Doubly linked list :-

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

NODE getnode()
{
   NODE x;
   x=(NODE) malloc (sizeof(struct node));
   if(x==NULL)
   {
      printf("Memory full\n");
      exit(0);
   }
   return x;
}
void freenode(NODE x)
{
   free(x);
}
```

```
NODE dinsert_front(int item, NODE head)
{
    NODE temp, cur;
    temp = getnode();
    temp->info = item.
    cur = head->rlink;
    head->rlink = temp;
    temp->llink = head;
    cur->llink = temp;
    temp->rlink = ~~head~~ cur;
    return head;
}

NODE dinsert_rear(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 ddelete_front(NODE head)
{
    NODE cur, next;
    if(head->rlink == head)
    {
        printf("The DLL is empty\n");
        return head;
    }
```

```
        next= cur->rlink;
        head->rlink= next;
        next->llink=head;
        printf("The node delleted is %d", cur->info);
        freenode(cur);
        return head;
    }
NODE ddelete_reer(NODE head)
{
      NODE cur, prev;
      if (head->rlink==head)
      {
         printf("The DLL is empty \n");
         return head;
      }
      cur= head->llink;
      prev= cur->llink;
      prev->rlink= head;
      head->llink= prev;
      freenode(cur);        printf("The node deleted is %d", cur->info);
      return head;
    }

NODE insert_left_pos(int item, NODE head)
{
     NODE temp,cur, prev;
     int item2;
     if( head->rlink == head)
     {
        printf("List empty \n");
        return head;
     }
```

```
        cur = head -> rlink;
       while (cur != head)
       {
          if (item == cur -> info) break;
          cur = cur -> rlink;
       }
       if (cur == head)
       {
          printf("Key not found\n");
          return head;
       }
       prev = cur -> llink;
       temp = getnode();
       printf("Enter towards left ~~of~~ %d : ", item);
       scanf("%d", &item2);
       temp -> info = item2;
       prev -> rlink = temp;
       temp -> llink = prev;
       cur -> llink = temp;
       temp -> rlink = cur;
       return head;
}
NODE insert_rightpos (int item, NODE head)
{
   NODE temp, cur, next;
   int item2;
   if ( head -> rlink == head)
   {
      printf("List Empty\n");
      return head;
   }
   cur = head -> rlink;
```

```
while (cur!=head)
{
    if(item==cur->info) break;
    cur=cur->rlink;
}
if(cur==head)
{
    printf("Key not found \n");
    return head;
}
next=cur->rlink;
temp=getnode();
printf("Enter towards the ~~Left~~ Right of %d: ",item);
scanf("%d",&item2);
temp->info=item2;
next->~~l~~link=temp;
temp->rlink=next;
~~temp~~ 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)
    {
        printf("The DLL is empty \n");
        return head;
    }
    count=0;
    cur=head->rlink;
    while(cur!=head)
    {
        if( item!=cur->~~l~~info)
            cur=cur->rlink;
        else
        {
            count++;
            prev = cur->llink;
            next= cur->rlink;
            prev->rlink =next;
            next->llink = prev;
            freenode (cur);
            cur= next;
        }
    }
    if (count==0):
    { printf(" Key not found \n");
    else
        printf("key found at %d positions!! and are deleted ", count);
    return head;
}
void searching (int key, NODE head)
{
    NODE temp, cur;
    if(head->rlink == head)
    {
        printf(" List empty \n");
        return;
    }
    cur= head->rlink;
    ~~while(cur!=head)~~
```

```
while(cur!=head)
{
    if(cur→info==key)
    {
        printf("Search Successfull\n");
        return;
    }
    cur=cur→rlink;
}
printf("Search is not successfull\n");
return;
}

NODE ddelete_duplicates(int item, NODE head)
{
    NODE prev, cur, next;
    int count=0;
    if(head→rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    cur=head→rlink;
    while(cur!=head)
    {
        if(cur→info!=item)
        {
            cur=cur→rlink;
        }
        else
        {
            count++;
            if(count==1)
            {
                cur=cur→rlink;
                ~~[illegible]~~ continue;
            }
            else
            {
                prev = cur->llink;
                next = cur->rlink;
                prev->rlink = next;
                next->llink = prev;
                free (cur);
                cur = next;
            }
        }
    }
    if(count==0)
    {   printf("No such ~~[illegible]~~ item found in the list \n");
    else
        printf(" Removed all the duplicate elements of the given item
                successfully \n");
    return head;
}
void display (NODE head)
{
    NODE temp;
    if( head->rlink ==head)
    {
        printf(" The DLL is empty \n");
        return;
    }
    printf(" The contents of the DLL ~~[illegible]~~ are: \n");
    temp = head->rlink;
    while( temp! = head)
    {
        printf("%d\n", temp->info);
        temp= temp->rlink;
    }
    printf(" \n");
}
```

```
void main()
{
    NODE head, last;
    int item, choice;
    head = getnode();
    head->rlink = head;
    head->llink = head;
    for(;;)
    {
      printf("\n1: Insert front\n2: Insert Rear \n3: Delete front \n
            4: Delete rear \n5: Insert_key_left \n 6: Insert_K
            Right \n7: Delete all keys \n8: Search item \n
            9: Delete Duplicate\n10: Display \n11: Exit");
      scanf("%d", &choice);
      switch(choice)
      {
         case 1: printf("Enter the item at front end : ");
                 scanf("%d", &item);
                 last = dinsert_front(item, head);
                 break;
         case 2: printf("Enter the item at rear end: ");
                 scanf("%d", &item);
                 last = dinsert_rear(item, head);
                 break;
         case 3: last = ddelete_~~rear~~front(head);
                 break;
         case 4: last = ddelete_head(head);
                 break;
         case 5: printf(" Enter the key item: ");
                 scanf("%d", &item);
                 head = insert_leftpos(item, head);
                 break;
         case 6: printf(" Enter the key item: ");
                 scanf("%d", &item);
                 head = insert_rightpos(item, head);
                 break;
case 7: printf("Enter the key  item: ");
        scanf("%d", &item);
        head= delete_all_key(item, head);
       break;
case 8: printf(" Enter the  key item: ");
       scanf("%d", &item);
       searching (item, head);
       break;
case 9: printf(" Enter the  key  item:  ");
       scanf("%d", &item);
       head = ddelete_duplicate (item, head);
       break;
case 10: display( head);
       break;
default : return;
}
}
}
```