Doubly linked list:-

```
Struct node
{
 int data;
 Struct node *next;
 Struct node * prev;
};
Struct node *head = 0, *newnode, *temp, *last;

Void create()
{
   int i, n;
   printf("Enter the no of elements: ");
   scanf("%d", &n);
   for(i=0; i<n; i++)
   {
      newnode = (Struct node *) malloc (sizeof
                (Struct node));
      printf("Enter the %d element: \n", i+1);
      scanf("%d", &newnode->data);
      newnode->prev = 0;
      newnode->next = 0;
      if(head==0)
      {
```

```
        temp = head->next;
    }
    else
    {
        temp->next = newnode;
        newnode->prev = temp;
        temp = newnode;
    }
  }
}

void display()
{
  temp = head;
  while(temp!=0)
  {
    printf("%d\n", temp->data);
    temp = temp->next;
  }
}

void insert_left()
{
  int pos, node, i=1;
  printf("Enter the position");
```

```
scanf("%d", &pos);
temp = head;
if (pos<1)
{
    printf("invalid position\n");
}
else if (pos==1)
{
    newnode = (struct node*) malloc(sizeof
        (struct node));
    printf("Enter data\n")
    scanf("%d", &newnode->data);
    newnode->prev=0;
    head->prev = newnode;
    newnode->next = head;
    head = newnode;
}
else
{
    newnode = (struct node *) malloc
        (sizeof(struct node));
    printf("Enter data\n");
    scanf("%d", &newnode->data);
    while (i<pos-1)
    {
```

```
        temp = temp->next;
        i++;
    }
    newnode->prev = temp;
    newnode->next = temp->next;
    temp->next = newnode;
    newnode->next->prev = newnode;
  }
}

void delete_pos()
{
    int pos, i=1;
    temp=head;
    printf("Enter position");
    scanf("%d", &pos);
    while(i<pos)
    {
        temp = temp->next;
        i++;
    }
    temp->prev->next = temp->next
    temp->next->prev = temp->prev
    free(temp);
}
```

```
void main()
{
    int choice, num;
    printf("enter \n1.create\n2. display\n3.
        insert left\n 4. delete at position\n5
        Exit\n");
    while(1)
    {
        printf("Enter operation");
        scanf("%d", &choice);
        if(choice == -1)
        {
            printf("completed\n");
        }
        else
        {
            switch(choice)
            {
                case 1: create();
                break;
                case 2: display();
                break;
                case 3: insert_left();
                break;
                default:
```

```
            case 5: exit(0);
                break;
         default : printf("invalid inp
                        ut\n");
        }
      }
   }
}
```

enter operation
1. create
2. display
3. insert left
4. delete at position
5. exit

enter operation
1
enter no of elements
4
enter element 1: 1
enter element 2: 2
enter element 3: 3
enter element 4: 4

enter operation
2
1
2
3
4
enter operation
3
enter position
3
enter data
enter operation
2
1
2
5
3
4
enter operation
4
enter position
3
enter operation
2
1
2