# INSERTION AND DELETION OF DOUBLY LINKLIST

```
#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");
    scanf("%d",&item);

   if(head==NULL)
   {
       ptr->next = NULL;
       ptr->prev=NULL;
       ptr->data=item;
       head=ptr;
   }
   else
```

```c
    {
        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");
        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()
{
    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;
       temp->next = ptr;
       temp->next->prev=ptr;
       printf("\nnode inserted\n");
   }
}
void deletion_beginning()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }

}
void deletion_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
```

```c
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
0        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }
}
void deletion_specified()
{
    struct node *ptr, *temp;
    int val;
    printf("\n Enter the data after which the node is to be deleted : ");
    scanf("%d", &val);
    ptr = head;
    while(ptr -> data != val)
    ptr = ptr -> next;
    if(ptr -> next == NULL)
    {
        printf("\nCan't delete\n");
    }
    else if(ptr -> next -> next == NULL)
    {
        ptr ->next = NULL;
    }
    else
    {
        temp = ptr -> next;
        ptr -> next = temp -> next;
```

```
            temp -> next -> prev = ptr;
            free(temp);
            printf("\nnode deleted\n");
        }
    }
    void display()
    {
        struct node *ptr;
        printf("\n printing values...\n");
        ptr = head;
        while(ptr != NULL)
        {
            printf("%d\n",ptr->data);
            ptr=ptr->next;
        }
    }
    void main ()
    {

            insertion_beginning();
            insertion_beginning();
            insertion_beginning();
                insertion_last();
            insertion_specified();
            deletion_beginning();
            deletion_last();
            display();
    }

```