WEEK 11- DOUBLY LINKED LIST PRIMITIVE OPERATIONS

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

 int data;

 struct node *prev;

 struct node *next;

};

typedef struct node *NODE;

int count;

NODE getnode()

{

 NODE p;

 p=(NODE)malloc(sizeof(struct node));

if(p==NULL)

{

    printf("mem is full\n");

    exit(0);

}
```

```
return p;

}

NODE insert_front(int item,NODE head)

{

  NODE newn,cur;

  newn=getnode();

  newn->data=item;

  newn->prev=NULL;

  newn->next=NULL;

  cur=head->next;

    head->next=newn;

    newn->prev=head;

    newn->next=cur;

    cur->prev=newn;

  return head;

}

NODE insert_rear(int item,NODE head)

{

  NODE newn,cur;

  newn=getnode();
  newn->data=item;
  newn->prev=NULL;
  newn->next=NULL;
  cur=head->prev;
  head->prev=newn;
  newn->next=head;
  cur->next=newn;
  newn->prev=cur;
  return head;
}
NODE insert_left(int item,NODE head)
{
    NODE newn,cur,left;

    if(head->next==head)
    {
        printf("DLL is empty");
       return head;
    }
    else
```

```
{
    cur=head->next;
    while(cur!=head)
    {
        if(cur->data==item)
        {
         break;
        }
        cur=cur->next;
    }
    if(cur==head)
    {
        printf("key node for insertion not found");
        return head;
    }
    left=cur->prev;//node left of key
    printf("Enter data towards left of %d=",item);
     newn=getnode();
     scanf("%d",&newn->data);
     newn->prev=left;
```

```
         left->next=newn;

         newn->next=cur;

         cur->prev=newn;

         return head;

    }


}

NODE insert_right(int item,NODE head)

{

    NODE newn,cur,right;

    if(head->next==head)

    {

        printf("DLL is empty");

        return head;

    }

    cur=head->next;

    while(cur!=head)

    {

        if(cur->data==item)

        {
            break;

        }

            cur=cur->next;

    }

    if(cur==head)

    {

        printf("key node for insertion not found");

        return head;

    }

    right=cur->next;

    newn=getnode();

    printf("Enter data towards right of %d",item);

    scanf("%d",&newn->data);

    right->prev=newn;

    newn->next=right;

   newn->prev=cur;

   cur->next=newn;

  return head;

}

NODE del_front(NODE head)
```

```
{
   NODE cur,next;
   if(head->next==head)
   {
       printf("DLL is empty");
       return head;
   }
   cur=head->next;
   next=cur->next;
   head->next=next;
   next->prev=head;
   printf("The node deleted is %d",cur->data);
   free(cur);
   return head;
}
NODE del_rear(NODE head)
{
    NODE cur,left;
    if(head->next==head)
    {
        printf("The DLL is empty");
    }
    cur=head->prev;
    left=cur->prev;
    left->next=head;
    head->prev=left;
    printf("The node deleted is %d",cur->data);
  free(cur);
  return head;
}
NODE dll_search(int item,NODE head)
{
    count=0;
    NODE cur;
    cur=head->next;
    while(cur!=head)
    {
        count++;
        if(cur->data==item)
        {
            printf("Item found at %d",count);
            return head;
        }
        cur=cur->next;
    }
    if(count==0)
    {
        printf("Element not found");
    }
    return head;
}
NODE del_all(int item,NODE head)
{
    NODE left,cur,right;
    if(head->next==head)
    {
        printf("DLL empty");
        return head;
    }
count=0;
```

```
cur=head->next;
while(cur!=head)
{
   if(item!=cur->data)
   {
       cur=cur->next;
   }
   else
   {
      count++;
      left=cur->prev;
      right=cur->next;
      left->next=right;
      right->prev=left;
      free(cur);
      cur=right;
   }
}
if(count==0)
{
    printf("key not found");
}
else
{
    printf("key found at %d positions and are deleted",count);
}
return head;
}
NODE remove_duplicates(int item,NODE head)
{
    NODE cur,left,right;
    count=0;
    if(head->next==head)
    {
        printf("DLL is empty");
    }
    cur=head->next;

    while(cur!=head)
    {
```

```
if(cur->data==item)
{
    count++;
    if(count>1)
    {
        left=cur->prev;
        right=cur->next;
        left->next=right;
        right->prev=left;
        free(cur);
        cur=right;
    }
    else
    {
        cur=cur->next;
    }
}
else if(cur->data!=item)
{
        cur=cur->next;
      }

   }
   if(count==1)
   {
      printf("Given key has no duplicates");
   }
   else
   {
     printf("Duplicates have been removed..only unique present");
   }

   return head;
}
void display(NODE head)
{
   NODE cur;
   if(head->next==head)
   {
        printf("DOUBLY LINKED LIST is empty");
        return;
    }
   cur=head->next;
   while(cur!=head)
   {
      printf("%d ",cur->data);
      cur=cur->next;
   }
   printf("\n");
}
int main()
{
    NODE head;
    head=getnode();
    head->next=head;
    head->prev=head;
    NODE p;
    int ch,item;
    int i,c,val,pos;
    for(;;)

    {

    printf("\n1.Insert front\n2.Insert rear\n3.Insert left of the given
node\n4.insert right of given node\n5.delete front\n6.delete
rear\n7.delete all occurences of key\n8.display\n9.searching for a given
element\n10.removing duplicates");

    printf("\nPlease enter your choice");

   scanf("%d",&ch);


      switch(ch)

      {

         case 1:

                printf("Enter the item:");

                   scanf("%d",&item);

                   head=insert_front(item,head);

                break;

         case 2:printf("Enter the key item");

                scanf("%d",&item);

                head=insert_rear(item,head);

                break;

         case 3:
```

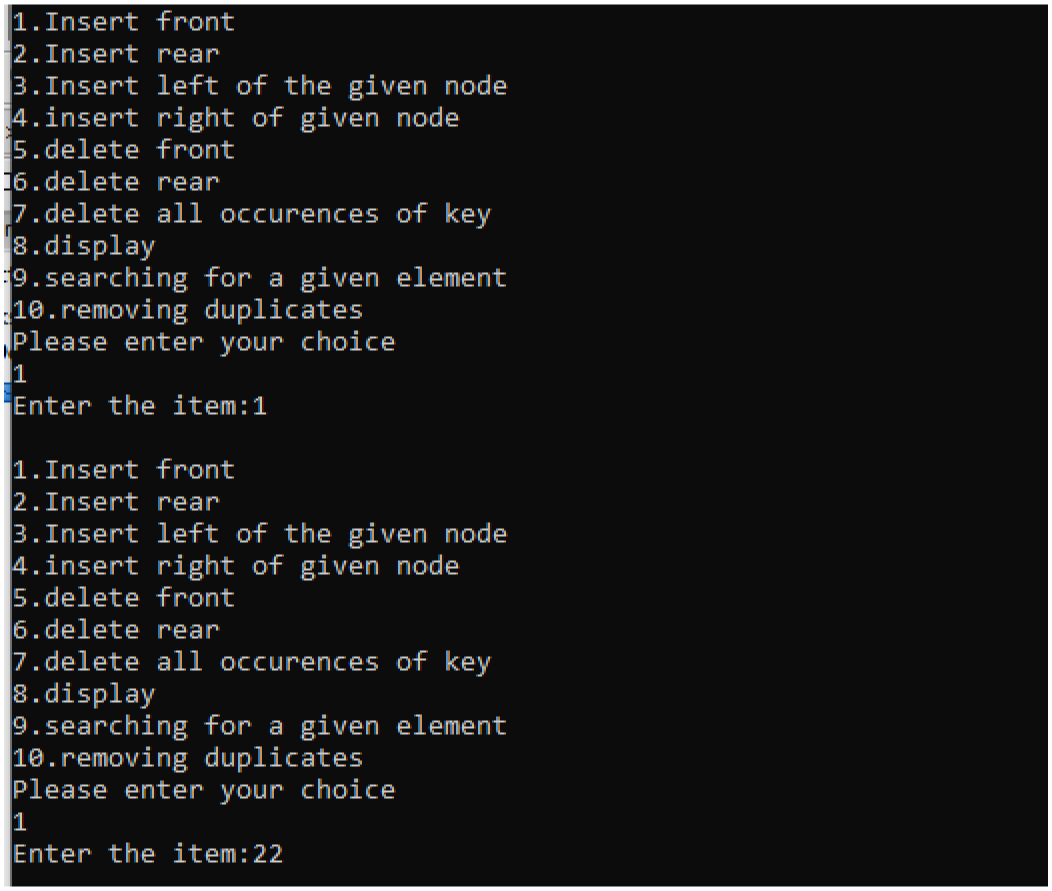
```
       printf("Enter the key item\n");
       scanf("%d",&item);
         head=insert_left(item,head);
         break;
case 4: c=0;
       printf("Enter the key item\n");
       scanf("%d",&item);
       head=insert_right(item,head);
       break;
case 5:del_front(head);
      break;
case 6:del_rear(head);
      break;
case 7:printf("Enter the key to be deleted");
        scanf("%d",&item);
        head=del_all(item,head);
        break;
case 8:display(head);
       break;
case 9:printf("Enter the value to be searched");
```

```
                 scanf("%d",&item);

                 head=dll_search(item,head);

                break;

         case 10:printf("Enter the key whose duplicates need to be
deleted");

                scanf("%d",&item);

                head=remove_duplicates(item,head);

                break;

        default:exit(0);

      }

   }

    return 0;

}
```

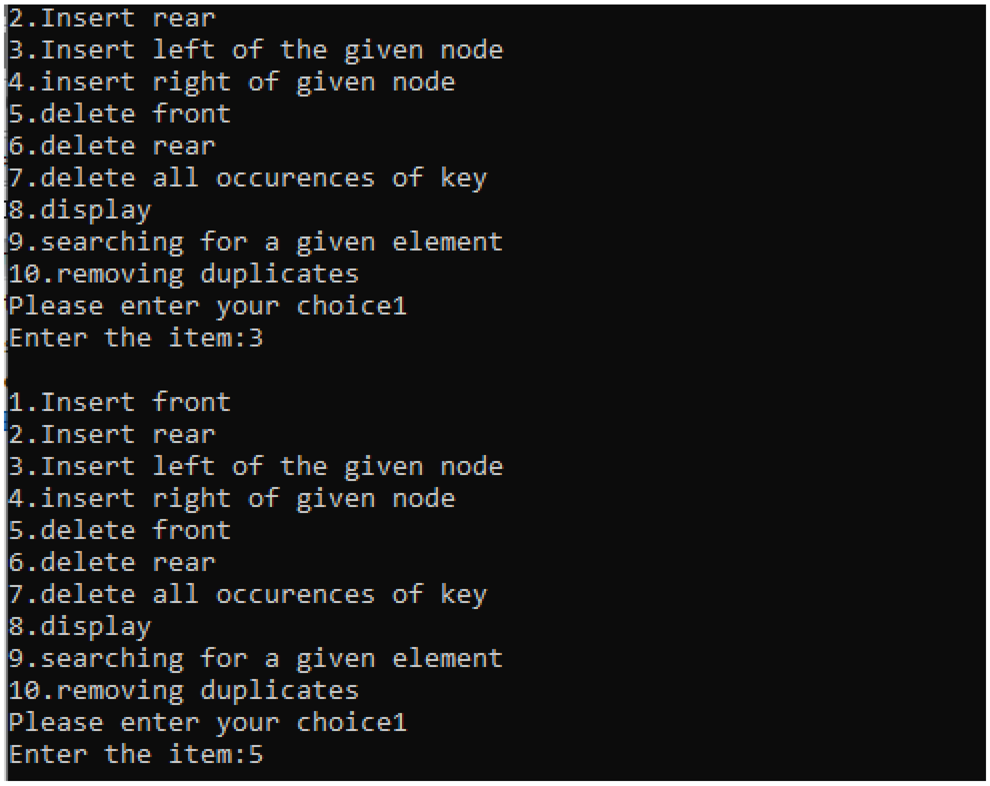
OUTPUT

Function wise:

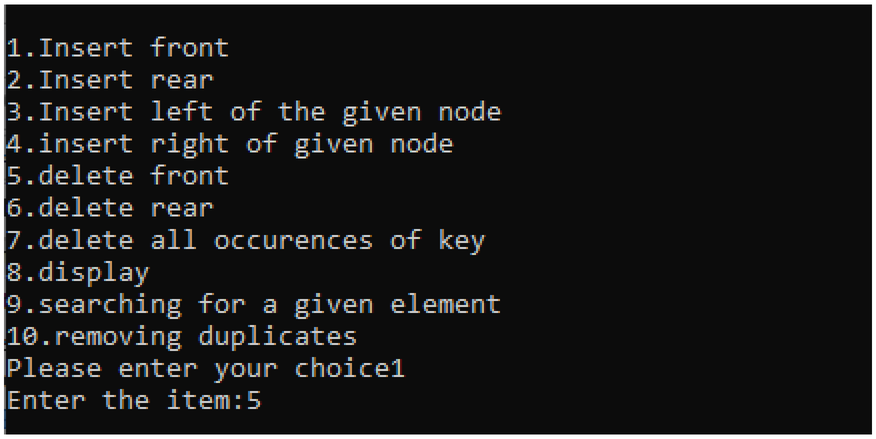
1.insert front



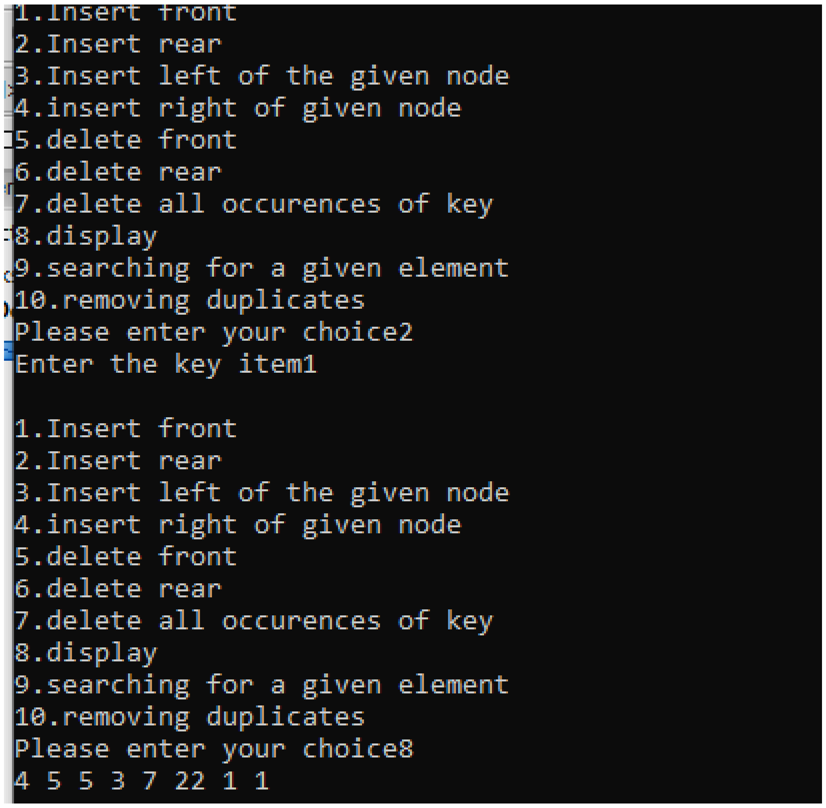
```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice
1
Enter the item:1

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice
1
Enter the item:22
```



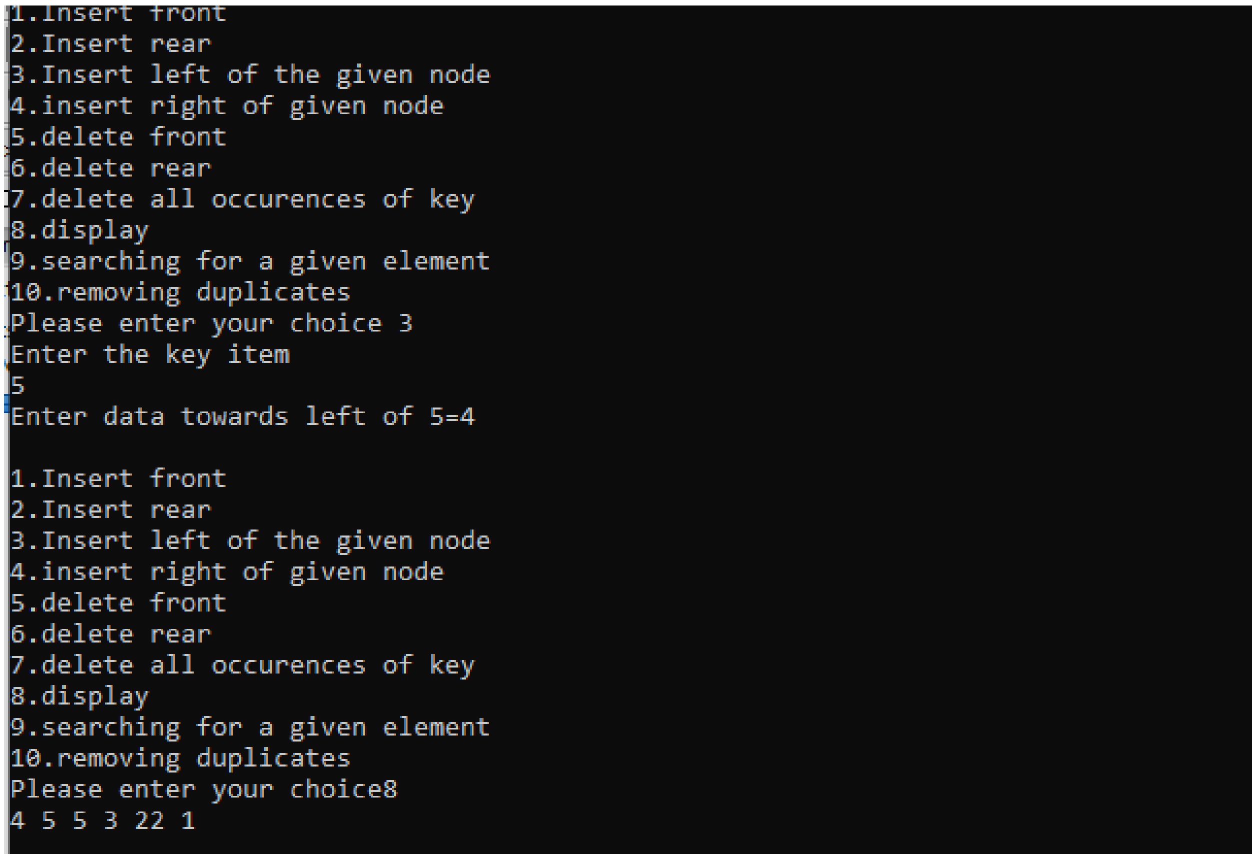
```
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:3

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:5
```


```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice1
Enter the item:5
```

2.insert rear

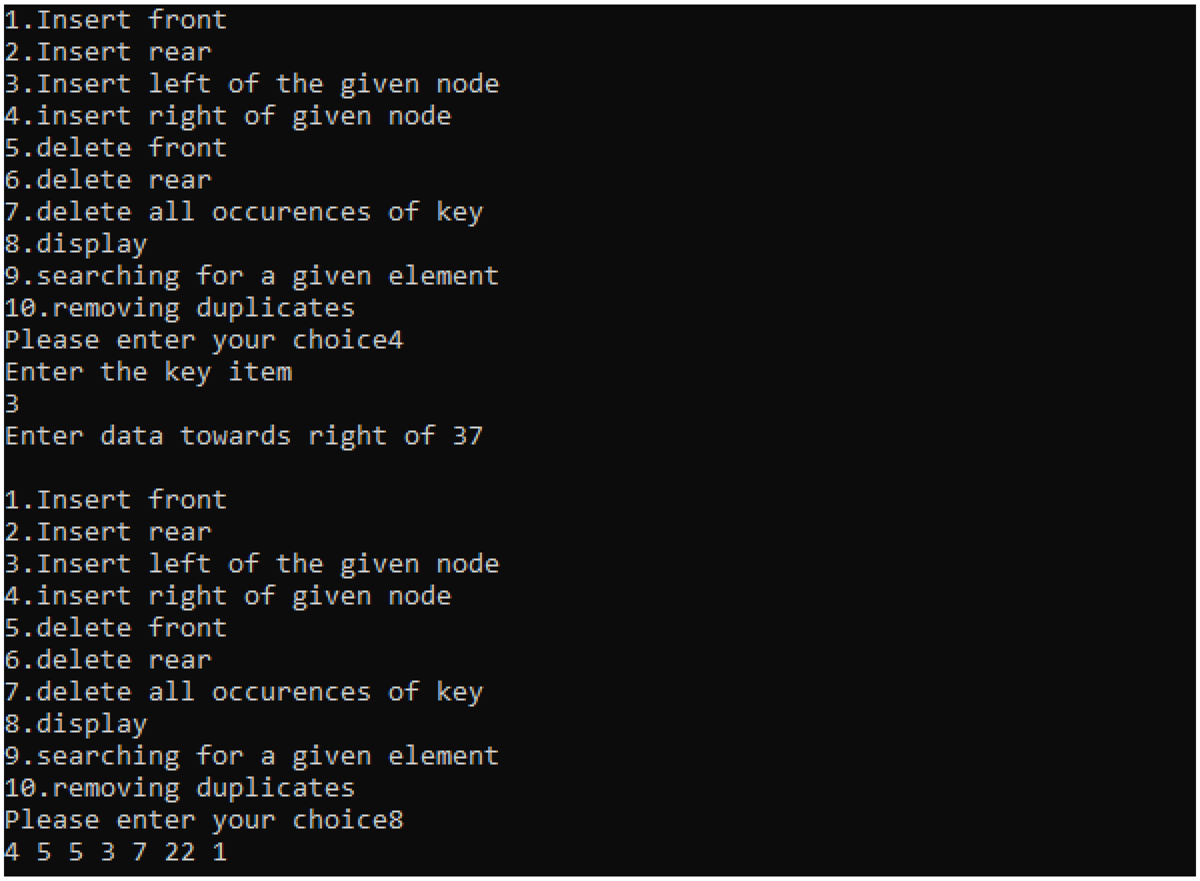

```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice2
Enter the key item1

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22 1 1
```

3.insert left



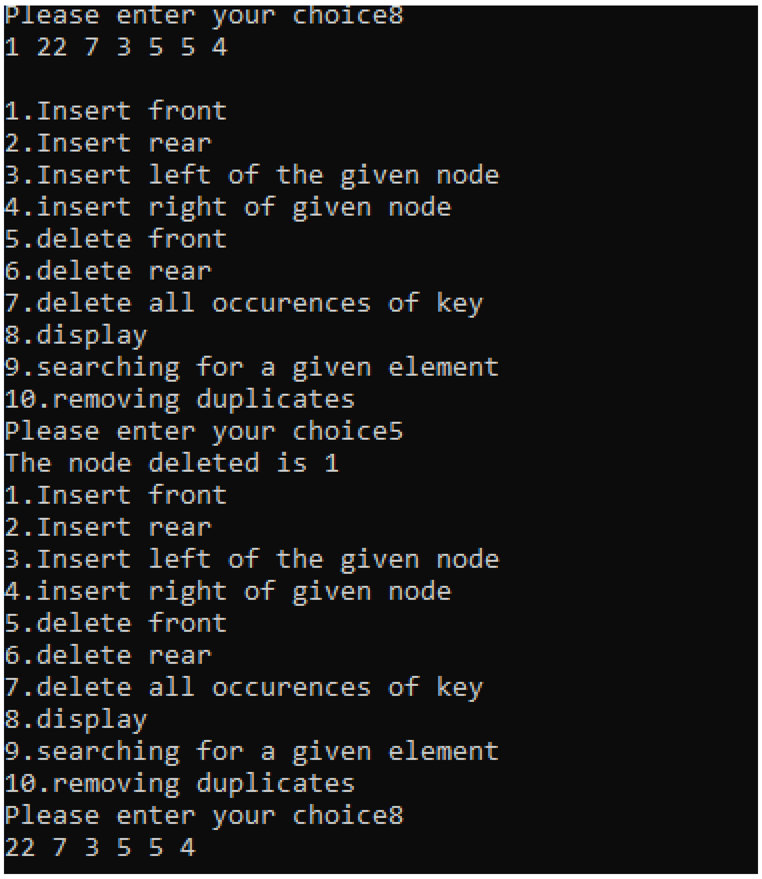
```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice 3
Enter the key item
5
Enter data towards left of 5=4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 22 1
```

4.insert right



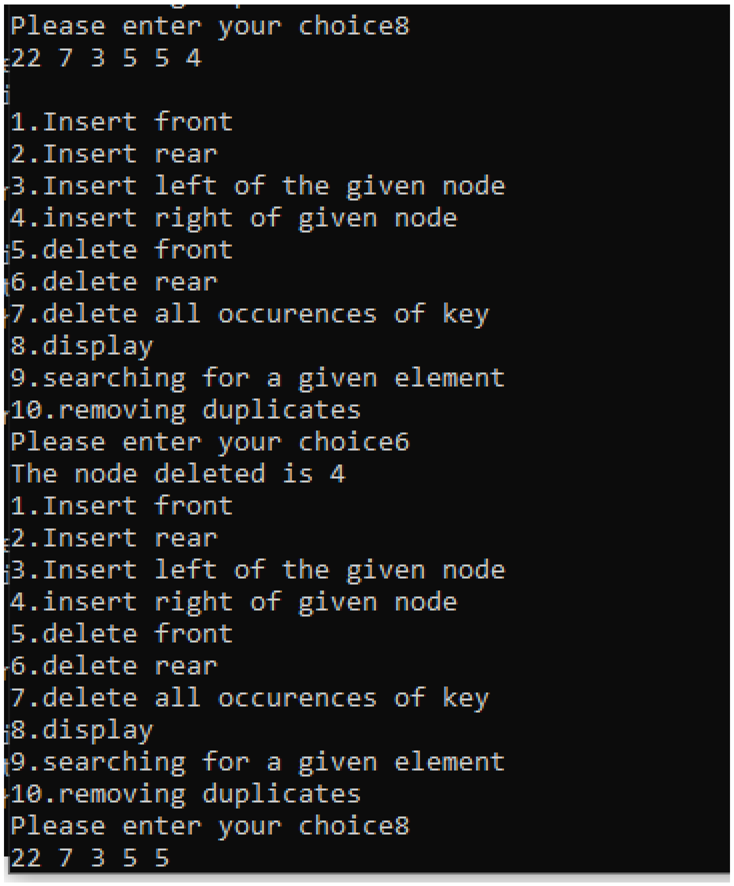
```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice4
Enter the key item
3
Enter data towards right of 37

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22 1
```

5.delete front



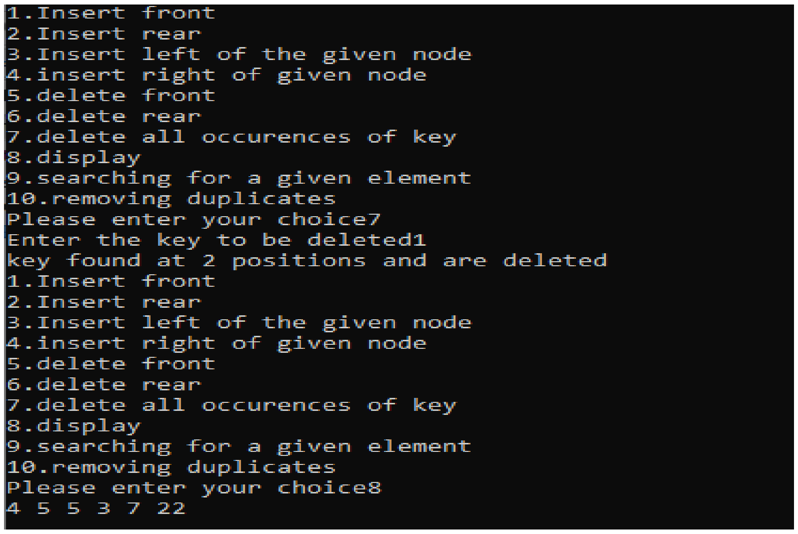
```
Please enter your choice8
1 22 7 3 5 5 4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice5
The node deleted is 1
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5 4
```

6.delete rear



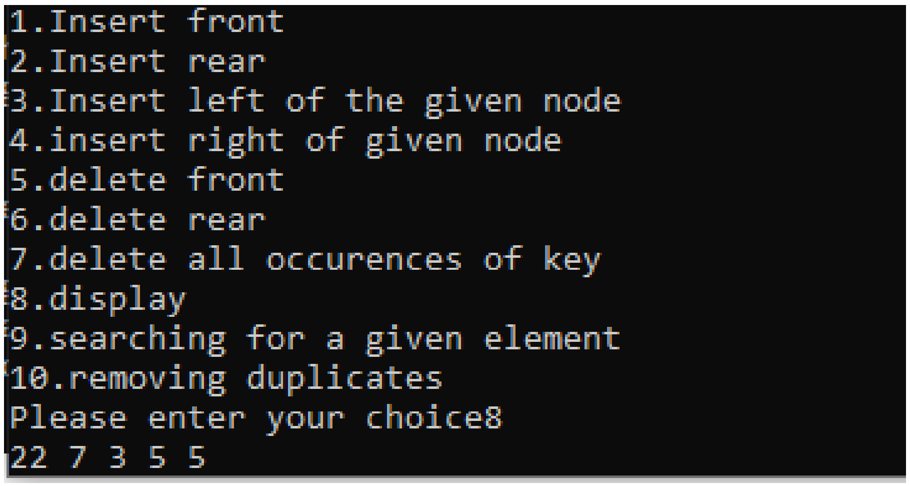
```
Please enter your choice8
22 7 3 5 5 4

1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice6
The node deleted is 4
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5
```

7.delete all occurences



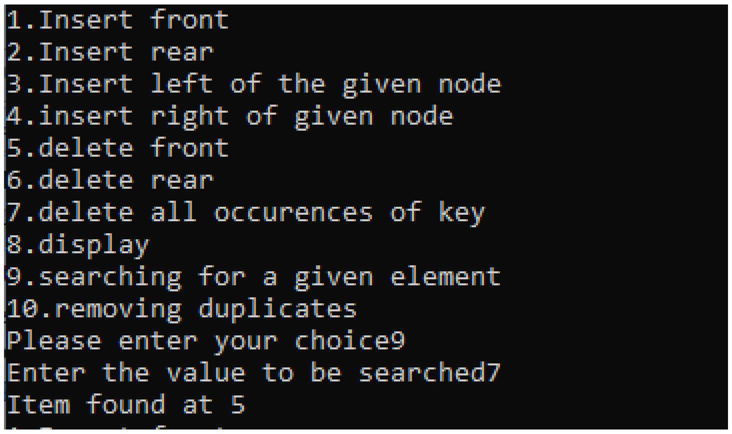
```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice7
Enter the key to be deleted1
key found at 2 positions and are deleted
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 5 3 7 22
```

8.display



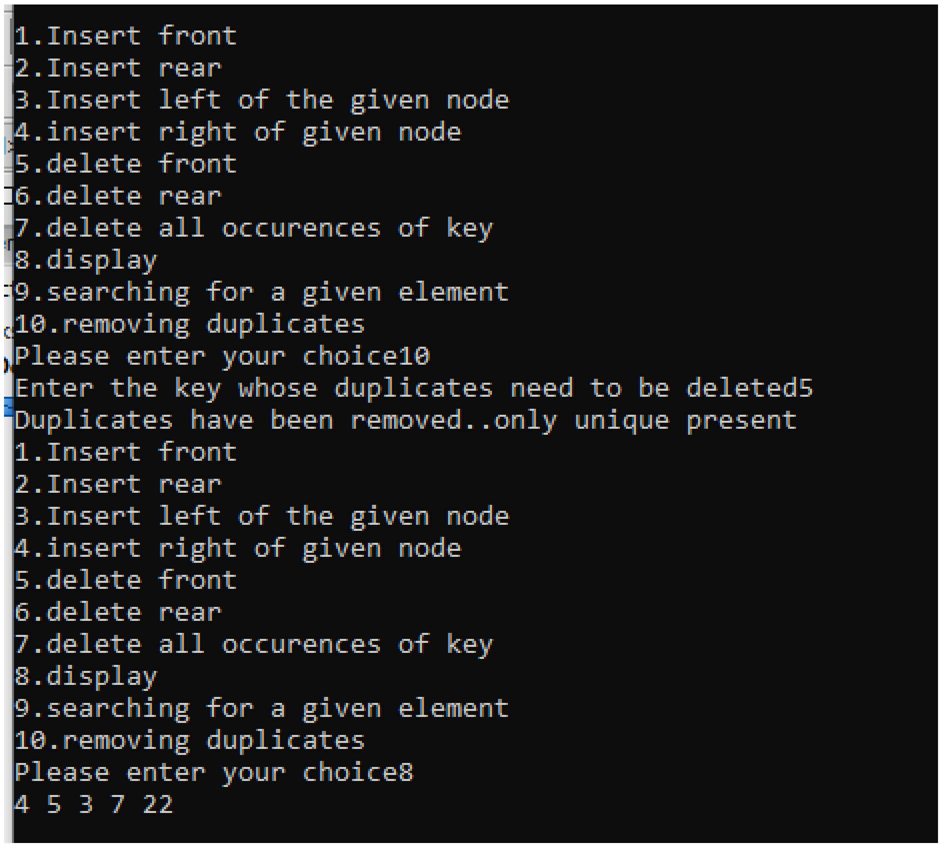
```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
22 7 3 5 5
```

9.searching for key



```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice9
Enter the value to be searched7
Item found at 5
```

10.removing duplicates



```
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice10
Enter the key whose duplicates need to be deleted5
Duplicates have been removed..only unique present
1.Insert front
2.Insert rear
3.Insert left of the given node
4.insert right of given node
5.delete front
6.delete rear
7.delete all occurences of key
8.display
9.searching for a given element
10.removing duplicates
Please enter your choice8
4 5 3 7 22
```