```c
//Doubly linked list with all functions
#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;
}
void freenode(NODE x)
{
    free(x);
}
NODE dinsert_rear(NODE head,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->rlink=NULL;
    temp->llink=NULL;
    temp->info=item;
    cur=head->llink;
    temp->llink=cur;
```

```
2501        temp->llink=cur;
2502        cur->rlink=temp;
2503        head->llink=temp;
2504        temp->rlink=head;
2505        head->info=head->info+1;
2506        return head;
2507    }
2508    NODE dinsert_front(int item,NODE head)
2509    {
2510        NODE temp,cur;
2511        temp=getnode();
2512        temp->info=item;
2513        cur=head->rlink;
2514        head->rlink=temp;
2515        temp->llink=head;
2516        temp->rlink=cur;
2517        cur->llink=temp;
2518        return head;
2519    }
2520    NODE f_tra(NODE head)
2521    {
2522        NODE cur;
2523        int i=0;
2524        if(head->rlink==head){
2525            printf("list empty\n");
2526            return head;
2527        }
2528        printf("List :- \n");
2529        cur=head->rlink;
2530        while(cur!=head){
2531            i++;
2532            if(i%2==0){
2533                printf("%d\n",cur->info);
2534            }
```

```
2534            }
2535            cur=cur->rlink;
2536        }
2537        return head;
2538    }
2539    NODE b_tra(NODE head)
2540    {
2541        NODE cur;
2542        int i=0;
2543        if(head->rlink==head){
2544            printf("list empty\n");
2545            return head;
2546        }
2547        printf("List :- \n");
2548        cur=head->llink;
2549        while(cur!=head){
2550            i++;
2551            if(i%2==0){
2552                printf("%d\n",cur->info);
2553            }
2554            cur=cur->llink;
2555        }
2556        return head;
2557    }
2558    NODE insert_leftpos(int item,NODE head)
2559    {
2560        NODE temp,cur,prev;
2561        if(head->rlink==head){
2562            printf("list empty\n");
2563            return head;
2564        }
2565        cur=head->rlink;
2566        while(cur!=head){
2567            if(item==cur->info)
```

```
2567            if(item==cur->info)
2568                break;
2569            cur=cur->rlink;
2570        }
2571        if(cur==head){
2572            printf("key not found\n");
2573            return head;
2574        }
2575        prev=cur->llink;
2576        printf("enter towards left of %d=",item);
2577        temp=getnode();
2578        scanf("%d",&temp->info);
2579        prev->rlink=temp;
2580        temp->llink=prev;
2581        cur->llink=temp;
2582        temp->rlink=cur;
2583        return head;
2584    }
2585    NODE insert_rightpos(int item,NODE head)
2586    {
2587        NODE temp,cur,prev;
2588        if(head->rlink==head){
2589            printf("list empty\n");
2590            return head;
2591        }
2592        cur=head->rlink;
2593        while(cur!=head){
2594            if(item==cur->info)
2595                break;
2596            cur=cur->rlink;
2597        }
2598        if(cur==head){
2599            printf("key not found\n");
2600            return head;
```

```
2600            return head;
2601        }
2602        prev=cur->rlink;
2603        printf("Enter item :- ");
2604        temp=getnode();
2605        scanf("%d",&temp->info);
2606        prev->llink=temp;
2607        temp->rlink=prev;
2608        cur->rlink=temp;
2609        temp->llink=cur;
2610        return head;
2611    }
2612    NODE ddelete(int item,NODE head){
2613        NODE cur,prev,next;
2614        int j=0;
2615        if(head->rlink==head){
2616            printf("List empty\n");
2617            return head;
2618        }
2619        cur=head->rlink;
2620        prev=cur->llink;
2621        next=cur->rlink;
2622        while(cur!=head){
2623            j++;
2624            if(cur->info==item){
2625                printf("Key found at %d position and is deleted..\n",j);
2626                freenode(cur);
2627                prev->rlink=next;
2628                next->llink=prev;
2629                return head;
2630            }
2631            cur=cur->rlink;
2632            prev=cur->llink;
2633            next=cur->rlink;
```

```
2633            next=cur->rlink;
2634        }
2635        printf("Key not found..\n");
2636        return head;
2637    }
2638    NODE ddelete_front(NODE head)
2639    {
2640        NODE cur,next;
2641        if(head->rlink==head){
2642            printf("List empty\n");
2643            return head;
2644        }
2645        cur=head->rlink;
2646        next=cur->rlink;
2647        head->rlink=next;
2648        next->llink=head;
2649        printf("Deleted key is %d",cur->info);
2650        freenode(cur);
2651        return head;
2652    }
2653    NODE ddelete_rear(NODE head)
2654    {
2655        NODE cur,prev;
2656        if(head->rlink==head){
2657            printf("List empty\n");
2658            return head;
2659        }
2660        cur=head->llink;
2661        prev=cur->llink;
2662        head->llink=prev;
2663        prev->rlink=head;
2664        printf("Deleted key is %d",cur->info);
2665        freenode(cur);
2666        return head;
2667    }
```

```
NODE delete_all_key(int item,NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head){
        printf("List empty..");
        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("Key not found");
    else
        printf("Key found at %d positions and are deleted\n",count);
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head->rlink==head){
```

```
2699        NODE temp;
2700        if(head->rlink==head){
2701            printf("List empty..\n");
2702            return;
2703        }
2704        for(temp=head->rlink;temp!=head;temp=temp->rlink){
2705            printf("%d\n",temp->info);
2706        }
2707    }
2708    void search(int item,NODE head){
2709        NODE temp;
2710        int j=0;
2711        temp=head->rlink;
2712        while(temp!=head){
2713            j++;
2714            if(temp->info==item){
2715                printf("Item found and its position is %d\n",j);
2716                return;
2717            }
2718            temp=temp->rlink;
2719        }
2720        printf("Search unsuccessful\n");
2721    }
2722    NODE delete_all(int item,NODE head)
2723    {
2724        NODE prev,cur,next;
2725        int count;
2726        if(head->rlink==head){
2727            printf("LE");
2728            return head;
2729        }
2730        count=0;
2731        int j=0;
2732        cur=head->rlink;
2733        while(cur!=head){
```

```
2732        cur=head->rlink;
2733        while(cur!=head){
2734            if(item!=cur->info)
2735                cur=cur->rlink;
2736            else{
2737                j++;
2738                if(j==1){
2739                    cur=cur->rlink;
2740                }
2741                if(item==cur->info){
2742                    count++;
2743                    prev=cur->llink;
2744                    next=cur->rlink;
2745                    prev->rlink=next;
2746                    next->llink=prev;
2747                    freenode(cur);
2748                    cur=next;
2749                }
2750            }
2751        }
2752        if(count==0)
2753            printf("Repeated key not found");
2754        else
2755            printf(" Repeated key found at %d positions and are deleted\n",count);
2756        return head;
2757    }
2758    int main()
2759    {
2760        int item,choice;
2761        NODE head;
2762        head=getnode();
2763        head->rlink=head;
2764        head->llink=head;
2765        for(;;){
2766            printf("\n1.Insert_rear\n2.Insert front\n3.Insert at left of given key\n4.Insert at right of given key\n5.Delete\n6.Delete fr
```

```
2767        printf("enter the choice\n");
2768        scanf("%d",&choice);
2769        switch(choice){
2770            case 1:printf("Enter the item at rear end :- ");
2771                scanf("%d",&item);
2772                head=dinsert_rear(head,item);
2773                break;
2774            case 2: printf("Enter the item at front end :- ");
2775                scanf("%d",&item);
2776                head=dinsert_front(item,head);
2777                break;
2778            case 3:printf("Enter the key in whose left position you want to insert an item :- ");
2779                scanf("%d",&item);
2780                head=insert_leftpos(item,head);
2781                break;
2782            case 4:printf("Enter the key in whose right position you want to insert an item :- ");
2783                scanf("%d",&item);
2784                head=insert_rightpos(item,head);
2785                break;
2786            case 5:printf("Enter the key :- ");
2787                scanf("%d",&item);
2788                head=ddelete(item,head);
2789                break;
2790            case 6:head=ddelete_front(head);
2791                break;
2792            case 7:head=ddelete_rear(head);
2793                break;
2794            case 8:printf("Enter the key item :- \n");
2795                scanf("%d",&item);
2796                head=delete_all_key(item,head);
2797                break;
2798            case 9:printf("Enter the key whose every copy you want to delete :- ");
2799                scanf("%d",&item);
```

```
2798            case 9:printf("Enter the key whose every copy you want to delete :- ");
2799                scanf("%d",&item);
2800                head=delete_all(item,head);
2801                break;
2802            case 10:printf("Enter item to be searched :- ");
2803                scanf("%d",&item);
2804                search(item,head);
2805                break;
2806            case 11:printf("List:-\n");
2807                display(head);
2808                break;
2809            case 12:head=f_tra(head);
2810                break;
2811            case 13:head=b_tra(head);
2812                break;
2813            case 14:exit(0);
2814                break;
2815            default:printf("Enter proper instructions!!!");
2816                break;
2817        }
2818    }
2819    return 0;
2820 }
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 1

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
2
Enter the item at front end :- 2

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
14.Exit
enter the choice
3
Enter the key in whose left position you want to insert an item :- 1
enter towards left of 1=3

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
4
Enter the key in whose right position you want to insert an item :- 1
Enter item :- 5

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
5
Enter the key :- 2
Key found at 1 position and is deleted..

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
6
Deleted key is 3
1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
7
Deleted key is 5
1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
8
Enter the key item :-
1
Key found at 1 positions and are deleted

1.Insert_rear
2.Insert front
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
9
Enter the key whose every copy you want to delete :- 3
LE
1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
11
List:-
List empty..

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
enter the choice
1
Enter the item at rear end :- 1

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 2

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 3

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 5

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 4

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
1
Enter the item at rear end :- 1

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
11
List:-
1
2
3
5
4
1

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
9
Enter the key whose every copy you want to delete :- 1
 Repeated key found at 1 positions and are deleted

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
14.Exit
enter the choice
10
Enter item to be searched :- 5
Item found and its position is 4

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
13
List :-
5
2

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
12
List :-
2
5

1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
```

"C:\Users\Shreshtha Aggarwal\Desktop\1stpro\bin\Debug\1stpro.exe"

```
1.Insert_rear
2.Insert front
3.Insert at left of given key
4.Insert at right of given key
5.Delete
6.Delete front
7.Delete rear
8.Delete all key element
9.Delete duplicates
10.Search
11.Display
12.Forward traverse
13.Backward traverse
14.Exit
enter the choice
14

Process returned 0 (0x0)   execution time : 175.508 s
Press any key to continue.
```