```
Lab Program - 9

>WAP to implement doubly link list with primitive operations
# include < stdio.h >
                            elifornia = exit (e);
# include < stdlib.h>
                               " breck;
 struct node
 int info;

struct node *rlink;

struct node * llink;
                           ces- 2 : printy ( QUEUE )
  };
typedef struct node *NODE;
  NODE getnode ()
                       : (1) b chart & ("bolt ) france.
      NODE X;
       x = (NODE) malloc (size of (struct node));
       if (z == NULL)
       2 print ("Memory full\n");
          exit (0);
     return x;
                    level - delete-freed frist);
```

```
wid freenode (NODE x)
  & free(x);
NODE dinsert-front (int item, NODE & head)
    NODE temp, cur;
     temp = getnede ();
      temp-info = item;
      temp - llink = NULL;
      temp-rlink = NULL;
   cus = head - rlink;
      head -> rlink = temp;
      temp - link = head;
      temp -> rlink = cur;
      rui -> blink = temps
      return head;
      dinsert_rear (int item, NODE head)
NODE
      NODE temp, eur;
      temp = getrode ();
       temp > info = item;
       temp - slink = NULL;
       temp - rlink = NULL;
       cus = head -> llink;
       head - llink = temp;
       temp - think = head;
       cur -> rlink = temp;
       temp > llink = cui;
    3 return head;
```

```
NODE ddelete-front (NODE head)
      NODE aur, next;
if (head -> rlink == head)
       1 printf (" List is empty In");
        return head;
      cus = head -> rlink;
      next = cur -> rlink;
      head -> rlink = next;
      next - link = head;
                               the front end is: %d\n', cus - wifo);
      printf (" Item deleted at
       free (cur);
       neturn head;
NODE Idelete-rear (NODE head)
     NODE au, prev;
if (head > rlink == head)
     frint (" list is empty \n");
     return head;
    cur = head - llink;
    prev = aur - llink;
                              daile had - has
     prev -> rlink = head;
     head -> llink = prev;
    print (" Item deleted at the rear end is %d)n")
                               cus - info);
                             the still regard
     free (cur);
    return head;
```

```
aduplay (NODE head)
     NODE temp; if (head -) rlink == head)
      printf ("List is empty \n");
    printf (" The contents of the list are : \n");
       temp = head -> rlink;
        while (temp ! = head)
       2 printf (" % din", temp > info);
         temp = temp > solink;
roid deearch (int key, NODE head)
    NODE RUS
    int count;
    if (head -> slink = = head)
      I printf (" list is emply In");
    ru = head -> rlink;
    count=1;
   while (cur! = head &d our - info! = key)
    2 cus = cur - rlink;
      count ++;
   if (cus == head)
      printf l' Search unsuccesfull \n");
    printf (" Key element found at the position " d \n", count);
```

```
NODE dinsert-left pos (int item, NODE head)
    NODE cur, prev, temp;
      if (head -> rlink == head)
      2 prints ("List is empty)n");
       return head;
    cus = head → rlink;
while (cur; = head)
      if (cus -> info = = iten)
         2 break;
      us = cur → rlink;
                               and assumed hith begunso
   if (cur = = head)
   2 printf ("No such item found in the list In");
return head;
   prev = me - link ;
   temp = getrode ();
   temp -> llink = NULL;
   temp -> rlink = NULL;
    printf ("Enter the iten to be inserted at the left of
the given item of n");
   scanf ("%d", & temp - info);
    prev -> rlink = temp;
     temp - link = prev;
temp - runk = run;
cur -> link = temp;
     return head;
and the house to bring the first of the
```

```
NODE dinsert-hightpos (int item, NODE head)
   NODE temp, cur, next; if (head -) which == head)
      Frints (" List is empty \n");
       return head;
    eu = head -> rlink;
   while ( cur! = head)
    if ( cu-) info == item)
        2 break;
     3 cus = cus -> slink;
   if ( un = = head)
  2 printf ("No such itam found in the list \n");
   3 return head;
  next = aur - rlink;
   temp = getrode ();
  temp > llink = NULL;
  prints ("Enter the item to be inserted at the right
   of the given item: \n");
   scanf ( %d", & temp > info);
    cus -> rlink = temp;
    temp - llenk = cus;
    next - llink= temp;
     temp - rlink = next;
    return head;
   3 - all pe stander designed all the 3 fines
```

NODE delete diplicates (int êtem, NODE head) NODE prev, eur, next; int count = 0; if (head -> rlink = = head) 2 printf (" List is empty \n"); return head; icus = land = jour ur = head -> rlink; while (cu ! = head) 2 if (au → info != item) 2 aur= cur -> rlink; else 2 court ++; Line on of fring & if (count == 1) 2 cur = cur -> rlink; continue; else 2 prev = cus -> llink; rest = cus - slink; prer - slink = next; next -> llink = prer; free (cus); cua = next; ignate study + 1200 if (count == 0)
i printf ("No such item found in the list \n"); print ("All the duplicate elements of the given item are removed successfully (n"); return head; 3

NODE delete -all-key lint item, NODE head) NODE prev, aus, next; int count; if (head -> rlink = = head) printf ("List empty"); return head; count =0; cu = head -> rlink; while (aus ! = head) if (eten!= cus -> info) cus = cus -> plink; else 2 count ++; prev = cur -> llink; next = cut -> rlink; prev -> rlink = next; next -> llink = per; felemode (cur); Cus = next; if (court == 0) printf I" Key not found "); printf (" Key found at " lod positions and are deleted)," return head;

```
int main ()
    NODE head;
    ent item, choice, key;
     head = getrode ();
     head -> llik = head;
      head -> rlink = head;
     for (;;)
 print (" m1: dinert-front \n2: dinsert-real \n3: ddelte-fin
1n4° ddelete rear \n 5° ddisplay \n 6° dsearch \n7° dinsex
 leftpos \n 8: dirsert rightpos\n9: delete duplicates
In10: ddelete-based on specified value (n/1: exist \n");
   prints (" Enta the choice \n");
    scarf (" %d", & choice);
    surtch (choice)
      case 1: printf ("Enter the "item at front end: In");

scarf ("% od", & item);

head = dinsert-front (item, head);
     case 2: peintf !" Enter the "item at rear end" In");
               scarf ("%d", & item);
               head = dinsert-lear (item, head);
               break;
       cax 3: head = ddelete-front (head);
                break;
      case 4° head = ddelete - rear (head);
                 break;
       case 5: daisplay (head);
               break;
     case 6: printf ("Enter the key element to be searched in
             scanf ("1.d", & key);
             dsearch (key, head);
            break;
```

```
case 7: print ("Enter the key element " \n");
         scanf (" 7/d; & key);
         head = dinsert_ liftpos (key, head);
         breaks
case 8: print (" Enter the key element: (n");
         scanf ("%d", &key);
         head = dinsert - rightpos (key, head);
         break;
 case 9% printf ("Enter the key element whose duplicates
      should be removed: "\n");
      scanf ("%d", & key);
      head = ddelete - duplicates (key, head);
      break;
 case 10: print ("Enter the key value (n");
       scanf (" %d", & item);
  head = delete - all - key (item, head);
      break;
   case 11: exit (0);
    default: printf ("Invalid choice n");
 return 0;
```