```
struct listADT
{
    char c;
    struct listADT * next,*prev;
};
void insertFront(struct listADT *l,char c);
void display(struct listADT *l);
void insertEnd(struct listADT *l,char c);
void insertMiddle(struct listADT *l, char c, char d);
void deleteItem(struct listADT*l, char c);
int searchItem(struct listADT*l, char c);
```

impl.h:

```
#include<stdlib.h>
#include<stdio.h>
#include "adt.h"
void insertFront(struct listADT *l,char c)
{
    struct listADT*temp=(struct listADT*)malloc(sizeof(struct listADT));
    temp->c=c;
    if(l->next==NULL)
    {
        temp->next=l->next;
        l->next=temp;
        temp->prev=l;
    }
    else
    {
        temp->next=l->next;
        l->next=temp;
        temp->prev=l;
        temp->next->prev=temp;
    }
    return;
}
void display(struct listADT *l)
{
    struct listADT*ptr=(struct listADT*)malloc(sizeof(struct listADT));
    ptr=l;
    while(ptr->next!=NULL)
    {
        printf("%c ",ptr->next->c);
```

```
        ptr=ptr->next;
    }
    printf("\nReverse order:\n");
    while(ptr!=l)
    {
        printf("%c ",ptr->c);
        ptr=ptr->prev;
    }
    return;
}

void insertEnd(struct listADT *l,char c)
{
    struct listADT*temp=(struct listADT*)malloc(sizeof(struct
listADT)),*ptr=l;
    temp->c=c;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    temp->next=ptr->next;
    temp->prev=ptr;
    ptr->next=temp;
    return;
}

void insertMiddle(struct listADT*l, char c, char d)
{
    struct listADT*ptr=l->next;
    while(ptr->c!=c)
    {
        ptr=ptr->next;
    }
    insertFront(ptr,d);
    return;
}

void deleteItem(struct listADT*l, char c)
{
    struct listADT*ptr=l,*temp;
    while(ptr->next->c!=c)
    {
        ptr=ptr->next;
    }
    temp=ptr->next;
    ptr->next=ptr->next->next;
    ptr->next->prev=ptr;
    free(temp);
```

```
    return;
}

int searchItem(struct listADT*l, char c)
{
    struct listADT*ptr=l->next;
    int count=0;
    while(ptr!=NULL)
    {
        if(ptr->c==c)
            count++;
        ptr=ptr->next;
    }
    return count;
}
```

appl.c:

```
#include<stdio.h>
#include "impl.h"
void palin(struct listADT*l)
{
    struct listADT*temp2=l,*temp1=l;
    int f=1;
    while(temp2->next!=NULL)
    {
        temp2=temp2->next;
    }
    temp1=temp1->next;
    while(temp2->next!=temp1 && temp2->next!=temp1->prev)
    {
        if(temp1->c!=temp2->c)
        {
            f=0;
            break;
        }
        else
        {
            temp1=temp1->next;
            temp2=temp2->prev;
        }
```

```
    }
    if(f==1)
        printf("It is a palindrome\n");
    else
        printf("It is not a palindrome\n");
}

void vowcon(struct listADT *h1,struct listADT*h2,struct listADT*l)
{
    l=l->next;
    while(l!=NULL)
    {
        if(l->c=='a' || l->c=='e' || l->c=='i' || l->c=='o' || l->c=='u')
            insertEnd(h1,l->c);
        else
            insertEnd(h2,l->c);
        l=l->next;
    }
}
void swap(struct listADT* l,int k)
{
    struct listADT*temp=l,ptr1,ptr2;
    while(temp->next!=NULL)
        temp=temp->next;
    int i=0;
    while(i<k)
    {
        l=l->next;
        i++;
    }
    i=1;
    while(i<k)
    {
        temp=temp->prev;
        i++;
    }
    char sw=l->c;
    l->c=temp->c;
    temp->c=sw;
}

int main()
{
    struct listADT*l=(struct listADT*)malloc(sizeof(struct listADT));
    l->next=NULL;
    char c;
    int ch;
    do
```

```
    {
        printf("\n1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete
6)search 7)exit\n");
        scanf("%d",&ch);
        if(ch==1)
        {
            printf("Enter character:");
            scanf(" %c",&c);
            insertFront(l,c);
        }
        else if(ch==2)
        {
            printf("Enter character:");
            scanf(" %c",&c);
            insertEnd(l,c);
        }
        else if(ch==3)
        {
            printf("Enter character after which character should be
inserted:");
            char d;
            scanf(" %c",&c);
            printf("Enter character to be inserted:");
            scanf(" %c",&d);
            insertMiddle(l,c,d);
        }
        else if(ch==4)
        {
            display(l);
        }
        else if(ch==5)
        {
            printf("Enter character that needs to be deleted:");
            scanf(" %c",&c);
            deleteItem(l,c);
        }
        else if(ch==6)
        {
            printf("Enter character to count its no.of occurrences:");
            scanf(" %c",&c);
            printf("count:%d",searchItem(l,c));
        }
    }while(ch!=7);

    printf("Applications:");
    printf("\nChecking palindrome or not:");
    palin(l);
```

```
    struct listADT *h1=(struct listADT *)malloc(sizeof(struct
listADT)),*h2=(struct listADT *)malloc(sizeof(struct listADT));
    h1->next=NULL;
    h2->next=NULL;
    vowcon(h1,h2,l);
    printf("\nVowels:");
    display(h1);
    printf("\nConsonants:");
    display(h2);
    int k;
    printf("\nEnter k to swap nodes' values:\n");
    scanf(" %d",&k);
    swap(l,k);
    display(l);
    return 0;
}
```

o/p:

```
1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
2
Enter character:b

1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
1
Enter character:a

1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
2
Enter character:b

1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
4
a b b
Reverse order:
b b a
1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
3
Enter character after which character should be inserted:b
Enter character to be inserted:c

1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
2
Enter character:a

1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
4
a b c b a
Reverse order:
a b c b a
```

```
1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
6
Enter character to count its no.of occurrences:a
count:2
1)insertfront 2)insertend 3)insertmiddle 4)display 5)delete 6)search 7)exit
7
Applications:
Checking palindrome or not:It is not a palindrome

Vowels:a a
Reverse order:
a a
Consonants:c b
Reverse order:
b c
Enter k to swap nodes' values:
2
a b c a
Reverse order:
a c b a
```