**Exercise 3: ListADT and its Applications**

**dlADT.h:**

struct node

{

char c;

struct node \*next;

struct node \*prev;

};

struct node \* insertFront(struct node \*L, char c);

struct node \* insertEnd(struct node \*L, char c);

struct node \* insertMiddle(struct node \*L, char c, char d);

void displayItems(struct node \*L);

struct node \* deleteItem(struct node \*L, char c);

void seperation(struct node \*,struct node \*\*C,struct node \*\*V);

int searchItem(struct node\*, char c);

int palindrome(struct node \*header);

struct node \* swapfl(struct node \*header);

void swapk(struct node \*header,int k,int count);

**dlimpl.h:**

#include<stdio.h>

#include<stdlib.h>

#include "dlADT.h"

struct node \* insertFront(struct node \*header, char c)

{

struct node \*ptr=(struct node \*)malloc(sizeof(struct node));

ptr->c=c;

if(header==NULL)

ptr->next=NULL;

else

{

ptr->next=header;

header->prev=ptr;

}

ptr->prev=NULL;

header=ptr;

return header;

}

struct node \* insertEnd(struct node \*header, char c)

{

struct node \*ptr=(struct node \*)malloc(sizeof(struct node));

ptr->c=c;

if(header==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

header=ptr;

}

else

{

struct node \*temp=header;

while(temp->next!=NULL)

{temp=temp->next;}

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

return header;

}

struct node \* insertMiddle(struct node \*header, char c, char d)

{

struct node \*ptr=(struct node \*)malloc(sizeof(struct node));

ptr->c=d;

if(header==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

header=ptr;

}

else

{

struct node \*temp=header;

while(temp->next!=NULL&&temp->c!=c)

{temp=temp->next;}

if(temp->next!=NULL)

{

ptr->next=temp->next;

ptr->prev=temp;

temp->next->prev=ptr;

temp->next=ptr;

}

else

{

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

}

return header;

}

void displayItems(struct node \*L)

{

struct node \*ptr=L;

if(ptr==NULL)

printf("\nList Empty\n");

else

{

printf("\nForward:\n");

while(ptr->next!=NULL)

{

printf("%c ",ptr->c);

ptr=ptr->next;

}

printf("%c ",ptr->c);

//ptr=ptr->next;

printf("\nRev:\n");

while(ptr!=NULL)

{

printf("%c ",ptr->c);

ptr=ptr->prev;

}

}

}

struct node \* deleteItem(struct node \*header, char c)

{

struct node \*ptr=header;

if(ptr==NULL)

printf("\nEmpty List\n");

else

{

/\*if(ptr->c==c)

{

header=ptr->next;

header->prev=NULL;

free(ptr);

return header;

}

ptr=ptr->next;

while(ptr->c!=c)

{

}\*/

while(ptr->c!=c&&ptr->next!=NULL)

{

ptr=ptr->next;

}

if(ptr->next==NULL&&ptr->c!=c)

{

printf("\nElement not found\n");

}

else{

if(header==ptr)

header=ptr->next;

if(ptr->next!=NULL)

ptr->next->prev=ptr->prev;

if(ptr->prev!=NULL)

ptr->prev->next=ptr->next;

free(ptr);

}

}

return header;

}

void seperation(struct node \*header,struct node \*\*C,struct node \*\*V)

{

struct node \*ptr=header;

/\*if(ptr==NULL)

printf("\nEmpty List\n");\*/

while(ptr!=NULL)

{

if(ptr->c=='a'||ptr->c=='e'||ptr->c=='i'||ptr->c=='o'||ptr->c=='u'||ptr->c=='A'||ptr->c=='E'||ptr->c=='I'||ptr->c=='O'||ptr->c=='U')

\*V=insertEnd(\*V,ptr->c);

else

\*C=insertEnd(\*C,ptr->c);

ptr=ptr->next;

}

}

int searchItem(struct node \*header, char c)

{

struct node \*ptr=header;

int count=0;

if(ptr==NULL)

printf("\nEmpty List,Cant search\n");

else

{

while(ptr!=NULL)

{

if(ptr->c==c)

count++;

ptr=ptr->next;

}

}

return count;

}

int palindrome(struct node \*header)

{

struct node \*start=header;

struct node \*tail=header;

if(header==NULL)

{

printf("\nEmpty List\n");

return -1;

}

while(tail->next!=NULL)

{

tail=tail->next;

}

while(start!=tail && tail->next!=start)

{

if(start->c!=tail->c)

return 0;

start=start->next;

tail=tail->prev;

}

return 1;

}

int total(struct node \*header)

{

int count=1;

struct node \*ptr=header;

while(ptr->next)

{

ptr=ptr->next;

count++;

}

return count;

}

struct node\* swapfl(struct node \*header)

{

struct node \*start=header;

struct node \*tail=header;

while(tail->next)

{tail=tail->next;}

tail->prev->next=start;

start->prev=tail->prev;

tail->next=start->next;

start->next->prev=tail;

tail->prev=NULL;

start->next=NULL;

return tail;

}

void swapk(struct node \*header,int k,int count)

{

struct node \*p=header;

struct node \*q=header;

int i=0;

while(q->next)

{

q=q->next;

}

if(k>count)

{

printf("\nInvalid value for k!\n");

return;

}

if(count%2?k==count/2+1:0)

return;

if(k>count/2)

k=count-k+1;

while(i<k-1)

{

p=p->next;

q=q->prev;

i++;

}

if(p->next==q)

{

p->next=q->next;

q->next->prev=p;

p->prev->next=q;

q->prev=p->prev;

q->next=p;

p->prev=q;

return;

}

struct node \*p1=p->prev;

struct node \*p2=p->next;

struct node \*q1=q->prev;

struct node \*q2=q->next;

p1->next=q;

q->prev=p1;

q->next=p2;

p2->prev=q;

q1->next=p;

p->prev=q1;

p->next=q2;

q2->prev=p;

}

**dlappl.c:**

#include<stdio.h>

#include "dlimpl.h"

int main()

{

struct node \*L=NULL;

int ch;

char c;

do

{

printf("\n\nMenu:\n1.Insert At Front\n2.Insert At End\n3.Insert in Middle\n4.Display List\n5.Delete first occurance\n6.Count the occurances\n7.Check whether palindrome\n8.Seperate into Consonants and Vowels\n9.Swap kth Node\n10.Exit\nEnter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter a char: ");

scanf(" %c",&c);

L=insertFront(L,c);

displayItems(L);

break;

case 2: printf("\nEnter a char: ");

scanf(" %c",&c);

L=insertEnd(L,c);

displayItems(L);

break;

case 3: printf("\nEnter the char after which it needs to be inserted: ");

char d;

scanf(" %c",&d);

printf("Enter a char: ");

scanf(" %c",&c);

L=insertMiddle(L,d,c);

displayItems(L);

break;

case 4: displayItems(L);

break;

case 5: printf("\nEnter a char to delete: ");

scanf(" %c",&c);

L=deleteItem(L,c);

displayItems(L);

break;

case 6: printf("\nEnter a char:");

scanf(" %c",&c);

int count=searchItem(L,c);

displayItems(L);

printf("\nCount of %c: %d\n",c,count);

break;

case 7: ;

int isP=palindrome(L);

displayItems(L);

if(isP==1)

printf("\nIt is a Palindrome!\n");

else if(isP==0)

printf("\nIt is not a Palindrome!\n");

break;

case 8: ;struct node \*C=NULL,\*V=NULL;

seperation(L,&C,&V);

displayItems(L);

printf("\n\nConst:\n");

displayItems(C);

printf("\n\nVowels:\n");

displayItems(V);

break;

case 9: printf("\nEnter value of k: ");

int k;

scanf("%d",&k);

int n=total(L);

if(k==1||k==n)

L=swapfl(L);

else

swapk(L,k,n);

displayItems(L);

break;

case 10:break;

default:printf("\nInvalid Input!\n");

}

}while(ch!=10);

/\*displayItems(L);

L=insertMiddle(L,'a','e');

displayItems(L);

L=insertEnd(L,'5');

displayItems(L);

L=insertFront(L,'a');

displayItems(L);

//printf("%c",L->c);

L=insertFront(L,'z');

displayItems(L);

L=insertEnd(L,'x');

displayItems(L);

L=insertMiddle(L,'5','E');

displayItems(L);

L=insertMiddle(L,'5','e');

displayItems(L);

L=insertMiddle(L,'Z','e');

displayItems(L);

L=insertMiddle(L,'Z','W');

displayItems(L);

//printf("%c",L->c);

//displayItems(L);

L=insertFront(L,'U');

displayItems(L);

L=insertEnd(L,'C');

displayItems(L);

L=insertFront(L,'g');

displayItems(L);

L=deleteItem(L,'z');

displayItems(L);

L=deleteItem(L,'5');

displayItems(L);

L=deleteItem(L,'W');

displayItems(L);

displayItems(L);

struct node \*C=NULL,\*V=NULL;

seperation(L,&C,&V);

printf("\n\nConst:\n");

displayItems(C);

printf("\n\nVowels:\n");

displayItems(V);

int count=searchItem(L,'p');

printf("\nCount of %c: %d\n",'p',count);

L=insertFront(L,'m');

L=insertFront(L,'A');

L=insertFront(L,'m');

displayItems(L);

int isP=palindrome(L);

if(isP==1)

printf("\nIt is a Palindrome!\n");

else if(isP==0)

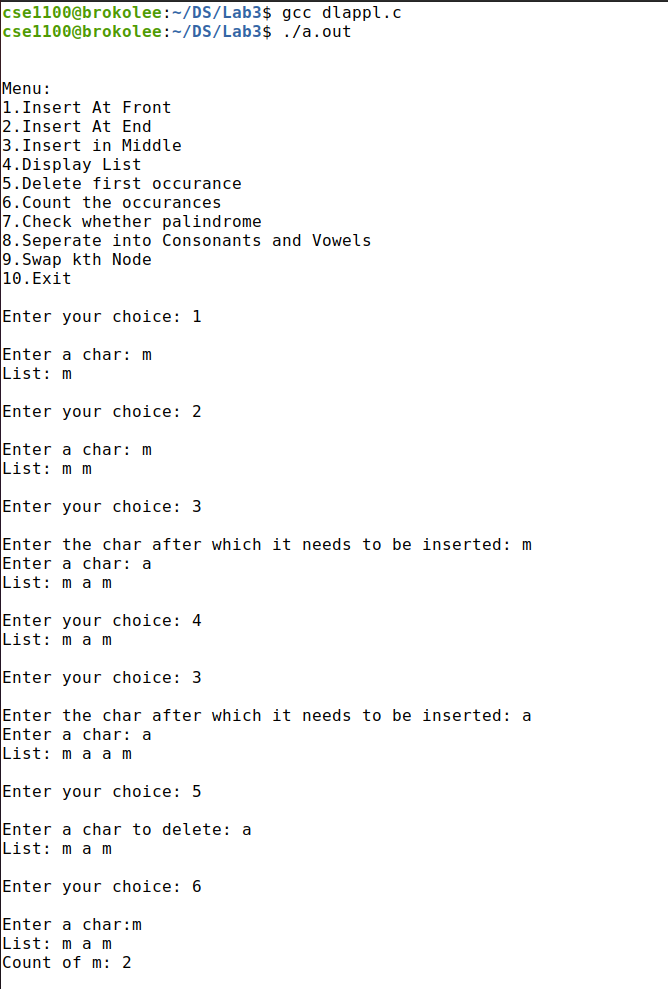
printf("\nIt is not a Palindrome!\n");\*/

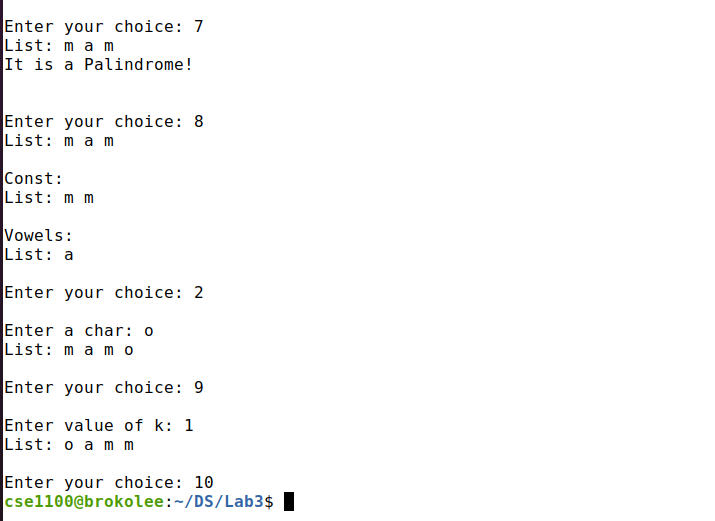
return 0;

}

**Sample I/O:**

**(new one):**

****

****

**(old one):**

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 1

Enter a char: s

Forward:

s

Rev:

s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 2

Enter a char: i

Forward:

s i

Rev:

i s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 3

Enter the char after which it needs to be inserted: s

Enter a char: h

Forward:

s h i

Rev:

i h s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 4

Forward:

s h i

Rev:

i h s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 5

Enter a char to delete: h

Forward:

s i

Rev:

i s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 2

Enter a char: s

Forward:

s i s

Rev:

s i s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 2

Enter a char: c

Forward:

s i s c

Rev:

c s i s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 6

Enter a char:s

Forward:

s i s c

Rev:

c s i s

Count of s: 2

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 7

Forward:

s i s c

Rev:

c s i s

It is not a Palindrome!

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 5

Enter a char to delete: c

Forward:

s i s

Rev:

s i s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 7

Forward:

s i s

Rev:

s i s

It is a Palindrome!

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 1

Enter a char: a

Forward:

a s i s

Rev:

s i s a

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 8

Forward:

a s i s

Rev:

s i s a

Const:

Forward:

s s

Rev:

s s

Vowels:

Forward:

a i

Rev:

i a

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 9

Enter value of k: 1

Forward:

s s i a

Rev:

a i s s

Menu:

1.Insert At Front

2.Insert At End

3.Insert in Middle

4.Display List

5.Delete first occurance

6.Count the occurances

7.Check whether palindrome

8.Seperate into Consonants and Vowels

9.Swap kth Node

10.Exit

Enter your choice: 10