DS LAB PROGRAM 6 WRITE UP:-

nuthul Raj

LAB PROGRAM 6

```
# include <stdio.h>
# include < conio.h>
struct node
{
int info;
Struct node * link;
};
typedef struct node * NODE;
NODE getnode ()
{
NODE x;
x = (NODE) malloc (sizeof (struct node));
if (x == NULL)
{
printf (" mem full \n");
exit(0);
}
return x;
}
void freenode (NODE x)
{
free(x);
}
NODE insert_front (NODE first, int ite
{
NODE temp;
temp = get node ();
temp-> info = item;
temp -> link = NULL;
if (first == NULL)
return temp;
temp-> link = first
first = temp;
```

```
temp -> link = first;
first = temp;
return first;
}
NODE delete_front (NODE first)
{
NODE temp;
if (first == NULL)
~~return temp~~
~~temp -> link = first;~~
~~first = item~~
{
printf("list is empty cannot delete\n");
return first;
}
temp = first;
temp = temp -> link;
printf ("item deleted at frond-end
is = %d\n", first -> info);
free(first);
return temp;
}
NODE insert_rear ( NODE first , int
item)
{
NODE temp, cur;
temp = getnode();
temp -> info = item;
temp -> link = NULL;
if ( first == NULL)
return temp;
cur = first;
while ( cur -> link != NULL)
```

```
cur = cur->link,
return first;
}
NODE delete_rear(NODE first)
{
NODE cur, prev
if (first == NULL)
{
printf(" list is empty cannot delete \n"),
return first,
}
if (first -> link == NULL)
{
printf(" item deleted is %d \n", first->info
);
free (first);
return NULL,
}
prev= NULL,
cur= first;
while (cur-> link != NULL)
{
prev= cur;
cur= cur-> link,
}
printf (`item delete at rear is %d', cur
-> info);
free cur;
prev-> link= NULL,
  return first;
}
NODE delete_info (int key, NODE first)
```

```
{
NODE prev, cur;
if(first == NULL)
{
printf("list is empty \n");
return NULL;
}
if(key == first->info)
{
cur = first;
first = first->link;
freenode (cur);
return first;
}
prev = NULL;
cur = first;
while(cur != NULL)
{
if(key == cur->info) break;
prev = cur;
cur = cur -> link;
}
if (cur == NULL)
{
printf("sear is unsccessfull \n");
return first;
}
prev -> link = cur-> link;
printf("key deleted is %d" cur-> info);
freenode(cur);
return first;
}
void display(NODE first)
```

```
{
NODE temp;
if (first == NULL) {
printf ("list empty cannot display items");
printf (" contents of lists : \n");
for (temp = first ; != NULL ; temp = temp ->
link )
{
printf("%d\n", temp -> info );
  }
}
void main ()
{
int item , choice , key ;
NODE first = NULL ;
 for( ; ; )
{
printf ("\n 1 : Insert_front \n 2 Delete_
 front \n 3 : Insert_rear \n 4 Delete_
 rear\n") ;
printf("5 delete_info \n 6 Display_list
\n 7 exit\n") ;
printf ("enter the choice \n") ;
Scanf ("%d", & choice) ;
Switch (choice)
{
case 1 : printf ("enter the item at
   front end\n") ;
   Scanf ("%d", &item) ;
   first = insert_front (first, item) ;
   break
Case 2 : first = delete _front (first) ;
         break ;
}

Case 3: printf ("enter the item at rear
 -end \n");
  Scanf ("%d", &item);
  first = Insert-rear (first, item);
  break;
Case4: first = delete-rear (first);
        break;
Case 5: printf (" enter the key to be
       deleted \n");
      Scanf ("%d", &key);
     first = delete-info (key, first);
      break;
Case 6: display (first);
          break;
default: exit (0)
        break;
   }
  }
 }
```