LAB 9
1BM19CS162

```
#include < stdio.h>
#include < stdlib.h>
struct node {
        int info ;
        struct node * rlink;
        struct node * llink;

    };
typedef structnode * NODE;
NODE getnode ()
  {
    NODE x;
   x = (NODE)malloc ( sizeof(struct node));
   if (x==NULL)
     {
       printf ("memoryfull \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 .
```

```
NODE insert_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  dedelete_front (NODE head )
 {
   NODE cur, next ;
   if (head ->rlink ==head )
    {
     printf ("dq empty \n") ;
     return head;
    }
   cur = head ->rlink ;
   next = cur ->rlink ;
   head ->rlink =next ;
   next ->llink=head ;
  printf ("node deleted is %d " , cur->info);
  freenode (cur ) ;
 return head ;
 }
```

```
NODE ddelete_rear (NODE head)
{
    NODE cur , prev ;
    if (head -> rlink == head)
        {
          printf ("dq empty \n ");
        return head ;
        }
      cur = head -> llink ;
      prev = cur -> llink ;
      head -> llink = prev ;
      prev -> rlink = head
     printf (" the node deleted is %d ", cur->info);
     free(cur) ;
     return head ;

    }
void display (node head)
 { NODE temp ;
  if ( head ->rlink == head )
    {
     printf (" dq empty \n ");
     return ;
    }
  printf ( "contents of dq \n ");
  temp = head -> rlink ;
```

```
~~while~~ (~~temp->rlink~~
while ( temp != head)
        printf ("%d \t" , temp->info ) ;
        temp = temp ->rlink ;
        }
        printf ("\n" ) ;
    }
Int length (NODE first ) .
  {
    NODE temp = first ->rlink ;
     int ct = 0 ;
    while (temp != first )
      {
        ct ++ ;
        temp = temp ->rlink ;
      }
    printf ("Length of the list is %d ", ct ) ;
      return ct ;
  }
NODE search ( NODE first )
 {
   NODE temp = first ->rlink ;
   int count = 0, key , flag = 0 ;
  printf (" Enter the key : " ) ;
   scanf (" %d " &key ) ;
 while ( first != ~~first~~ )
   {
        if ( temp ->infor == key)
        {
           flag = 1;
           printf ("key %d found in position %d ",key,
                                 count);
        }
        temp = temp  ->rlink  ;
     }
   if (flag ==0)
     {
        printf (" key is not fount in list " );
     }
   return first ;
}
NODE insert_after (NODE first)
 { int  key ,item, flag=0;
  printf ("Enter the element: " );
  scanf ("%d",&key);
  NODE temp = first->rlink;
  NODE ptr =getnode();
  while (temp  != first)
 {
  ~~pri~~ if(temp ->info == key)
    { printf (" Enter the item need to beinserted: "):
    scanf ("%d", &item);
    ptr  ->info = item;
   ptr->rlink = temp ->rlink;
   ptr->llink = temp;
  temp -> rlink = ptr;
```

```
      flag = 1 ;
      return first ;
    } temp = temp ->rlink ;
   }
   if (flag == 0)
           printf ("There is no such element ");
        return first ;
   }
NODE insert_after (NODE first )
   {
     int key, item, flag = 0 ;
     printf ("Enter the element , " ) ;
     scanf ( "%d", &key) ;
      NODE temp = first ->rlink ;
       NODE  ptr = getnode() ;
       while (temp != first )
      { if (temp ->info == key )
         {
          printf ("Enter the item need to be insetted")
         scanf ("%d", &item ) ;
        ptr -> info = item ;
           ptr -> rlink = temp ;
          ptr -> llink = temp ->llink ;
           ptr ->llink = ptr ;
           frag = 1;
           return first;
        }.
           temp = temp ->rlink ;
      }
    if (flag==0)
       printf ("There is no such element ");
    return first;
   }
   void delete-dup (NODE head)
    {
       NODE cur, temp, ptr, prev;
      if (head->rlink==head)
        {
         printf ("list is empty \n");
          return;
        }
     temp = head->rlink;
    cur = head->rlink;
    while ( cur != head )
      {if ( temp->info = cur->info)
      {
       ptr=cur->rlink; ptr->llink=cur->llink;
       ptr=cur->llink; ptr->rlink=cur->rlink;
       freenode (cur);
     }
   cur=cur->rlink;
   }
  temp =temp->rlink;
 }
 return;
 }
```

```
void main() {
    NODE head , last ;
    int item , choice, len ;
    head = getnode( ) ;
      head ->rlink = head ;
       head ->llink = head ;
     for (;; )
   {
printf ("\n 1 : insert front \n 2 : insert rear \n 3:
        3 : delete ~~foront~~ \n 4 : delete rear \n 5 : display
        \n 6 : exit \n 7 : delete duplicate items \n 8 :
         insert beofore \n 11 : insert after \n" ) ;
  printf ("enter the choice \n" ) ;
   scanf (" %d" , &choice ) ;
  switch (choice ) {
     {
   case 1  : printf ("enter the item at front end \n") ;
            scanf (" %d" , &item ) ;
            last = dinsert - front (item, head ) ;
            break ;
  case 2 : printf ("enter the item at rear end \n")
           scanf (" %d" , &item ) ;
           last = dinsertrear(item, head.
            break ;
  case  3 : last = ddeletefront(head ) ;
            break ;
  case  4 : last = deleterear ~~front~~ (head ) ;
           break ;
case 5 : display( head ) ;
         break;
case 7 : delete_dup (head) ;
         break ;
case 8 : ~~hea~~ len - length(head) ;
         break ;
case 9 : head = search(head) ;
         break ;
case 10 : head = insert_after(head) ;
          break ;
case 11 : head = insert_before(head) ;
          break;
default : break
 }
}
}
```