DS LAB PROGRAM 9 WRITE UP

nikhil Ray

lab program (9)

```
# include <stdio.h>
# include <stdlib.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("mem 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 - temp,
temp ->rlink = head,
cur ->llink = 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->r link = head,
temp->l link = cur,
cur-> l link = temp,
return head,
}
NODE ddelete_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("the 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);
freenode (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 != head)
{
```

```
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\n4. delete rear \5. display
\n6: exit \n"),
printf
scanf
switch (choice)
{
case1: printf ("enter the item at front\n
        scanf ("%d", &item),
      last = dinsert_front (item, head),
        break,
case2 : printf ("enter the item at rear\n
        scanf ("%d", &item),
      last = dinsert _rear (item, head),
        break
case3 : last = ddelete:front (head),
        break,
case4 : last = ddelete_rear (head),
        break,
case 5 display (head),
        break;
    default : exit (0),
  } }
}
```