LAB-8 LINKED LIST:

```
#include<stdio.h>
#include <stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
  {
    printf("mem full\n");
    exit(0);
  }
  return x;
}

NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
```

```
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}

NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
```

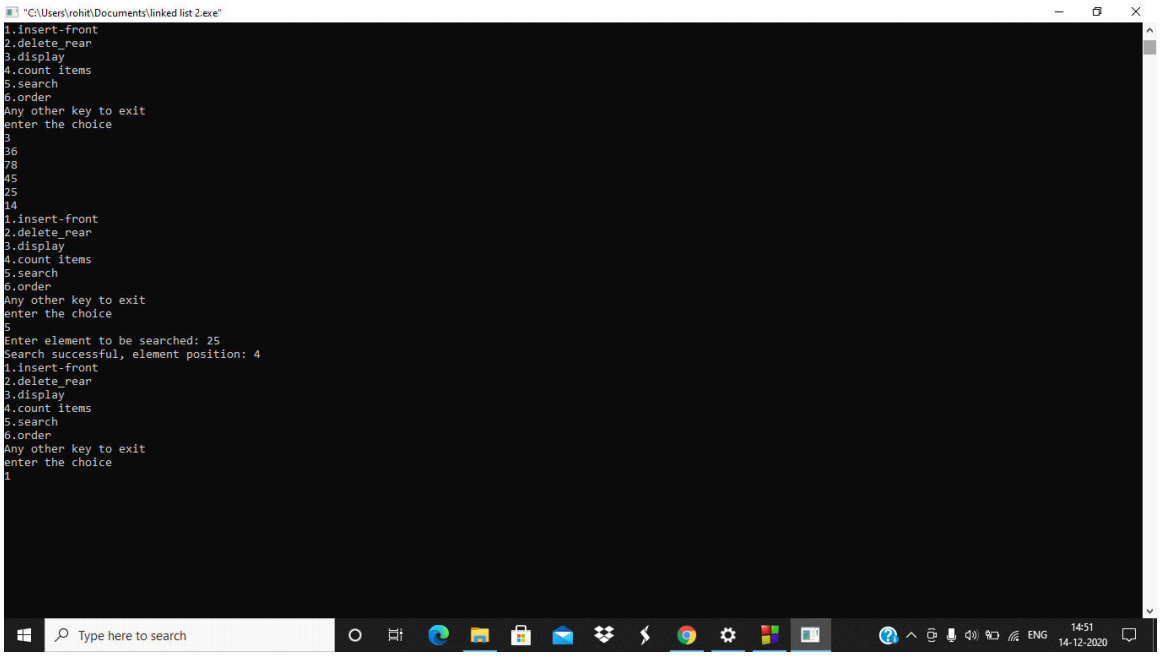
```
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

NODE order_list(NODE first)
{
        int swapped, i;
        NODE ptr1,lptr=NULL;

           if (first == NULL)
           return first;

        do
        {
                swapped = 0;
                ptr1 = first;

                while (ptr1->link != lptr)
                {
                        if (ptr1->info > ptr1->link->info)
                        {
                            int temp = ptr1->info;
                            ptr1->info = ptr1->link->info;
                            ptr1->link->info = temp;
                            swapped = 1;
                        }
```

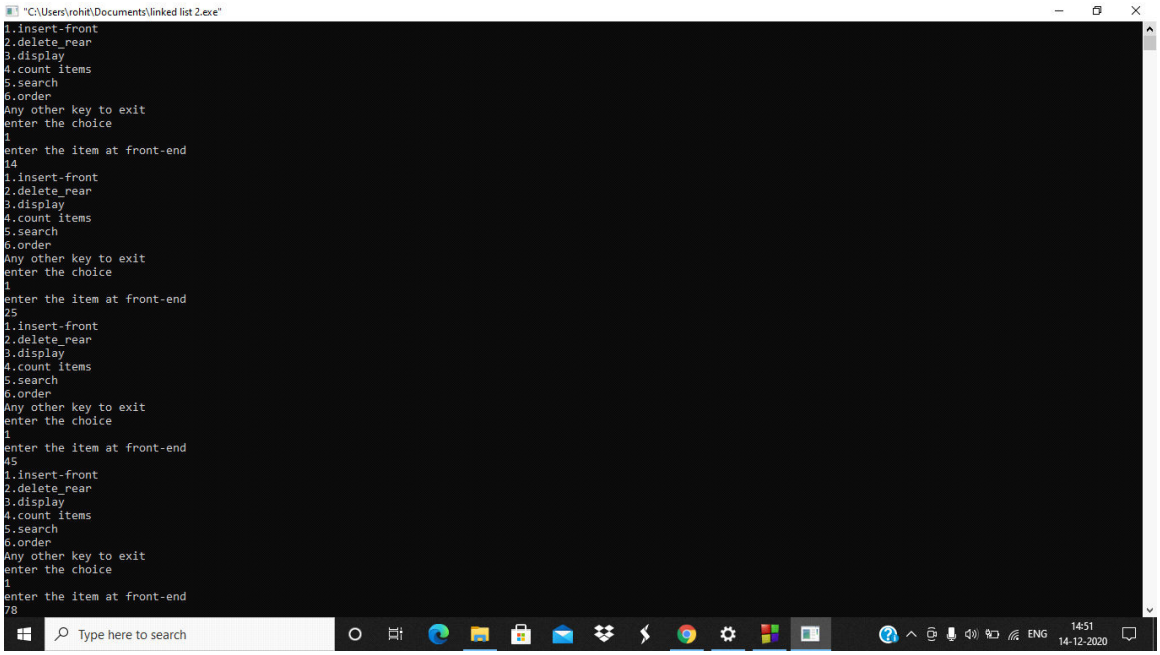
```
                ptr1 = ptr1->link;
            }
            lptr = ptr1;
        }
        while (swapped);
        return first;
}

void count(NODE first){
        NODE temp;
        temp=first;
        int c=0;
        while(temp!=NULL){
        temp=temp->link;
        c++;
        }
        printf("Number of elements: %d\n",c);
}

void list_search(NODE first, int key){
        NODE temp;
        temp=first;
        int c=0,f=0;
        while(temp!=NULL){
            c++;
            if(temp->info==key){
                printf("Search successful, element position: %d\n",c);
                f=1;break;
```

```
                }
                temp=temp->link;
        }
        if(f==0)
        printf("Search Unsuccessful!\n");
}
void display(NODE first)
{
  NODE temp;
  if(first==NULL)
  printf("list empty cannot display items\n");
  for(temp=first;temp!=NULL;temp=temp->link)
    {
    printf("%d\n",temp->info);
    }
}
int main(){
        int item,choice,pos,i,n;
        NODE first=NULL;
        for(;;)
                {
                printf("1.insert-front\n2.delete_rear\n3.display\n4.count items\n5.search\n6.order\nAny
other key to exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
  {
    case 1:printf("enter the item at front-end\n");
              scanf("%d",&item);
```

```
            first=insert_front(first,item);
            break;
    case 2:first=delete_rear(first);
            break;
    case 3:display(first);
            break;
    case 4:count(first);
    break;
    case 5:printf("Enter element to be searched: ");
    scanf("%d",&item);
    list_search(first,item);
    break;
    case 6:
    first=order_list(first);
    break;
  default:exit(0);
  }
}
}
```

OUTPUT:





LAB-9 DOUBLE LINKED LIST:

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

```
#include <stdlib.h>

typedef struct node{
	struct node *prev;
	int data;
	struct node *next;
}*NODE;

NODE makeNode(int x){
	NODE temp = (NODE)malloc(sizeof(struct node));
	temp->prev = NULL;
	temp->data = x;
	temp->next = NULL;
	return temp;
}

/*
	Insert Functions
*/

NODE insertFront(NODE head){
	int ele;
	printf("\nElement:");
	scanf("%d", &ele);

	NODE temp = makeNode(ele);
	temp->next = head;
```

```
        return temp;
}

NODE insertRear(NODE head){
        int ele;
        printf("\nElement:");
        scanf("%d", &ele);

        NODE temp = makeNode(ele);

        if(head == NULL){
                return temp;
        }
        else{
                NODE p = head;

                // p will point to last element
                while((p->next) != NULL){
                        p = p->next;
                }
                p->next = temp;
                temp->prev = p;

                return head;
        }
}

NODE insertPos(NODE head){
```

```
        int ele, pos;
        printf("\nElement & Position: ");
        scanf("%d %d", &ele, &pos);

        NODE temp = makeNode(ele);

        if(head == NULL){
                if(pos != 1)
                        printf("\nPosition doesnt exist!");
                else
                        return temp;
        }
        else{
                NODE p = head;

                // p will point to pos-1'th element
                for(int c=0; c<pos-1; c++){
                        p = p->next;
                }
                // backup p->next
                (p->next)->prev = temp;
                temp->next = p->next;
                p->next = temp;
                temp->prev = p;

                return head;
        }
}
```

```
/*
	Delete Functions
*/

NODE deleteFront(NODE head){
	if(head == NULL)
		printf("\nList Empty!");
	else{
		NODE temp = head->next;
		if(temp == NULL){
			return NULL;
		}
		else{
			temp->prev = NULL;
			free(head);
			return temp;
		}
	}
}

NODE deleteRear(NODE head){
	if(head == NULL)
		printf("\nList Empty!");
	else{
		NODE temp = head;
		// temp goto last but 1 ele
		while((temp->next)->next != NULL){
```

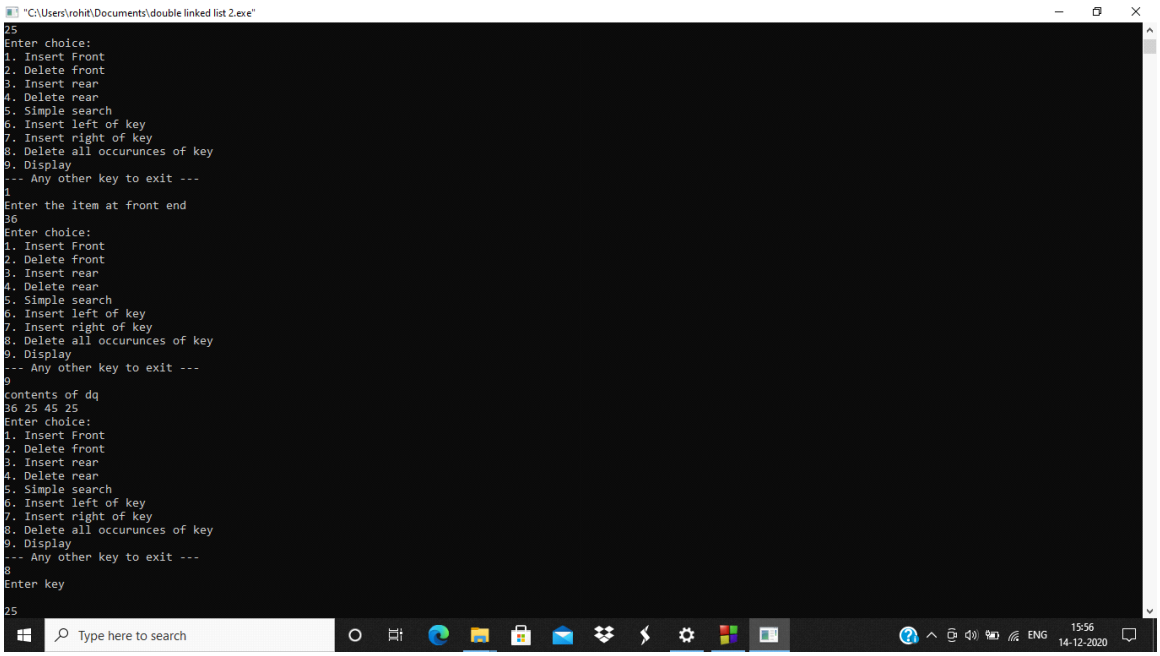
```
            temp = temp->next;
        }
        free(temp->next);
        temp->next == NULL;
        return head;
    }
}

NODE deletePos(NODE head){
    if(head == NULL)
        printf("\nList Empty!");
    else{
        int pos;
        NODE temp = head;
        printf("Enter Position:");
        scanf("%d", &pos);

        if(pos == 1)
            head = deleteFront(head);
        else{
            int i=1;
            while(i < pos){
                if(temp->next != NULL)
                    temp = temp->next;
                else{
                    printf("Position doesnt exist!");
                    return head;
                }
```

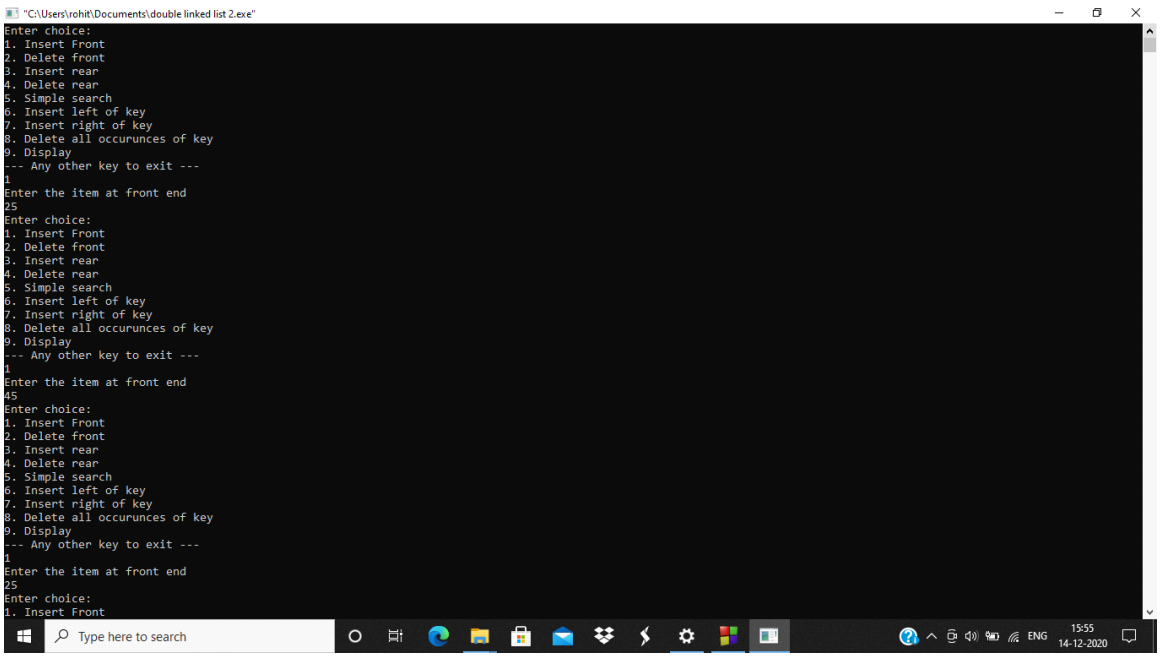
```
                i++;
            }
            NODE posnd = temp->next;
            (posnd->next)->prev = temp;
            temp->next = posnd->next;
            free(posnd);
            return head;
        }
    }
}

void display(NODE head){
    if(head == NULL)
        printf("\nEmpty List!");
    else{
        NODE p = head;

        printf("\nLIST >> ");
        while(p != NULL){
            /* data view */
            printf("%d ", p->data);

            /* full view */
            // printf("\n%d\t%d\t%d", &(p->prev), p->data, &(p->next));

            p = p->next;
        }
    }
```

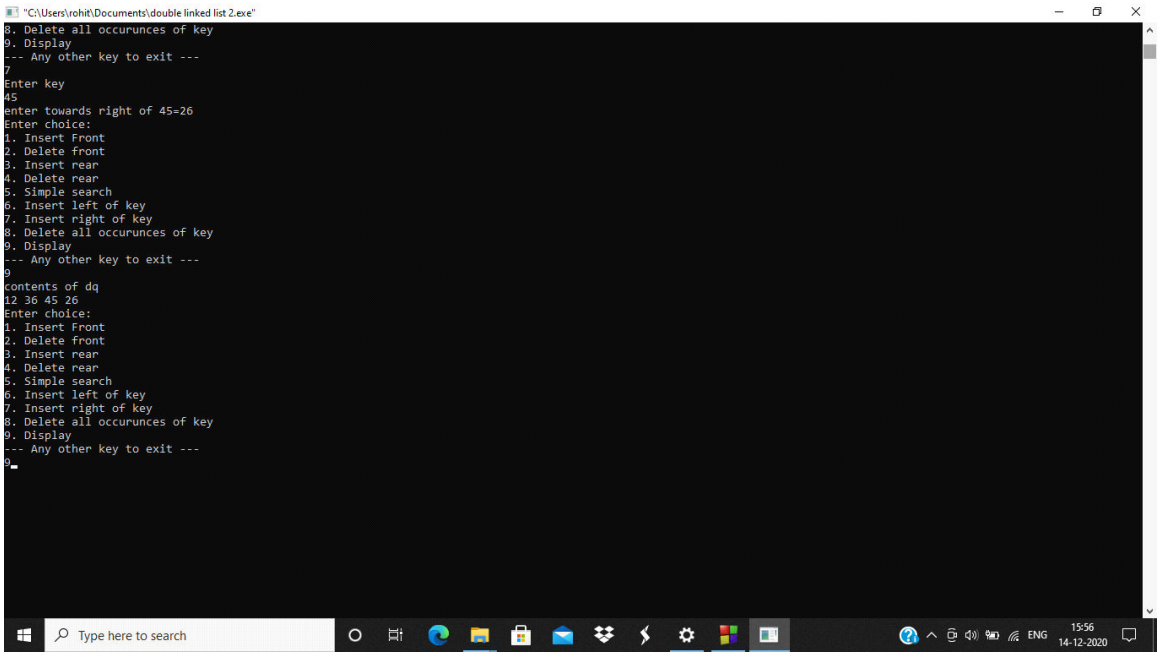
```
}

void main(){
	NODE head = NULL;
	int ch;

	printf("\n\n---MENU---\n1.Ins Fr\n2.Ins Rr\n3.Ins Ps");
	printf("\n4.Del Fr\n5.Del Rr\n6.Del Ps\n8.Exit\n");
	while(1){
		// printf("\n\n---MENU---\n1.Ins Fr\n2.Ins Rr\n3.Ins Ps");
		// printf("\n4.Del Fr\n5.Del Rr\n6.Del Ps\n420.Exit\n");
		printf("\nChoice:");
		scanf("%d", &ch);
		switch(ch){
			case 1:
				head = insertFront(head);
				display(head);
				break;
			case 2:
				head = insertRear(head);
				display(head);
				break;
			case 3:
				head = insertPos(head);
				display(head);
				break;
			case 4:
				head = deleteFront(head);
```

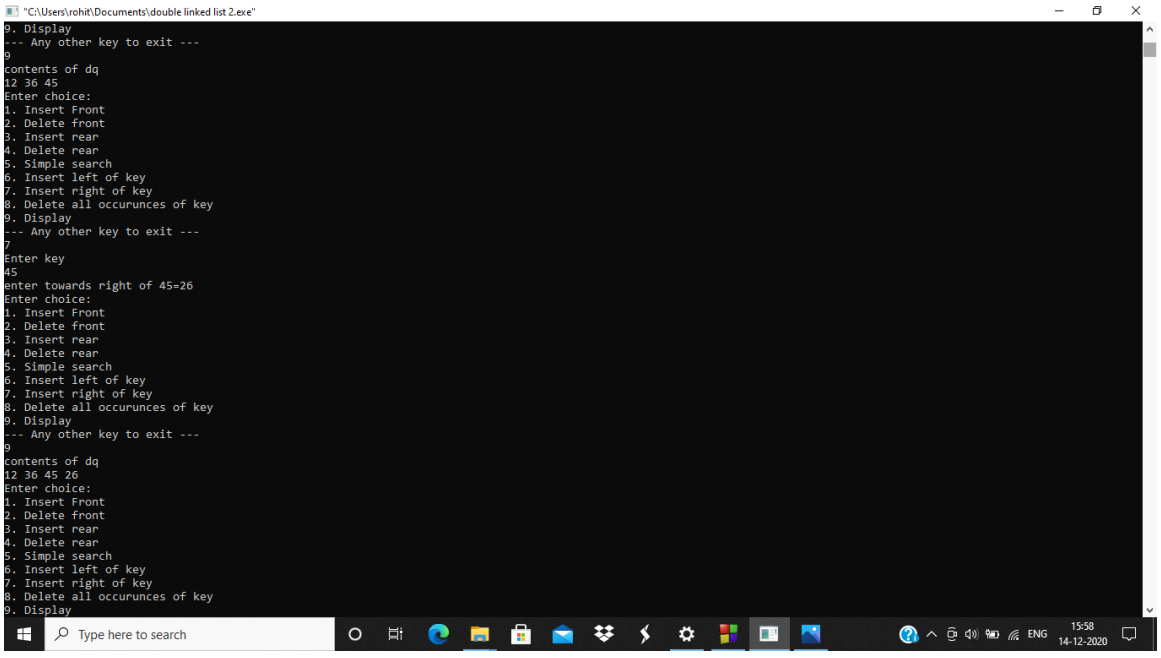
```
                display(head);
                break;
            case 5:
                head = deleteRear(head);
                display(head);
                break;
            case 6:
                head = deletePos(head);
                display(head);
                break;
            case 8:
                printf("\nExiting!");
                exit(1);
            default:
                printf("\nWrong Input!");
        }
    }
}
```

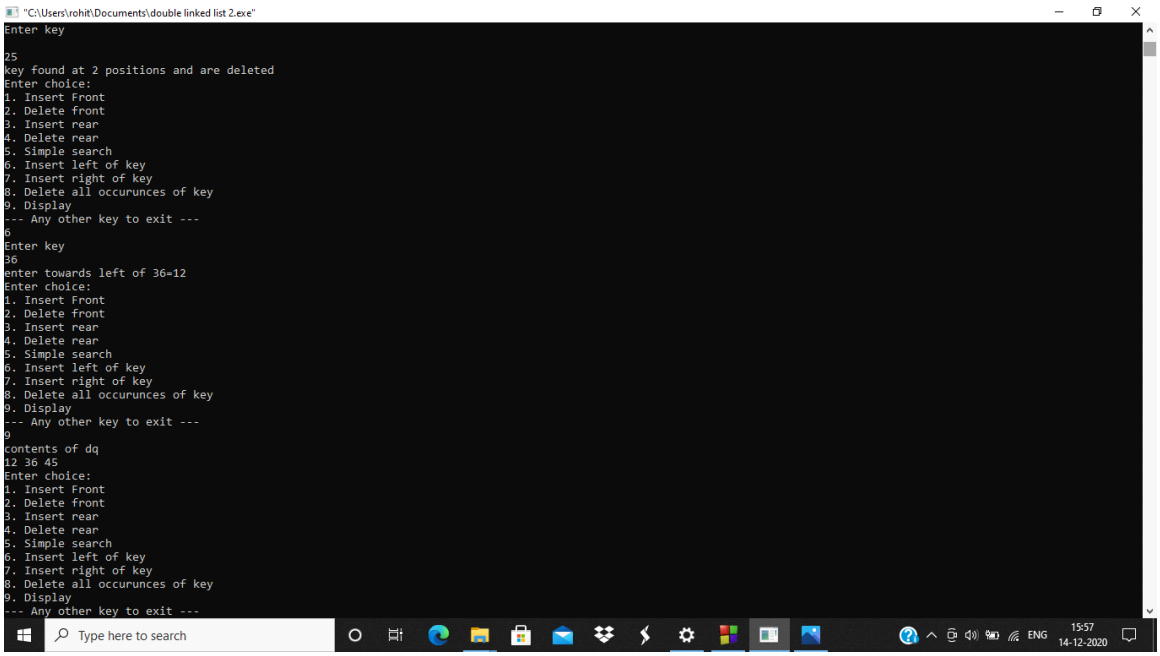
OUTPUT:


"C:\Users\rohit\Documents\double linked list 2.exe"
25
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
1
Enter the item at front end
36
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
contents of dq
36 25 45 25
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
8
Enter key
25
Type here to search
ENG
15:56
14-12-2020


"C:\Users\rohit\Documents\double linked list 2.exe"
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
1
Enter the item at front end
25
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
1
Enter the item at front end
45
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
1
Enter the item at front end
25
Enter choice:
1. Insert Front
Type here to search
ENG
15:55
14-12-2020


"C:\Users\rohit\Documents\double linked list 2.exe"
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
7
Enter key
45
enter towards right of 45=26
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
contents of dq
12 36 45 26
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
Type here to search
ENG
15:56
14-12-2020


"C:\Users\rohit\Documents\double linked list 2.exe"
9. Display
--- Any other key to exit ---
9
contents of dq
12 36 45
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
7
Enter key
45
enter towards right of 45=26
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
--- Any other key to exit ---
9
contents of dq
12 36 45 26
Enter choice:
1. Insert Front
2. Delete front
3. Insert rear
4. Delete rear
5. Simple search
6. Insert left of key
7. Insert right of key
8. Delete all occurunces of key
9. Display
Type here to search
ENG
15:58
14-12-2020



LAB-10 BINARY TREE:

```
#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("mem full\n");
		exit(0);
	}
	return x;
}
void freenode(NODE x)
{
	free(x);
}
NODE insert(NODE root,int item)
{
	NODE temp,cur,prev;
	temp=getnode();
	temp->rlink=NULL;
	temp->llink=NULL;
	temp->info=item;
	if(root==NULL)
	return temp;
	prev=NULL;
	cur=root;
	while(cur!=NULL)
	{
		prev=cur;
		cur=(item<cur->info)?cur->llink:cur->rlink;
	}
```

```
        if(item<prev->info)
        prev->llink=temp;
        else
        prev->rlink=temp;
        return root;
}
void display(NODE root,int i)
{
        int j;
        if(root!=NULL)
        {
                display(root->rlink,i+1);
                for(j=0;j<i;j++)
                printf("    ");
                printf("%d\n",root->info);
                display(root->llink,i+1);
        }
}
NODE delete(NODE root,int item)
{
        NODE cur,parent,q,suc;
        if(root==NULL)
        {
                printf("empty\n");
                return root;
        }
        parent=NULL;
        cur=root;
```

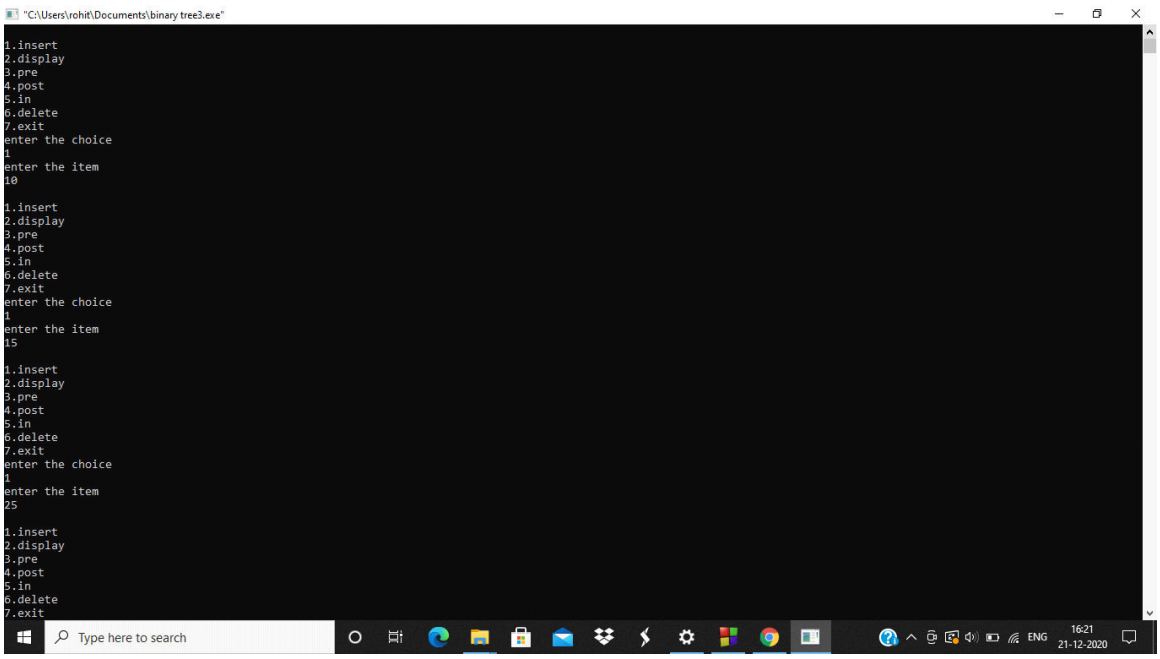
```
while(cur!=NULL&&item!=cur->info)
{
        parent=cur;
        cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
        printf("not found\n");
        return root;
}
if(cur->llink==NULL)
        q=cur->rlink;
else if(cur->rlink==NULL)
        q=cur->llink;
else
{
        suc=cur->rlink;
        while(suc->llink!=NULL)
        suc=suc->llink;
        suc->llink=cur->llink;
        q=cur->rlink;
}
if(parent==NULL)
        return q;
if(cur==parent->llink)
        parent->llink=q;
else
        parent->rlink=q;
```

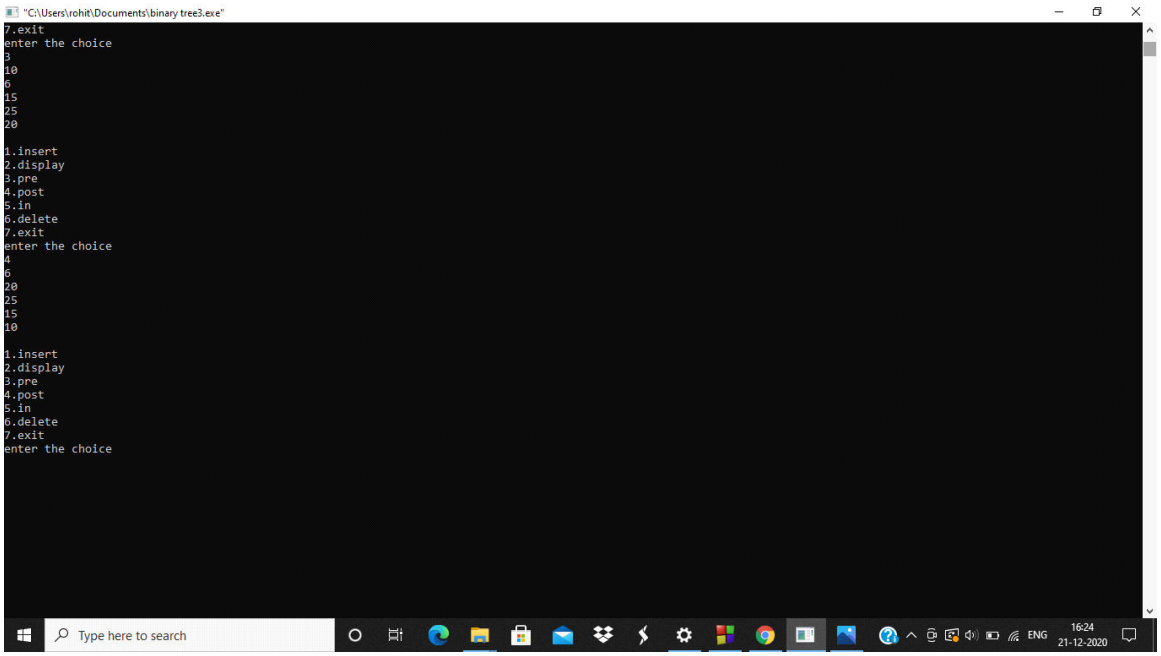
```
                freenode(cur);
                return root;
        }

void preorder(NODE root)
{
        if(root!=NULL)
        {
                printf("%d\n",root->info);
                preorder(root->llink);
                preorder(root->rlink);
        }
}
void postorder(NODE root)
{
        if(root!=NULL)
        {

                postorder(root->llink);
                postorder(root->rlink);
                printf("%d\n",root->info);
        }
}
void inorder(NODE root)
{
        if(root!=NULL)
        {
```

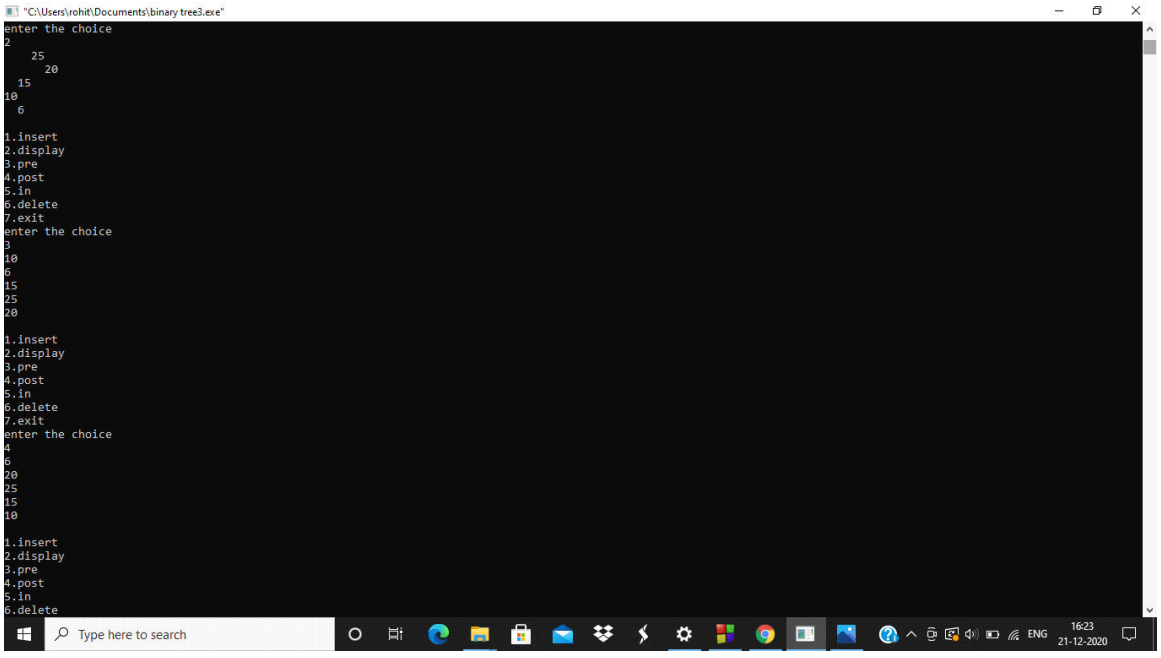
```
                inorder(root->llink);
                printf("%d\n",root->info);
                inorder(root->rlink);
        }
}
void main()
{
        int item,choice;
        NODE root=NULL;
        for(;;)
        {
                printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
                printf("enter the choice\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:printf("enter the item\n");
                                scanf("%d",&item);
                               root=insert(root,item);
                                break;
                        case 2:display(root,0);
                                break;
                        case 3:preorder(root);
                                break;
                        case 4:postorder(root);
                                break;
                        case 5:inorder(root);
                                break;
```

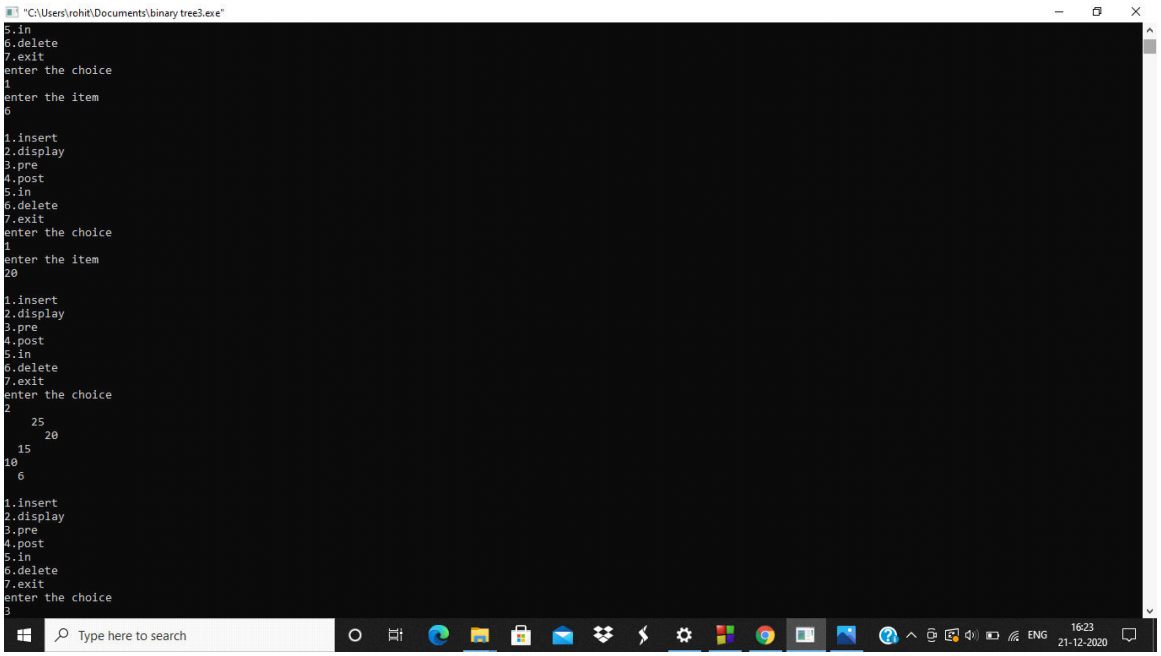
```
                case 6:printf("enter the item\n");
                       scanf("%d",&item);
                       root=delete(root,item);
                       break;
                default:exit(0);
                       break;
            }
        }
}
```

OUTPUT:




"C:\Users\rohit\Documents\binary tree3.exe"
7.exit
enter the choice
3
10
6
15
25
20
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
6
20
25
15
10
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
Type here to search
ENG
16:24
21-12-2020


"C:\Users\rohit\Documents\binary tree3.exe"
enter the choice
2
25
20
15
10
6
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
10
6
15
25
20
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
6
20
25
15
10
1.insert
2.display
3.pre
4.post
5.in
6.delete
Type here to search
ENG
16:23
21-12-2020


"C:\Users\rohit\Documents\binary tree3.exe"
5.in
6.delete
7.exit
enter the choice
1
enter the item
6
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
20
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
25
20
15
10
6
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
Type here to search
ENG
16:23
21-12-2020