for(i=0;i<loc;i++)
98          {
99              temp = temp->next;
100             if(temp == NULL)
101             {
102                 printf("\nThere are less than %d elements\n", loc);
103                 return;
104             }
105         }
106         printf("\nEnter value\n");
107         scanf("%d",&item);
108         ptr->data = item;
109         ptr->next = temp->next;
110         ptr -> prev = temp;
111         temp->next = ptr;
112         temp->next->prev=ptr;
113         printf("\nNode inserted\n");
114     }
115 }
116 void deletion_beginning()
117 {
118     struct node *ptr;
119     if(head == NULL)
120     {
121         printf("\n UNDERFLOW\n");
122     }
123     else if(head->next == NULL)
124     {
125         head = NULL;
126         free(head);
127         printf("\nNode deleted\n");
128     }
129     else
130     {
131         ptr = head;
132         head = head -> next;
133         head -> prev = NULL;
134         free(ptr);
135         printf("\nNode deleted\n");
136     }
137
138 }
139 void deletion_last()
140 {
141     struct node *ptr;
142     if(head == NULL)
143     {
144         printf("\n UNDERFLOW\n");
145     }
146     else if(head->next == NULL)
147     {
148         head = NULL;
149         free(head);
150         printf("\nNode deleted\n");
151     }
152     else
153     {
154         ptr = head;
155         if(ptr->next != NULL)
156         {
157             ptr = ptr -> next;
158         }
159         ptr -> prev -> next = NULL;
160         free(ptr);
161         printf("\nNode deleted\n");
162     }
163 }
164 void deletion_specified()
165 {
166     struct node *ptr, *temp;
167     int val;
168     printf("\nEnter the data after which the node is to be deleted : ");
```

```
169         scanf("%d", &val);
170         ptr = head;
171         while(ptr -> data != val)
172         ptr = ptr -> next;
173         if(ptr -> next == NULL)
174         {
175             printf("\nEmpty List\n");
176         }
177         else if(ptr -> next -> next == NULL)
178         {
179             ptr ->next = NULL;
180         }
181         else
182         {
183             temp = ptr -> next;
184             ptr -> next = temp -> next;
185             temp -> next -> prev = ptr;
186             free(temp);
187             printf("\nNode deleted\n");
188         }
189     }
190     void display()
191     {
192         struct node *ptr;
193         if(head==NULL)
194             printf("\nEMPTY LIST\n");
195         else{
196         printf("\n Elements of list are:\n");
197         ptr = head;
198         while(ptr != NULL)
199         {
200             printf("%d ",ptr->data);
201             ptr=ptr->next;
202         }printf("\n");}
203     }
204     void search()
205     {
206         struct node *ptr;
207         int item,i=0,flag;
208         ptr = head;
209         if(ptr == NULL)
210         {
211             printf("\nEmpty List\n");
212         }
213         else
214         {
215             printf("\nEnter item which you want to search?\n");
216             scanf("%d",&item);
217             while (ptr!=NULL)
218             {
219                 if(ptr->data == item)
220                 {
221                     printf("\nItem found at location %d \n",i+1);
222                     flag=0;
223                     break;
224                 }
225                 else
226                 {
227                     flag=1;
228                 }
229                 i++;
230                 ptr = ptr -> next;
231             }
232             if(flag==1)
233             {
234                 printf("\nItem not found\n");
235             }
236         }
237     }
238     void main ()
239     {
240     int choice =0;
241         while(choice != 9)
242         {
243
244             printf("\n1.Insert in beginning\n2.Insert at last\n3.Insert at any random
    location\n4.Delete from Beginning\n5.Delete from last\n6.Delete the node after the given
    data\n7.Search\n8.Show\n9.Exit\n");
245             printf("\nEnter your choice?\n");
246             scanf("\n%d",&choice);
247             switch(choice)
248             {
249                 case 1:
250                 insertion_beginning();
```

```
251                 break;
252                 case 2:
253                         insertion_last();
254                 break;
255                 case 3:
256                 insertion_specified();
257                 break;
258                 case 4:
259                 deletion_beginning();
260                 break;
261                 case 5:
262                 deletion_last();
263                 break;
264                 case 6:
265                 deletion_specified();
266                 break;
267                 case 7:
268                 search();
269                 break;
270                 case 8:
271                 display();
272                 break;
273                 case 9:
274                 exit(0);
275                 break;
276                 default:
277                 printf("Please enter valid choice..");
278             }
279         }
280     }
281
282
283
284
```