

Data Structures Laboratory
Assignment-3

ListADT and its Applications – Doubly linked lists

Name: Sai Shashaank R

Register No: 205001086

Class: CSE-B

adt.h

```
struct listADT
```

```
{
```

```
    char item;
```

```
    struct listADT *next;
```

```
    struct listADT *prev;
```

```
};
```

```
void insertFront(struct listADT *l, char c);
```

```
void insertEnd(struct listADT *l, char c);
```

```
void insertMiddle(struct listADT *l, char c, char d);
```

```
void displayItems(struct listADT *l);
```

```
void deleteItem(struct listADT *l, char c);
```

```
int searchItem(struct listADT *l, char c);
```

```
/*-----Extra functions-----*/
```

```
void init (struct listADT *l); //To initialise the header of a list
```

```
int listCompare(struct listADT *l1, struct listADT *l2); //To compare two doubly linked lists
```

```
void input(struct listADT *l); //To take input for the doubly linked list
```

impl.h

```
#include "adt.h"
```

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

```
#include <stdlib.h>
```

```
void init(struct listADT *l)
```

```
{  
    l->next = NULL;  
    l->prev = NULL;  
}
```

```
void insertFront(struct listADT *l, char c)
```

```
{  
    struct listADT *temp = (struct listADT *) malloc(sizeof(struct listADT));  
    temp->item = c;  
    temp->next = l->next;  
  
    if(l->next != NULL)  
        l->next->prev = temp;  
  
    l->next = temp;  
    temp->prev = l;  
}
```

```
void insertEnd(struct listADT *l, char c)
```

```
{  
    struct listADT *temp = (struct listADT *) malloc(sizeof(struct listADT));  
  
    temp->item = c;  
    temp->next = NULL;  
  
    if(l->next == NULL)  
    {  
        l->next = temp;  
        temp->prev = l;  
    }
```

```
}
```

```
else
```

```
{
```

```
    struct listADT *ptr = l;
```

```
    while(ptr->next!=NULL)
```

```
        ptr = ptr->next;
```

```
    ptr->next = temp;
```

```
    temp->prev = ptr;
```

```
}
```

```
}
```

```
void insertMiddle(struct listADT *l,char c,char d)
```

```
{
```

```
    int flag = 0;
```

```
    struct listADT *ptr = l->next;
```

```
    while(ptr!=NULL)
```

```
    {
```

```
        if(ptr->item == c)
```

```
        {
```

```
            flag = 1;
```

```
            break;
```

```
        }
```

```
        ptr = ptr->next;
```

```
    }
```

```
    if(flag == 1)
```

```
    {
```

```
        struct listADT *temp = (struct listADT *) malloc(sizeof(struct listADT));
```

```
        temp->item = d;
```

```
        temp->next = ptr->next;
```

```

        if(ptr->next != NULL)
            ptr->next->prev = temp;

        ptr->next = temp;
        temp->prev = ptr;
        printf("\nElement has been inserted.");
    }

    else
        printf("\nERROR: Element not found.");
}

```

```

void displayItems(struct listADT *l)
{
    if(l->next == NULL)
        printf("\nList is empty.");

    else
    {
        struct listADT *ptr = l->next;
        while(ptr!=NULL)
        {
            printf("%c",ptr->item);
            ptr=ptr->next;
        }
    }
}

```

```

void deleteItem(struct listADT *l,char c)
{
    int flag = 0;
    struct listADT *ptr = l->next;

    while(ptr!=NULL)
    {

```

```

        if(ptr->item == c)
        {
            flag = 1;
            break;
        }

        ptr = ptr->next;
    }

    if(flag == 1)
    {
        ptr->prev->next = ptr->next;
        if(ptr->next != NULL)
            ptr->next->prev = ptr->prev;

        printf("%c has been deleted successfully!",ptr->item);
        free(ptr);
    }

    else
        printf("\nItem wasn't found.");
}

```

```

int searchItem(struct listADT *l,char c)
{
    int counter = 0;
    struct listADT *ptr = l->next;

    while(ptr!=NULL)
    {
        if(ptr->item == c)
            counter++;

        ptr = ptr->next;
    }
}

```

```

        return counter;
    }

int listCompare(struct listADT *l1, struct listADT *l2)
{
    int flag = 0;

    struct listADT *p1 = l1->next;
    struct listADT *p2 = l2->next;

    while(p1!=NULL && p2!=NULL)
    {
        if(p1->item != p2->item)
        {
            flag = 1;
            break;
        }

        p1 = p1->next;
        p2 = p2->next;
    }

    if(p1 != NULL)
        flag = 1;

    else if(p2 != NULL)
        flag = 1;

    return flag;
}

void input(struct listADT *l)
{
    char str[50];

```

```
printf("\nEnter the string: ");  
scanf(" %s",str);  
for(int i=0; str[i]!='\0'; i++)  
{  
    insertEnd(l,str[i]);  
}  
}
```

appl.c

```
#include "impl.h"
```

```
void main()
```

```
{
```

```
    struct listADT *l;
```

```
    char option;
```

```
    int choice;
```

```
    int k;
```

```
    struct listADT *ptr;
```

```
    struct listADT *temp;
```

```
    /*To store consonants*/
```

```
    struct listADT *c;
```

```
    c = (struct listADT *)malloc(sizeof(struct listADT));
```

```
    init(c);
```

```
    /*To store vowels*/
```

```
    struct listADT *v;
```

```
    v = (struct listADT *)malloc(sizeof(struct listADT));
```

```
    init(v);
```

```
    do
```

```
    {
```

```
        l = (struct listADT *)malloc(sizeof(struct listADT));
```

```
        init(l);
```

```
        input(l); //Taking input for l
```

```
        printf("\n1) Check whether the list is palindrome or not.");
```

```
        printf("\n2) Create separate lists containing vowels and consonants from the list.");
```

```
        printf("\n3) Swap kth node from the beginning with kth node from the end.");
```

```
        printf("\n4) Number of times a letter occurs in a list.");
```



```

printf("\n5) Insert a character in the middle.");
printf("\n6) Delete a character.");
printf("\n7) Exit.");
printf("\nEnter your choice: ");
scanf("%d",&choice);

switch(choice)
{
    case 1: ptr = l->next;
            temp = (struct listADT *)malloc(sizeof(struct listADT));
            init(temp);

            while(ptr!=NULL)
            {
                insertFront(temp,ptr->item);
                ptr = ptr->next;
            }

            if(listCompare(temp,l)==0)
            {
                printf("\n\n");
                displayItems(l);
                printf(" is a palindrome.");
            }

            else
            {
                printf("\n");
                displayItems(l);
                printf(" is NOT a palindrome.");
            }

            break;

    case 2: ptr = l->next;
            while(ptr!=NULL)

```

```

        {
            if(ptr->item == 'A' || ptr->item == 'E' || ptr->item == 'T' || ptr-
>item == 'O' || ptr->item == 'U' )
                insertEnd(v,ptr->item);

            else
                insertEnd(c,ptr->item);

            ptr = ptr->next;
        }

```

```

printf("\nVowels: ");
displayItems(v);

```

```

printf("\nConsonants: ");
displayItems(c);
break;

```

```

case 3: printf("\nEnter the value of k: ");
scanf("%d",&k);

```

```

int counter = 1;
struct listADT *pstart, *pend;

```

```

pend = l->next;
pstart = l->next;

```

```

while(pend->next!=NULL)
    pend = pend->next;

```

```

while(counter != k)
{
    pstart = pstart->next;
    pend = pend->prev;
    counter++;
}

```

```
}
```

```
char temp = pstart->item;  
pstart->item = pend->item;  
pend->item = temp;
```

```
printf("\nList after swapping is ");  
displayItems(l);  
break;
```

```
case 4: printf("\nEnter the character: ");  
char item;
```

```
scanf(" %c",&item);  
printf("\nNo of occurrences = %d",searchItem(l,item));  
break;
```

```
case 5: printf("\nEnter the character after which a character has to be inserted:  
");
```

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

```
printf("\nEnter the character to be inserted: ");  
char t;
```

```
scanf(" %c",&t);
```

```
insertMiddle(l,item,t);  
printf("\nNew list = ");  
displayItems(l);  
break;
```

```
case 6: printf("\nEnter the character to be deleted: ");  
scanf(" %c",&item);
```

```
deleteItem(l,item);
```

```
printf("\nNew list = ");  
displayItems(l);  
break;
```

```
case 7: exit(0);
```

```
default: printf("\nWrong choice. Try again.");
```

```
}
```

```
printf("\nDo you want to continue?(y/n): ");
```

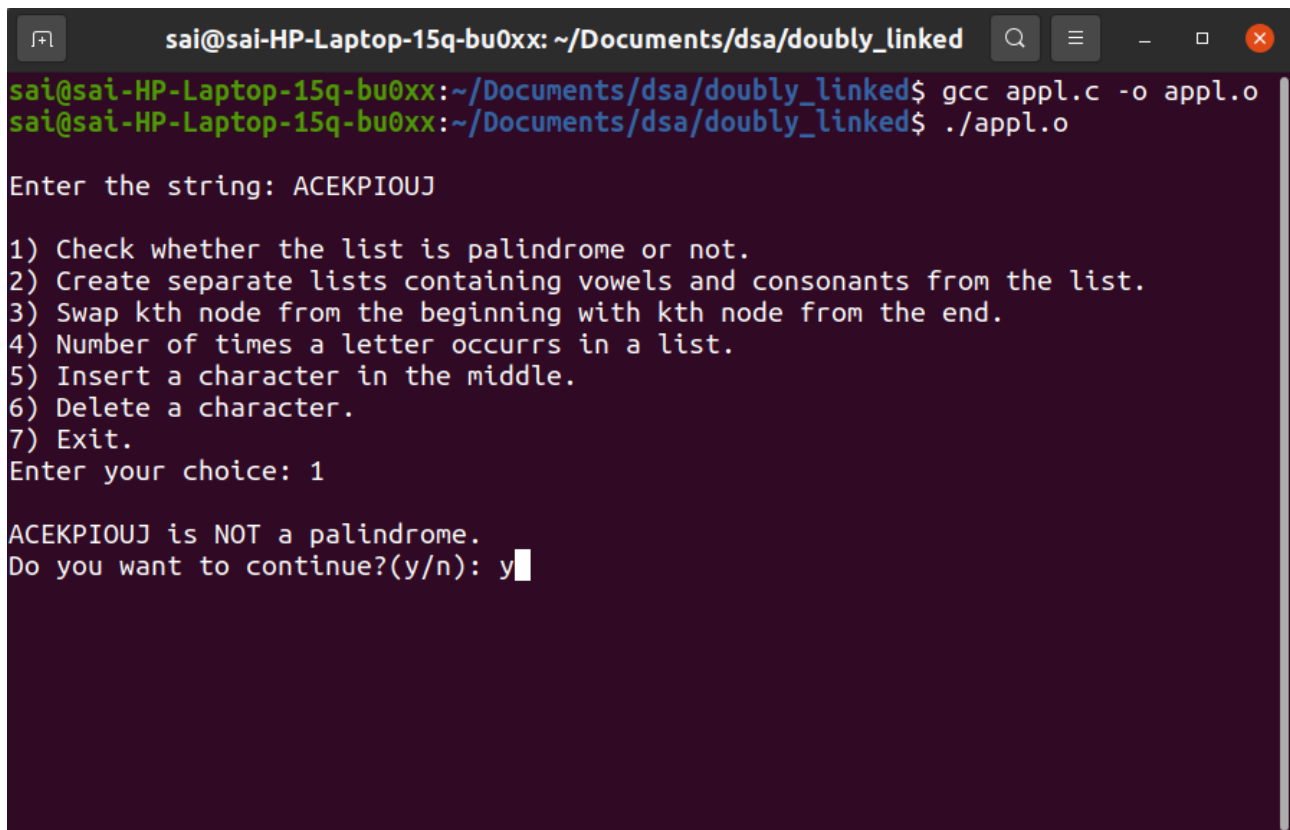
```
scanf(" %c",&option);
```

```
}while(option == 'y');
```

```
}
```

Screenshots

1) Testing if a string is a palindrome.

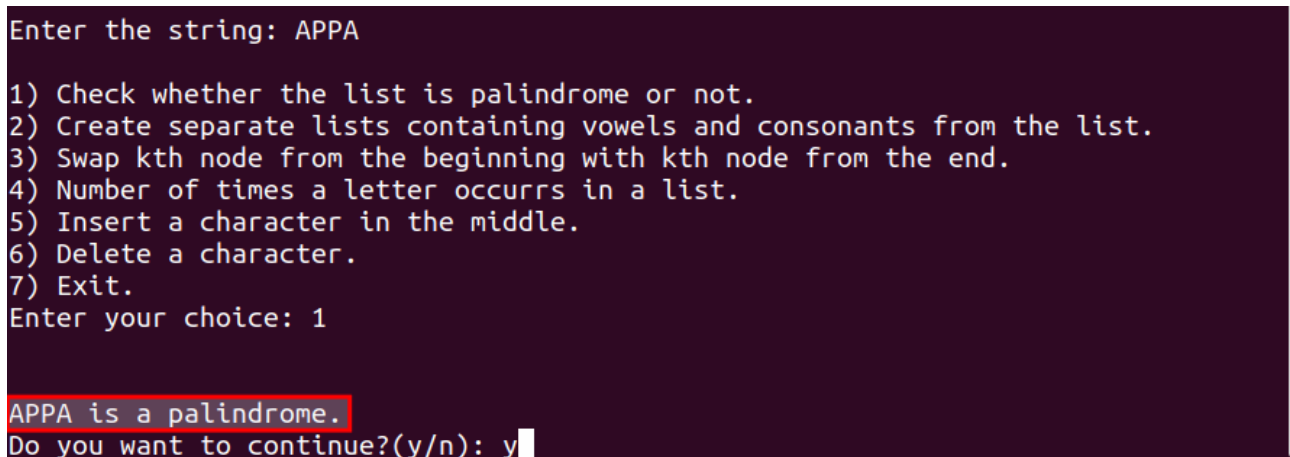


```
sai@sai-HP-Laptop-15q-bu0xx: ~/Documents/dsa/doubly_linked
sai@sai-HP-Laptop-15q-bu0xx:~/Documents/dsa/doubly_linked$ gcc appl.c -o appl.o
sai@sai-HP-Laptop-15q-bu0xx:~/Documents/dsa/doubly_linked$ ./appl.o

Enter the string: ACEKPIOUJ

1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 1

ACEKPIOUJ is NOT a palindrome.
Do you want to continue?(y/n): y
```



```
Enter the string: APPA

1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 1

APPA is a palindrome.
Do you want to continue?(y/n): y
```

2) Create separate list containing vowels and consonants.

```
Enter the string: ACEKPIOUJ

1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 2

Vowels: AEIOU
Consonants: CKPJ
Do you want to continue?(y/n): y
```

3) Swap kth node from beginning with kth node from end

```
Enter the string: ACEKPIOUJ

1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 3

Enter the value of k: 2

List after swapping is AUEKPIOCJ
Do you want to continue?(y/n): y
```

4) Number of times a letter occurs in a list

```
Enter the string: SHASHAANK

1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 4

Enter the character: A

No of occurrences = 3
Do you want to continue?(y/n): y
```

5) Insert a character in the middle

```
Enter the string: TESTING
1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 5
Enter the character after which a character has to be inserted: S
Enter the character to be inserted: A
Element has been inserted.
New list = TESATING
Do you want to continue?(y/n): y
```

6) Delete a character

```
Enter the string: TESTING
1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 6
Enter the character to be deleted: S
S has been deleted successfully!
New list = TETING
Do you want to continue?(y/n): y
```

7) Exit

```
Enter the string: test
1) Check whether the list is palindrome or not.
2) Create separate lists containing vowels and consonants from the list.
3) Swap kth node from the beginning with kth node from the end.
4) Number of times a letter occurs in a list.
5) Insert a character in the middle.
6) Delete a character.
7) Exit.
Enter your choice: 7
sai@sai-HP-Laptop-15q-bu0xx:~/Documents/dsa/doubly_linked$
```