6) Design, Develop and implement C program for the following operations on doubly linked list.

a. Create doubly linked list of N nodes with integer data by adding each node at the front.
b. Delete the node of a given data if it is found, otherwise display appropriate message.
c. Insert a node to the left of the node whose key value is read as input.
d. Display the contents of the list.

```
#include <stdio.h>
#include<stdlib.h>
typedef struct student
{
  int data;
  struct student *next, *prev;
}NODE;

NODE* getnode( )
{
  NODE *x;
  x=(NODE*)malloc(sizeof(NODE));
  printf("\n Enter Data of Node to be Inserted: ");
  scanf("%d",&x->data);
  x->next=x->prev=NULL;
 return x;
}
NODE* insert_front(NODE* first)
{
        NODE *temp;
        if(first==NULL)
        {
                temp=getnode();
                first=temp;
        }
        else
        {
                temp=getnode();
                temp->next=first;
                first->prev=temp;
                first=temp;
        }
        return first;
}
NODE* insert_left(NODE* first)
{
  NODE *temp,*cur,*pre;
  int data;
  if(first==NULL)
  {
        temp=getnode();
        first=temp;
  }
 else
 {
        printf("Enter the node data to which left part new node to be inserted: ");
        scanf("%d",&data);
         temp=getnode();
        cur=first;
        while(cur->data!=data)
        {
                pre=cur;
               cur=cur->next;
         }
         pre->next=temp;
         temp->prev=pre;
         temp->next=cur;
        cur->prev=temp;
 }
 return first;
}
```

NODE* delete_node(NODE* first)

```
{
 NODE *cur;
 int data;
 cur=first;
 printf("Enter the data of the NODE to be deleted:  ");
 scanf("%d",&data);
 if(first==NULL)
 {
        printf("\n List is empty\n");
 }
 else if(first->data==data)
 {
         first=first->next;
         free(cur);
 }
 else
 {
        while(cur!=NULL)
        {
               if(cur->data==data)
               break;
               cur=cur->next;
        }
        if(cur!=NULL)
        {
               if(cur->next!=NULL)
               {
                      (cur->next)->prev=cur->prev;
                      (cur->prev)->next=cur->next;
                      free(cur);
```

```
                    }
                    else
                    {
                              (cur->prev)->next=NULL;
                              free(cur);
                    }
          }
          else
          {
                    printf("No such node is present in the list\n");
          }
  }
  return first;
}
```

NODE* display(NODE* first)

```
{
          NODE *cur;
          if(first == NULL)
                    printf("No nodes present\n");
          else
          {
                    cur=first;
                    while(cur!=NULL)
                    {
                    printf("-->%d", cur->data);
                    cur = cur->next;
                    }
          }
          return first;
}
```

int main()

```
{
  NODE *first;
  first=NULL;
  int ch;
  while(1)
  {
    printf("\n1.InsertFront\t 2. InsertLeft\t 3.Delete\t 4.Display\t 5.exit\n");
    printf("Enter Your Choice: ");
    scanf("%d",&ch);
    switch(ch)
    {
      case 1:first=insert_front(first);
      break;
      case 2:first=insert_left(first);
      break;
      case 3:first=delete_node(first);
      break;
      case 4:first=display(first);
      break;
      case 5:exit(0);
      break;
      default: printf("\n Invalid choice\n");
      break;
    }
  }
  return 0;
}
```