



*Untitled - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.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=head;
    temp->rlink=cur;
    cur->llink=temp;
    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)
```



*Untitled - Notepad

File Edit Format View Help

```
}
```

```
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;
```

```
}
```



```
NODE insert_leftpos(int item,NODE head)
```

```
{
```

```
NODE temp,cur,prev;
```

```
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)
```

```
{
```

*Untitled - Notepad

File Edit Format View Help

```
if(cur==head)
{
    printf("key not found\n");
    return head;
}
prev=cur->llink;
printf("enter towards left of %d=%d",item);
temp=getnode();
scanf("%d",&temp->info);
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;
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;
printf("enter towards right of %d=%d",item);
temp=getnode();
scanf("%d",&temp->info);
cur->rlink=temp;
temp->llink=cur;
next->llink=temp;
temp->rlink=next;
return head;
}
NODE search(NODE head,int item)
{
NODE temp,cur;
int flag=0;
if(head->rlink==head)
```

*Untitled - Notepad

File Edit Format View Help

```
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
    flag=1;
    break;
}
cur=cur->rlink;
}
if(cur==head)
printf("search unsuccessfull\n");
if(flag==1)
printf("search successfull\n");
}
NODE delete_all_key(int item,NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head)
    {
        printf("list empty\n");
        return head;
    }
    count=0;
    cur=head->rlink;
    while(cur!=head)
    {
        if(item!=cur->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("not found\n");
    else{
        printf("found at %d positions and are deleted" ,count);
    }
}
```

*Untitled - Notepad

```
File Edit Format View Help
printf("found at %d positions and are deleted",count);
    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("%d",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
//clrscr();
for(;;)
{
    printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of key element\n6:insert right of key element\n7:search\n8:delete repeating occurrences\n9:display\n10: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);
            last=dinsert_front(item,head);
            break;
        case 2: printf("enter the item at rear end\n");
            scanf("%d",&item);
            last=dinsert_rear(item,head);
            break;
        case 3:last=ddelete_front(head);
            break;
        case 4: last=ddelete_rear(head);
            break;
    }
}
```

*Untitled - Notepad

File Edit Format View Help

```
head=getnode();
head->rlink=head;
head->llink=head;
//clrscr();
for(;;)
{
    printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert left of key element\n6:insert right of key element\n7:search\n8:display\n9: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);
            last=dinsert_front(item,head);
            break;
        case 2: printf("enter the item at rear end\n");
            scanf("%d",&item);
            last=dinsert_rear(item,head);
            break;
        case 3:last=ddelete_front(head);
            break;
        case 4: last=ddelete_rear(head);
            break;

        case 5:
            printf("enter the key element\n");
            scanf("%d",&item);
            last=insert_leftpos(item,head);
            break;
        case 6:
            printf("enter the key element\n");
            scanf("%d",&item);
            last=insert_rightpos(item,head);
            break;
        case 7:
            printf("enter the search element\n");
            scanf("%d",&item);
            search(head,item);
            break;
        case 8: printf("enter element to be deleted\n");
            scanf("%d",&item);
            last=delete_all_key(item,head);
            break;
        case 9: display(head);
            break;
        default:exit(0);
    }
}
getch();
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
```

I

```
1
enter the item at front end
34
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
```

```
enter the choice
```

2

```
enter the item at rear end
54
```

```
1:insert front
2:insert rear
```

enter the item at rear end

54

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit

enter the choice

2

enter the item at rear end

54

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit

enter the choice

8

enter element to be deleted

54

found at 2 positions and are deletedcontents of dq

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
8
enter element to be deleted
54
found at 2 positions and are deletedcontents of dq
34

1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert left of key element
6:insert right of key element
7:search
8:delete repeating occurrences
9:display
10:exit
enter the choice
9
contents of dq
34
```

LAB PROGRAM - 9

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.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)
    {
        fprintf (stderr, "mem full\n");
        exit (0);
    }
    return x;
}

void freeNode (NODE x)
{
    free (x);
}

NODE insert (int item, NODE head)
{
    NODE temp, cur;
    temp = getNODE ();
    temp->info = item;
    temp->link = head;
```

cur = head \rightarrow rlink;
head \rightarrow rlink = temp;
temp \rightarrow llink = head;
temp \rightarrow rlink = cur;
cur \rightarrow llink = temp;
return head;

}

NODE insert(node item, NODE head)

{ NODE temp; cur;
temp = getNode();
temp.info = item;
cur = head \rightarrow llink;
head \rightarrow llink = temp;
temp \rightarrow rlink = head;
temp \rightarrow llink = cur;
cur \rightarrow rlink = temp;

return head;

}

NODE delete(node head)

{

NODE cur, next;

if (head \rightarrow rlink == head)

{

cout \ll ("the list is empty \n");

return head;

}

cur = head \rightarrow rlink;

next = cur \rightarrow rlink;

head \rightarrow rlink = next;

next \rightarrow llink = head;

cout \ll ("the node deleted is : " \ll cur.info);

free node (cur);

return head; } }

NODE delete (NODE head)

{ NODE cur, chrd;

if (head == rlink == chrd)

{ print ("deleting empty node"); }

return head;

} cur = head \rightarrow llink;

your = cur \rightarrow llink;

chrd \rightarrow llink = chrd;

chrd \rightarrow rlink = chrd;

print ("The node deleted is .d, cur = info);

free node (cur);

return head;

}

NODE insert - leftpos (int info, NODE head)

{ NODE temp, cur, chrd;

if (head \rightarrow rlink == chrd)

{ print ("list empty\n"); }

return head;

}

cur = head \rightarrow rlink;

while (cur != head)

{

if (item == (cur \rightarrow info) break;

cur = cur \rightarrow rlink;

}

if (cur == head)

{

if (key == cur->key) {
 cout << "Key found\n";
 return head;

{

else if (key < cur->key) {

cout << "Search left of " << cur->key << endl;

temp = getnode();

if (temp == NULL) {

cout << "No memory available\n";

exit(1);

cur = temp;

temp = cur->left;

return head;

{

else if (key > cur->key) {

{

NODE temp; cur, next;

if (cur->right == NULL)

{

cout << "List empty\n";

return head;

{

cur = cur->right;

while (cur != NULL)

{

if (item == cur->info) break;

cur = cur->right;

{

if (cur == NULL)

{

cout << "Key not found\n";

return head;

{

next = cur \rightarrow rlink;

prev f ("onto toward right of .l.d = ", item);
 temp = getnode();

Scan f (" / .d ", & temp \rightarrow info);

cur \rightarrow rlink = temp;

temp \rightarrow llink = cur;

next \rightarrow llink = temp;

temp \rightarrow rlink = next;

creation done,

}

NODE search (NODE head, int item)

{

NODE temp, cur;

int flag = 0;

if (head \rightarrow rlink == head)

}

cur = cur \rightarrow rlink;

}

if (cur == head)

printf ("Search unsuccessful \n");

if (flag == 1)

printf ("Search successful \n");

}

NODE delete all - Key (int item, NODE head)

{

NODE prev, cur, next;

int cur;

if (head \rightarrow rlink == head)

{

printf ("list is empty. \n");

creation done;

}

count = 0;
cur = head \rightarrow rlink;
while (cur != head)

2
if (item != cur->info)
 cur = cur \rightarrow rlink;
 else

3 Count++;
 prev = cur \rightarrow llink;
 next = cur \rightarrow rlink;
 prev \rightarrow rlink = next;
 next \rightarrow llink = prev;
 decrement (cur);
 cur = next;

if (Count == 0)
 printf ("not found\n");
else

 printf ("found at %d position in %d", Count, head);
 return head;

2

3
void display (nodehead)

{

 Node *temp;

 if (head \rightarrow rlink == head)

{

 printf ("empty list\n");
 return;

3

 printf ("contents of list\n");
 temp = head \rightarrow rlink;
 while (temp != head)

{ priority C " / d ", front \Rightarrow info);
front = front \Rightarrow rlink;

}
front \neq C " / n ");

}
void main ()

Node head, last;

int item, choice;

choice = getChoice();

choice \Rightarrow rlink \rightarrow head;

((drScr ());

choice ());

if print of C " / n : insert front \n 2: insertion \n 3: delete

front \n 4: delete head; 5. insert left of key

element \n 6: insert right of key above \n 7: search \n

8: delete separating occurred \n 9: display \n 10: exit \n);

Case 1: print of C " onto the item at front or C " / n ");

strcmp (" / d ", & item);

choice = insert front (item, choice,

break;

Case 2: print of C " onto the item after head \n);

strcmp (" / d ", & item);

choice = insert after head (item, choice,

break;

Case 3: last = delete - front (choice);

break;

Case 4: last = delete - rear (choice);

break;

Case 5:

printf ("enter the key element \n");

Scny C". / .d", & item);

last = insert ~~at top~~ item, choice;

break;

Case 6:

printf ("enter the key element \n");

Scny C". / .d", & item);

last = insert ~~at top~~ item, choice;

break;

Case 7:

printf ("enter the search element \n");

Scny C". / .d", & item);

Search (choice, item);

break;

Case 8: printf ("enter element to be delete \n");

Scny C". / .d", & item);

last = delete - cell - key (item, choice);

Case 9: display (choice);

break;

default : exit(0);

}

}

getch();

}