

Kab Program - 9

q) Write a program to implement doubly linked list with following operations:

- i) Insert front iv) Delete rear
- ii) Insert rear v) Display
- iii) Delete front vi) Insert after/ before key node
- vii) Search
- viii) Delete duplicate

A) #include < stdio.h >

#include < conio.h >

#include < process.h >

#include < std.lib.h >

#include < malloc.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");

exit(0);

}

return x;

}

void freenode(NODE x) {

free(x);

}

NODE insert_front(int item, NODE head) {

NODE temp, cur;

```
temp = getnode();
temp->info = item;
cur = head->glink;
head->glink = temp;
temp->llink = head;
temp->glink = cur;
cur->llink = temp;
return head;
```

{

```
NODE insert_gear(int item, NODE head) {
```

```
NODE temp, cur;
temp = getnode();
temp->info = item;
cur = head->llink;
head->llink = temp;
temp->glink = head;
temp->llink = cur;
cur->glink = temp;
return head;
```

{

```
NODE delete_front(NODE head) {
```

```
NODE cur, next;
if (head->glink == head)
    printf("dq empty\n");
return head;
```

{

```
cur = head->glink;
next = cur->glink;
head->glink = next;
next->llink = head;
```

```
printf("the node deleted is %d", cur->info);
free(node(cur));
```

```
return head;
```

{

```
NODE delete_near(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 r. 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(" r. d ", temp->info);
```

```
        temp = temp->rlink;
```

```
}
```

```
    printf(" \n");
```

```
}
```

```
NODE delete_all_keys(int item, NODE head){
```

```
    NODE prev, cur, next;
```

```
    int count;
```

```
    if(head->rlink == head){
```

```
        printf(" list Empty \n");
```

```
        return head;
```

```
}
```

```
count = 0;  
cur = head -> glink;  
cur = cur -> glink;  
while (cur != head) {  
    if (item != cur -> info) {  
        cur = cur -> glink;  
    } else {  
        count++;  
        forev = cur -> llink;  
        nextf = cur -> rlink;  
        forev -> rlink = nextf;  
        nextf -> llink = forev;  
        freeNode (cur);  
        cur = nextf;  
    }  
}
```

```
if (count == 0)  
    printf ("key not found\n");  
else  
    printf ("key found at %d positions and are deleted\n", count);  
return head;
```

```
g  
NODE insert_left (int item, NODE head) {  
    NODE temp, cur, forev;  
    if (head -> glink == head) {  
        printf ("Empty\n");  
        return head;  
    }
```

```
    cur = head -> glink;  
    while (cur != head) {  
        if (item == cur -> info) break;  
        cur = cur -> glink;
```

```
    if (cur == head)  
        printf ("key not found\n");
```

return head;
}

prev = cur -> llink;
printf("Enter towards left of %d: ", item);
temp = getnode();
scanf("%d", &temp->info);
prev -> llink = temp;
temp -> llink = prev;
cur -> llink = temp;
temp -> rlink = cur;
return head;

3
NODE insert_right(int item, NODE head){
NODE temp, cur, prev;
if(head -> llink == head){
printf("Empty \n");
return head;

cur = head -> llink;
while(cur != head){
if(item == cur -> info), break;
cur = cur -> llink;

if(cur == head){
printf("key not found \n");
return head;

prev = cur -> rlink;
printf("Enter towards right of %d: ", item);
temp = getnode();
scanf("%d", &temp->info);
prev -> rlink = temp;
temp -> rlink = prev;
cur -> rlink = temp;

```
temp->link = cur;
return head;

}

NODE search(NODE head) {
    NODE temp = head->link;
    int count = 0, key, flag = 0;
    printf("Enter the key : ");
    scanf("%d", &key);
    while (temp != head) {
        count++;
        if (temp->info == key) {
            flag = 1;
            printf("key %d found in position %d", key, count);
        }
        temp = temp->link;
    }
    if (flag == 0)
        printf("key is not found in list");
    return head;
}

int main() {
    NODE head, last;
    int item, choice, option;
    head = getnode();
    head->link = head;
    head->llink = head;
    system("cls");
    for (;;) {
        printf("In 1: insert front In 2: insert rear In 3: delete front In
4: delete rear In 5: display In 6: Delete duplicate occurrences In 7:
search In 8: Insert node after/before key node In 9: Exit In ");
        printf("enter the choice In ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                item = getitem();
                insertfront(&head, item);
                break;
            case 2:
                item = getitem();
                insertright(&head, item);
                break;
            case 3:
                deletefront(&head);
                break;
            case 4:
                deleterear(&head);
                break;
            case 5:
                display(head);
                break;
            case 6:
                deleteDuplicates(&head);
                break;
            case 7:
                key = getitem();
                search(&head, key);
                break;
            case 8:
                item = getitem();
                pos = getpos();
                insertAfterBefore(&head, item, pos);
                break;
            case 9:
                exit(0);
        }
    }
}
```

switch (choice) {

case 1: printf (" enter the item at front end \n"),
scanf ("%d", &item);
last = insert_front (item, head);
break;

case 2: printf (" enter the item at rear end \n"),
scanf ("%d", &item);
last = insert_rear (item, head);
break;

case 3: last = delete_front (head);
break;

case 4: last = delete_rear (head);
break;

case 5: display (head);
break;

case 6: printf (" enter the item \n"),
scanf ("%d", &item);
last = delete_all_key (item, head);
break;

case 8: printf (" press 1 : for insert behind 2 : for insert after \n"),
scanf ("%d", &option);
if (option == 1) {
printf (" enter # key node \n");
scanf ("%d", &item);
last = insert_left_pos (item, head); }
else if (option == 2) {
printf (" enter key node \n");
scanf ("%d", &item);
last = insert_right_pos (item, head); }
break;
}

case 7: search (head);
break;

default: exit(0);

```
    }  
    getch();  
    return (0);  
}
```



TDM-GCC 4.9.2 64-bit Release

(globals)

```
doubly_linked_lists.cpp
1 #include<stdio.h>
2 #include<conio.h>
3 #include<process.h>
4 #include<stdlib.h>
5 #include <malloc.h>
6 struct node
7 {
8     int info;
9     struct node *llink;
10    struct node *rlink;
11 };
12 typedef struct node *NODE;
13
14
15 NODE getnode()
16 {
17     NODE x;
18     x=(NODE)malloc(sizeof(struct node));
19     if(x==NULL)
20     {
21         printf("mem full\n");
22         exit(0);
23     }
24     return x;
25 }
26
27
28 void freenode(NODE x)
29 {
30     free(x);
31 }
32
33
34 NODE insert_front(int item, NODE head)
35 {
36     NODE temp, cur;
37     temp=getnode();
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035

Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



TDM-GCC 4.9.2 64-bit Release

```
doubly_linked_lists.cpp
37 |     temp=get_node();
38 |     temp->info=item;
39 |     cur=head->rlink;
40 |     head->rlink=temp;
41 |     temp->llink=head;
42 |     temp->rlink=cur;
43 |     cur->llink=temp;
44 |     return head;
45 |
46 |
47
48 NODE di nsert _rear(int item, NODE head)
49 {
50     NODE temp, cur;
51     temp=get_node();
52     temp->info=item;
53     cur=head->llink;
54     head->llink=temp;
55     temp->rlink=head;
56     temp->llink=cur;
57     cur->rlink=temp;
58     return head;
59 }
60
61
62 NODE dde let e_fron t(NODE head)
63 {
64     NODE cur, next;
65     if(head->rlink==head)
66     {
67         printf("dq empty\n");
68         return head;
69     }
70     cur=head->rlink;
71     next=cur->rlink;
72     head->rlink=next;
73     next->llink=head;
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds

Type here to search



18:29 ENG 15-12-2020 4



(globals)

```
doubly_linked_lists.cpp

72     head->rlink=next;
73     next->llink=head;
74     printf("the node deleted is %d", cur->info);
75     freenode(cur);
76     return head;
77 }

78
79
80 NODE ddelete_rear(NODE head)
81 {
82     NODE cur, prev;
83     if(head->rlink==head)
84     {
85         printf("dq empty\n");
86         return head;
87     }
88     cur=head->llink;
89     prev=cur->llink;
90     head->llink=prev;
91     prev->rlink=head;
92     printf("the node deleted is %d", cur->info);
93     freenode(cur);
94     return head;
95 }

96
97
98 void display(NODE head)
99 {
100    NODE temp;
101    if(head->rlink==head)
102    {
103        printf("dq empty\n");
104        return;
105    }
106    printf("contents of dq\n");
107    temp=head->rlink;
108    while(temp!=head)
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

doubly_linked_lists.cpp

```
109 {  
110     printf ("%d ", temp->info);  
111     temp=temp->rlink;  
112 }  
113     printf ("\n");  
114 }  
115  
116  
117 NODE delete_all_key(int item, NODE head)  
118 {  
119     NODE prev, cur, next;  
120     int count;  
121     if (head->rlink==head)  
122     {  
123         printf ("List Empty");  
124         return head;  
125     }  
126     count=0;  
127     cur=head->rlink;  
128     cur=cur->rlink;  
129     while (cur!=head)  
130     {  
131         if (item==cur->info)  
132             cur=cur->rlink;  
133         else  
134         {  
135             count++;  
136             prev=cur->llink;  
137             next=cur->rlink;  
138             prev->rlink=next;  
139             next->llink=prev;  
140             freenode(cur);  
141             cur=next;  
142         }  
143     }  
144     if (count==0)  
145         printf ("key not found");
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

```
doubly_linked_lists.cpp
146     else
147     printf("key found at %d positions and are deleted\n", count);
148
149     return head;
150
151
152 }
153 NODE insert_left pos(int item, NODE head){
154     NODE temp, cur, prev;
155     if(head->rlink==head){
156         printf("Empty \n");
157         return head;
158     }
159     cur=head->rlink;
160     while(cur!=head){
161         if(item==cur->info)break;
162         cur=cur->rlink;
163     }
164     if(cur==head){
165         printf("key nt found \n");
166         return head;
167     }
168     prev=cur->llink;
169     printf("Enter towards left of %d:", item);
170     temp=get node();
171     scanf ("%d", &temp->info);
172     prev->rlink=temp;
173     temp->llink=prev;
174     cur->llink=temp;
175     temp->rlink=cur;
176     return head;
177 NODE insert_right pos(int item, NODE head){
178     NODE temp, cur, prev;
179     if(head->llink==head){
180         printf("Empty \n");
181         return head;
182 }
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



18:29 ENG 15-12-2020 4



TDM-GCC 4.9.2 64-bit Release

(globals)

```
152 L }
153 NODE insert_left(int item, NODE head){
154     NODE temp, cur, prev;
155     if(head->rlink==head){
156         printf("Empty \n");
157         return head;
158     }
159     cur=head->rlink;
160     while(cur!=head){
161         if(item==cur->info)break;
162         cur=cur->rlink;
163     }
164     if(cur==head){
165         printf("key not found \n");
166         return head;
167     }
168     prev=cur->llink;
169     printf("Enter towards left of %d:", item);
170     temp=getnode();
171     scanf("%d", &temp->info);
172     prev->rlink=temp;
173     temp->llink=prev;
174     cur->llink=temp;
175     temp->rlink=cur;
176     return head;
177 NODE insert_right(int item, NODE head){
178     NODE temp, cur, prev;
179     if(head->llink==head){
180         printf("Empty \n");
181         return head;
182     }
183     cur=head->llink;
184     while(cur!=head){
185         if(item==cur->info)break;
186         cur=cur->llink;
187     }
188     if(cur==head){
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

doubly_linked_lists.cpp

```
187 }
188 if (cur == head){
189     printf("key not found \n");
190     return head;
191 }
192 prev=cur->rlink;
193 printf("Enter towards right of %d: ", item);
194 temp=get_node();
195 scanf("%d", &temp->info);
196 prev->llink=temp;
197 temp->rlink=prev;
198 cur->rlink=temp;
199 temp->llink=cur;
200 return head; }

201
202
203 NODE search( NODE head)
204 {
205     NODE temp=head->rlink;
206     int count=0, key, flag=0;
207     printf("Enter the key : ");
208     scanf("%d", &key);
209     while(temp!=head)
210     {
211         count++;
212         if (temp->info==key)
213         {
214             flag=1;
215             printf("key %d found in position %d", key, count);
216             temp=temp->rlink;
217         }
218         if (flag==0)
219         {
220             printf("Key is not found in list");
221         }
222     }
223 }
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



18:30 ENG 15-12-2020 4



doubly_linked_lists.cpp

```
223 L }
224
225
226 int main()
227 {
228     NODE head, last;
229     int item, choice, option;
230     head=get_node();
231     head->rlink=head;
232     head->llink=head;
233     system("cls");
234     for(;;)
235     {
236         printf("\n1: insert front\n2: insert rear\n3: delete front\n4: delete rear\n5: display\n6: Delete repeating occurrences\n7: search\n 8:");
237         printf(" enter the choice\n");
238         scanf("%d", &choice);
239         switch(choice)
240         {
241             case 1: printf("enter the item at front endl");
242             scanf("%d", &item);
243             last=dinsert_front(item,head);
244             break;
245             case 2: printf("enter the item at rear endl");
246             scanf("%d", &item);
247             last=dinsert_rear(item,head);
248             break;
249             case 3: last=ddel ete_front(head);
250             break;
251             case 4: last=ddel ete_rear(head);
252             break;
253             case 5: display(head);
254             break;
255             case 6: printf("enter the item \n");
256             scanf("%d", &item);
257             last=delete_all_key(item,head);
258             break;
259             case 8: printf("press 1:for insert behind    2: for insert after");
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:30 ENG 15-12-2020 4

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.cpp - Dev-C++ 5.11
- Menu Bar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help
- Toolbar:** Includes icons for file operations like Open, Save, Print, and Build.
- Compiler Status:** TDM-GCC 4.9.2 64-bit Release
- Code Editor:** The main window displays the C++ source code for "doubly_linked_lists.cpp". The code handles various operations on a doubly linked list, including insertion at front/rear, deletion from front/rear, displaying the list, deleting repeating occurrences, and searching for items. It uses functions like dinsert_front, dinsert_rear, ddelte_front, ddelte_rear, display, delte_all_key, and insert_left_pos/right_pos.
- Compiler Tab:** Shows the status: Done parsing in 0.016 seconds.
- Search Bar:** Type here to search
- System Icons:** Taskbar icons for File Explorer, Mail, Settings, Task View, Google Chrome, and DEV.
- System Status:** 18:30, ENG, 15-12-2020, battery level 4.

```
233 | system("cls");
234 | for(;;)
235 | {
236 |     printf("\n1: insert front\n2: insert rear\n3: delete front\n4: delete rear\n5: display\n6: Delete repeating occurences\n7: search\n 8:");
237 |     printf("enter the choice\n");
238 |     scanf("%d", &choice);
239 |     switch(choice)
240 |     {
241 |         case 1: printf("enter the item at front end\n");
242 |             scanf("%d", &item);
243 |             last=dinsert_front(item,head);
244 |             break;
245 |         case 2: printf("enter the item at rear end\n");
246 |             scanf("%d", &item);
247 |             last=dinsert_rear(item,head);
248 |             break;
249 |         case 3: last=ddelte_front(head);
250 |             break;
251 |         case 4: last=ddelte_rear(head);
252 |             break;
253 |         case 5: display(head);
254 |             break;
255 |         case 6: printf("enter the item \n");
256 |             scanf("%d", &item);
257 |             last=delte_all_key(item,head);
258 |             break;
259 |         case 8: printf("press 1:for insert behind      2: for insert after");
260 |             scanf("%d", &option);
261 |             if(option==1){
262 |                 printf("enter key node\n");
263 |                 scanf("%d", &item);
264 |                 last=insert_left_pos(item,head); }
265 |             else if(option==2){
266 |                 printf("enter key node\n");
267 |                 scanf("%d", &item);
268 |                 last=insert_right_pos(item,head);
269 |             break;
```



(globals)

doubly_linked_lists.cpp

```
244     break;
245     case 2: printf("enter the item at rear end\n");
246     scanf("%d", &item);
247     last = insert_rear(item, head);
248     break;
249     case 3: last = delete_front(head);
250     break;
251     case 4: last = delete_rear(head);
252     break;
253     case 5: display(head);
254     break;
255     case 6: printf("enter the item \n");
256     scanf("%d", &item);
257     last = delete_all_key(item, head);
258     break;
259     case 8: printf("press 1:for insert behind    2: for insert after");
260     scanf("%d", &option);
261     if(option==1){
262     printf("enter key node\n");
263     scanf("%d", &item);
264     last = insert_left_pos(item, head); }
265     else if(option==2){
266     printf("enter key node\n");
267     scanf("%d", &item);
268     last = insert_right_pos(item, head);
269     break;
270     }
271     case 7: search(head);
272     break;
273     default: exit(0);
274   }
275   getch();
276   return(0);
277 }
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds

Type here to search



18:30 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
34

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
55

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
67

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
34

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
55

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
67

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
67 55 34 9 13
```

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurrences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
7
Enter the key :55
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
7
Enter the key :55
key 55 found in position 2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
33

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
1

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
1 67 55 34 9 13 33

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
3
the node deleted is 1
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
4
the node deleted is 33
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
67 55 34 9 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
9 67 55 34 9 13 9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
6
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
9:Exit
enter the choice
6
enter the item
9
key found at 2 positions and are deleted

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
9 67 55 34 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
8
press 1:for insert behind    2: for insert after1
enter key node
67
Enter towards left of 67:12
Enter the key :12
key 12 found in position 2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
enter the choice
8
press 1:for insert behind    2: for insert after1
enter key node
67
Enter towards left of 67:12
Enter the key :12
key 12 found in position 2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
9 12 67 55 34 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
9
```

Process exited after 150.6 seconds with return value 0
Press any key to continue . . .

