Mahantesh Yattinahatti 1BM19EE025.

Demonstrating Doubly Linked Lists.

```
#include <stdio.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("memory 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)
{
    NODE cur, next;
    if (head->rlink == head)
    {
        printf("dll empty\n");
        return head;
    }
    cur = head->rlink;
    next = cur->rlink;
    head->rlink = next;
    next->llink = head;
    printf("the node deleted is %d\n", cur->info);
    freenode(cur);
    return head;
}
NODE ddelete_rear(NODE head)
{
    NODE cur, prev;
    if(head->rlink == head)
    {
        printf("dll 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)
    {
        printf("key not found\n");
        return head;
    }
    prev = cur->llink;
    printf("enter towards left of %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->link;
  }
  if(cur == head)
  {
    printf("key not found\n");
    return head;
  }
  next = cur->rlink;
  printf("enter towards the right of %d = ", item);
  temp = getnode();
  scanf("%d", &item->info);
  cur->rlink = temp;
  temp->llink = cur;
  temp->rlink = next;
  next->llink = temp;
  return head;
}
```

```
void search (int item, NODE head)
{
    NODE temp;
    if(head->rlink = head)
    {
        printf("dll empty\n");
        return;
    }
    temp = head->rlink;
    while(temp != head)
    {
        if (temp->info == item)
        { printf("Element found\n");
          return;
        }
        temp =temp->rlink;
    }
    printf("Element not found !!!");
}
void Delete_duplicates(int item, NODE head)
{
    NODE prev, cur, next;
    int count;
    if(head->rlink == head)
    {
        printf("List empty\n");
        return;
    }
```

```
count = 0;
cur = head -> rlink;
while (cur != head)
{   if (item != cur -> info)
      cur = cur -> rlink
      else
      {
          count++;
          if (count > 1)
          {
              prev = cur -> llink;
              next = cur -> rlink;
              prev -> rlink = next;
              next -> llink = prev;
              freenode (cur);
              cur = next;
          }
          else
          {
            cur = cur -> rlink;
          }
      }
}
 if (count == 0)
    printf(" key not found");
 else if(count == 1)
    printf(" duplicates not found\n");
 else
 printf (" duplicates found at %d positions and are
deleted\n", count-1);
}
```

```
void display (NODE head)
{
    NODE temp;
    if (head->rlink == head)
    {
        printf("dll empty \n");
        return;
    }
    printf("contents of dll \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;
    for (;;)
    {   printf("\n1: insert front \n2: insert rear \n3: delete
front \n4 delete rear \n5: insert to the left of the key \n6: insert
to the right of the key \n7: search \n8: delete duplicates
```

```
\n9: display \n10: exit \n");
printf(" enter the choice \n");
scanf("%d", &choice);
switch (choice)
{   case 1: printf(" Enter the item at the 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 item \n");
            scanf("%d", &item);
            last = insert_leftpos(item, head);
            break;
    case 6: printf(" enter the key item \n");
            scanf("%d", &item);
            search(item, head);
            break;
    case 8: printf(" enter the item \n");
            scanf("%d", &item);
            delete_duplicates(item, head);
            break;
    case 9: display(head);
            break;
```

```
        case 10: exit (0);
        default: exit (0);
    }
  }
}
```

Expected output:

```
1: insert front
2: insert rear
3: delete front
4: delete rear
5: insert to the left of the key
6: insert to the right of the key
7: search
8: Delete duplicates
9: display
10: exit
enter the choice
1
enter the item at the front end
1
1: insert front
2: insert rear
3: delete front
4: delete rear
```

5 : insert to the left of the key
6 : insert to the right of the key
7 : search
8 : Delete duplicates
9 : display
10 : exit
enter the choice
10 .