```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
    struct node *prev;
};
typedef struct node *Node;
Node getNode() {
    Node x;
    x = (Node) malloc (sizeof (struct node));
    if (x == NULL) {
        Pf ("Mem full \n");
        exit (0);
    }
    return x;
}
```

```
Node insertFront(int item, Node head){
    Node temp, cur;
    temp = getNode();
    temp -> data = item;
    cur => head -> next
    head -> next = temp;
    temp -> prev = head;
    temp -> next = cur;
    cur -> prev = temp;
    return head;
}

Node insertRear(int item, Node head){
    Node temp, cur;
    temp = getNode();
    temp -> data = item;
    cur = head -> prev;
    temp -> next = head;
    temp -> Prev = cur;
    cur -> next = temp;
    head -> prev = temp;
    return head;
}
```

```
Node deleteFront (Node head) {
    Node cur, next;
    if (head -> next == head) {
        pf ("Empty \n");
        return head
    }
    cur = head -> next;
    next = cur -> next;
    head -> next = next;
    next -> prev = head;
    pf ("deleld node with data %d", cur -> data)
    free (cur);
    return head;
}
Node deleteRear (Node head) {
    Node cur, prev;
    if (head -> next == head) {
        pf ("List empty \n");
        return head;
    }
    cur = head -> prev;
    prev = cur -> prev;
    head -> prev = prev;
    prev -> next = head;
```

```
pf(" deleted node with data : %d", cur->data);
free(cur);
return head;

}

Node insertBeforeKey(int item, int Key, Node
        Node temp, cur;          head) {
    temp = getNode();
    temp -> data = item;
    temp -> next = NULL;
    temp -> prev = NULL;
    cur = heade -> next;
    while ( cur != head ) {
        if (cur -> data == key)
            break;
        cur = cur -> next;
    }
    if ( cur == head ) {
        pf ( " item not found \n");
        return head
}

cur -> prev -> next = temp;
 temp -> prev = cur -> prev;
 cur -> prev = temp;
 temp -> next = cur ; return head; }
```

```
Node insertafterKey (int item, int key, Node head)
    Node temp, cur;
    temp = getNode();
    temp -> data = item;
    temp -> next = NULL;
    temp -> prev = NULL;
    cur = head -> next;
    while (cur != head) {
        if (cur -> data == key) {
            break;
        }
        cur = cur -> next;
    }
    if (cur == head) {
        Pf ("key not found \n");
        return head;
    }
    cur -> next -> prev = temp;
    temp -> next = cur -> next;
    cur -> next = temp;
    temp -> prev = cur;
    return head;
}
```

```
Node search (int item, Node head){
    Node cur;
    int count;
    if (head -> next == head){
        pf("empty\n");
        return head;
    }
    cur = head -> next;
    while (cur != head){
        if (item != cur->data){
            cur = cur -> next;
        } else {
            pf("search successful\n");
            return head;
        }
    }
    pf("Search unsuccessful\n");
    return head;
}
```

```
Node deleteDuplicates(int item, Node head){
    Node prev, cur, next;
    int count = 0;
    if(head->next == head){
        Pf("List is empty\n");
        return head;
    }
    cur = head->next;
    while(cur != head){
        if(cur->data != item){
            cur = cur->next;
        }
    }
    else{
        count++;
        if(count == 1){
            cur = cur->next;
            continue;
        }
        else{
            prev = cur->prev;
            next = cur->next;
            prev->next = next;
            next->prev = prev;
            free(cur);
            cur = next;
        }
    }
}
```

```
if (count == 0) {
    pf ("No item found\n");
} else {
    pf ("Remove all dupes \n");
}
return head;
}

void display (Node head) {
    Node temp;
    if (head -> next == head) {
        pf ("empty \n");
        return;
    }
    printf ("Contents of DLL : \n");
    temp = head -> next;
    while (temp != head) {
        pf ("%d", temp -> data);
        temp = temp -> next;
    }
    pf ("\n");
}
```

```
void main() {
    int choice, item, flag=1, key;
    Node head;
    head = getNode();
    head->next = head;
    head->prev = head;
while (flag == 1) {
    pf ("\n1. Insert Front \n2. Insert Rear \n
        3. Delete Front \n4. Delete Rear \n5.
        Insert Before key \n6. Insert After key \n
        7. Search \n8. Delete Duplicates \n9.
        Display \n10. Exit \n");
pf (" Enter choice: ");
sconf ("%d", &choice);
switch (choice) {
    case 1: pf ("Enter item : \n");
            sconf ("%d", &item);
            head = insertFront (item, head);
            break;
```

```
case 2 : printf ("Enter item : \n");
         scanf ("%d", &item);
         head = insertRear (item, head);
         break;

case 3 : head = deleteFront (head);
         break;

case 4 : head = deleteRear (head);
         break;

case 5 : Pf ("Enter item : \n");
         scanf ("%d", &item);
         Printf ("Enter key : \n");
         Scanf ("%d", &key);
         break
         head = insertBeforeKey (item, key, head);
         break;

case 6 : printf ("Enter item : \n");
         &scanf ("%d", &item);
         printf ("Enter key : \n");
         scanf ("%d", &key);
         head = insertAfterKey (item, key, head);
         break;

case 7 : printf ("Enter key to search \n");
         scanf ("%d", &key);
         head = search (item, head);
         break;
```

```
        case 8: printf ("Enter key : \n");
                scanf ("%d", &key);
                head = deleteDuplicates (key, head);
                break;
        case 9: display (head);
                break;
        default: exit(0);
    }
  }
}
```