WAP to implement doubly linked list operations:
a) creation b) insertion of node to left/right of node
c) Delete node based on specific value
d) Display contents

```
#include< stdio.h>
# include < process.h>
  struct node
  {
      int info
      struct node * llink;
        struct node * rlink.
  };
  typedef struct node * NODE;
   NODE getnode ()
   {

       NODE x.
       x = (NODE) malloc (sizeof( structnode));
       if (x=NULL)
         {
               printf(" Mem full \n");
               exit(0);
         }
         return x;
   }
   NODE dinsertfront (int item, NODE head)
   {

            NODE temp , cur;
            temp= getnode().
            temp -> info =item;
              cur = head -> rlink;
               head -> rlink= temp ;
               temp -> llink = head;
temp -> rlink = cur;
  cur -> llink = temp;
  return head;
}
NODE dinsertrear (int item, NODE head)
 {
      NODE temp, cur;
      temp = getnode();
        temp -> info = item;
        cur = head -> llink;
          head -> llink = temp;
            temp -> rlink = head;
              temp -> llink = cur;
                cur -> rlink = temp;
            return head;
   }
   NODE ddeletefront (NODE head)
  {
      NODE    cur, next;
      if (head -> rlink == head)
        {
             printf ("Empty \n");
               return head;
        }
      cur = head -> rlink;
      next = cur -> rlink;
         head -> rlink = next;
         next -> llink = head;
          pf(" node deleted %d", cur->info);
           free (cur);
           return head;
     }
```

```
NODE  ddeletesear( NODE head)
{
        NODE cur,prev.
         if (head->rlink==head)
            {
                  printf("Empty list\n");
                  returnhead;
            }
           cur = head-> llink;
            prev = cur-> llink;
              head->llink= prev;
                prev->rlink =head;
                  printf("Node deleted %d", cur->info);
                  free (curr);
                  return head;
      }
      NODE  insertleftpos (int item, NODE head)
       {
            NODE temp, cur, prev;
              if (head->rlink ==head)
               {
                     Pf("Empty \n").
                  while (cur != head)
                  {
                      if (item == cur->info)
                          break;
                        cur =cur-> rlink;
                  }
                  if (cur == head)
                 {  Pf ("Key not found \n");
                          return head;
                 }
                 prev = cur-llink;
```

```
pf(" Enter towards left of %d=", item);
  temp = getnode();
   scanf("%d", & temp-> info);
    prev-> rlink= temp;
     temp-> llink= prev;
      cur-> llink = temp;
       temp-> rlink = cur;
        return head;
  }
  NODE insertrightpos(int item, NODE head)
  {
        NODE temp, cur, prev;
         if (head-> rlink ==head)
           {
              pf(" Empty list \n");
              return head;
           }
        cur =head -> llink;
         while (cur != head)
          {
             if (item == cur-> info)
              break;
              cur=cur-> llink;
          }
          if (cur ==head)
           {
              pf (" Key not found\n");
              return head;
           }
        prev= cur-> rlink;
         pf (" Enter towards right of %d =", item);
          temp= getnode();
           scanf("%d", &temp-> info);
```

```
prev->llink = temp;
 temp->rlink= prev.
 cur->rlink = temp;
  temp-> llink = cur;
  return head;
}
NODE delete all key ( int item, NODE head)
{
        NODE prev, cur, next,
        int count;
        if (head -> rlink== head)
          {
               pf (" Empty list\n");
               return head;
          }
        count=0;
        cur= head -> rlink.
          while ( cur != head)
          {
               if (item != cur->info)
                 cur= cur-> rlink.
               else
               {
                    count ++.
                    prev = cur -> llink;
                    next = cur -> rlink.
                    prev -> rlink = next,
                      next -> llink = prev.
                         free (cur);
                         cur =next,
                    }
                    }
                    if (count==0)
```

```
pf("Key not found \n");
else
pf(" key found @ %d pos & ase del \n", count);
   return head;
}
void display (NODE head)
 {
      NODE temp;
      if (head -> rlink == head)
        {
           pf ("List is empty \n");
           return;
          }
          pf ("contents of the link: \n");
          temp = head -> rlink;
          while (temp != head)
           {
              pf("%d", temp -> info);
                temp = temp -> rlink;
           }
          printf ("\n");
    void main()
     {
           NODE head, last;
           int item, ch;
           head=getnode();
             head-> rlink= head
             head-> llink= head;
               for(;;)
               {
                 printf ("\n\n 1. ins front \n 2. ins rear \n
                       3. Delete front \n 4. Delete rear
                       \n 5. Insert left pos \n 6. insert
```

```
right pos \n 7. Delete all key elements \n 8. Display
   \n 9. Exit \n");
  Scanf("%d", &ch);
  Switch( ch)
  {
     case 1:   printf(" Enter item to  be inserted @
                    front-end :");
               scanf ("%d", &item);
               last = dinsert front( item, head);
               break;
      Case 2:   printf(" Enter item  to insert @ rear
                     end :.");
                Scanf ("%d", &item);
                last = dinsert rear( item, head);
                break;
       Case 3:  last = ddelete front (head);
                break;
       Case 4:  last = ddelete rear (head);
                break;
        case 5:  pf ("Enter key item :");
                 Scanf ("%d", &item);
                 head = insert leftpos (item, head);
                 break;
          case 6:   pf (" Enter key item: ");
                       Scanf (" %d", & item);
                    head = insert rightpos (item, head);
                    break;
        Case 7 :    pf ("Enter key :");
                    Scanf ("%d", & item);
                    head = deleteallkey (item, head)
                    break;
         Case 8:    display (head);
                      break;
        default : exit(0);
      }
    }
  }
```