



## Double linked list

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node * rlink;
```

```
    struct node * llink;
```

```
};
```

```
typedef struct node * NODE;
```

```
NODE getnode() {
```

```
    NODE x;
```

```
    x = (NODE) malloc (sizeof (struct  
node));
```

```
    if (x == NULL)
```

```
{
```

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

```
        exit(0);
```

```
}
```

```
    return x;
```

```
}
```

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

```
    NODE temp, cur;
```

```

temp → info = item;
cur = head → rlink;
head → rlink = temp;
temp → llink = head;
temp → rlink = cur;
cur → llink = 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 item deleted is %d\n",

```

```

        cur);

```

```

    return head;
}

```

```

}

```

```

NODE ddelete-rear (NODE head)

```





{

NODE cur, prev;

if (head → link == head)

{

printf ("dq empty\n")

return head;

}

cur = head → link;

prev = cur → link;

head → link = prev;

prev → link = head;

printf ("The item deleted is %d\n",  
cur → info);

free (cur);

return head;

}

NODE \* search (NODE head, int key,

int z) {

NODE cur, prev, temp;

int f = 0, c = 1;

if (head → link == head)

{

printf ("list empty\n");

return head;

}

```
if (cur → info == key) {
```

```
    f = 1;
```

```
    break;
```

```
}
```

```
cur = cur → rlink;
```

```
c++;
```

```
}
```

```
if (f == 1 & & z == 0) {
```

```
    printf("Search Successful, found at  
index %d\n", c);
```

```
    return head;
```

```
}
```

```
if (f == 1 & & z == 1) {
```

```
    prev = cur → llink;
```

```
    printf("Enter towards left of %d  
= ", key);
```

```
    temp = get node();
```

```
    scanf("%d", &temp → info);
```

```
    prev → llink = temp;
```

```
    temp → llink = prev;
```

```
    cur → llink = temp;
```

```
    temp → rlink = cur;
```

```
    return head;
```

```
}
```

```
if (f == 1 & & z == 2) {
```





```
prev = cur;  
cur = cur → link;  
printf("Enter towards right of %d = key);  
temp = get node ();  
scanf("%d", &temp → info);  
prev → rlink = temp;  
temp → link = prev;  
cur → llink = temp;  
temp → rlink = temp;  
return head;
```

```
}  
printf("Search Unsuccessful\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 ++
```

```
    prev = cur → llink;
```

```
next = cur → x link;
```

```
prev → x link = next;
```

```
next → link = prev;
```

```
free(cur);
```

```
cur = cur;
```

```
cur = next;
```

```
}
```

```
{
```

```
if (count == 0)
```

```
printf("Key not found\n");
```

```
else
```

```
printf("Key found at %d post  
& deleted\n", count);
```

```
return head;
```

```
}
```

```
void display(NODE head)
```

```
{
```

```
    NODE temp;
```

```
    if (head → x link == head)
```

```
    {
```

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

```
        return;
```

```
    }
```

```
    printf("Contents of dg\n");
```

```
    temp = head → x link;
```





```
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;
```

```
    for (;;) {
```

```
        printf ("Enter choice: \n 1. Insert front  
        \n 2. Delete front \n 3. Insert rear  
        \n 4. Delete rear \n 5. Simple  
        search \n 6. Insert left of  
        key \n 7. Insert right of  
        key \n 8. Delete all occurrences  
        of key \n 9. Display \n --  
        Any other key to exit -- \n")
```

```

switch (choice) {
    case 1: printf("Enter the item at
front end\n");
scanf("%d", &item);
head = insert-front(item, head);
break;
    case 3: printf("enter the item at
rear end\n");
scanf("%d", &item);
head = insert-rear(item, head);
break;
    case 2: head = delete-front(head);
break;
    case 4: head = delete-rear(head);
break;
    case 5: printf("Enter key\n");
scanf("%d", &item);
head = search(head, item, 1);
break;
    case 7: printf("Enter key\n");
scanf("%d", &item);
head = search(head, item, 2);
break;
}

```