Experiment 11
Date 01-12-2022

LINKED LIST

Aim

Write a menu driven C program for performing the following operations on a singly Linked List:

a. Insert at Beginning b. Insert at End c. Insert at a specified Position d. Delete from a specified Position e. Delete from Beginning f. Delete from End

1 Linked List

1.1 Algorithm

```
1. Start
2. Create a structure list having fields data and link
3. Allocate memory for header node
4. Print ’’Menu’’
Print ’’1. Insert at Beginning’’
Print ’’2. Insert at End’’
Print ’’3. Insert at a specified Position’’
Print ’’4. Delete from a specified Position’’
Print ’’5. Delete from Beginning’’
Print ’’6. Delete from End’’
Print ’’7. Exit’’
5. Input ch
6. If ch = 1, then insertFront(), go to step 2
7. If ch = 2, then insertRear(), go to step 2
8. If ch = 3, then insertPos(), go to step 2
9. If ch = 4, then deletePos(), go to step 2
10. If ch = 5, then deleteFront(), go to step 2
11. If ch = 6, then deleteRear(), go to step 2
12. Stop


Start of function insertFront()
1. Allocate memory for new node
2. Input new → data
3. Let new → link ← head → link and head → link ← new
```

```
4. Return
Start of function insertRear()
1. Allocate memory for new node
2. Input new → data
3. Let ptr ← header
4. If ptr → link = NULL, go to step 6
5. Let ptr ← ptr → link
6. Let new → link ← NULL and ptr → link ← new
7. Return
Start of function insertPos()
1. Input key
2. Let ptr ← header
3. If ptr → data = key or ptr → link = NULL), go to step 5
4. Let ptr ← ptr → link
5. If ptr → data != key, go to step 10
6. Allocate memory for new node
7. Input new → data
8. Let new → link ← ptr → link2
9. Let ptr → link ← n
10. Print ”Key not found”
11. Return
Start of function deletePos()
1. Input key
2. If header → link = NULL, Return
3. Let ptr ← header
4. If ptr → data = key or ptr → link = NULL), go to step 7
5. Let temp ← ptr
6. Let ptr ← ptr → link
7. If ptr → data != key, go to step 10
8. Print ptr → data
9. Let temp → link ← ptr → link
10. Print ”Key not found”
11. Return
Start of function deleteFront()
1. If header → link = NULL, Return
2. Let ptr ← header → link
3. Print ptr → data
4. Let ptr1 ← ptr → link
5. Let header → link ← ptr1
6. Return
Start of function deleteRear()
1. If header → link = NULL, Return
2. Let ptr ← header
3. If ptr → link = NULL, go to step 6
4. Let temp ← ptr
5. Let ptr ← ptr → link
```

```
6. Print ptr → data
7. Let temp → link ← NULL
8. Return
```

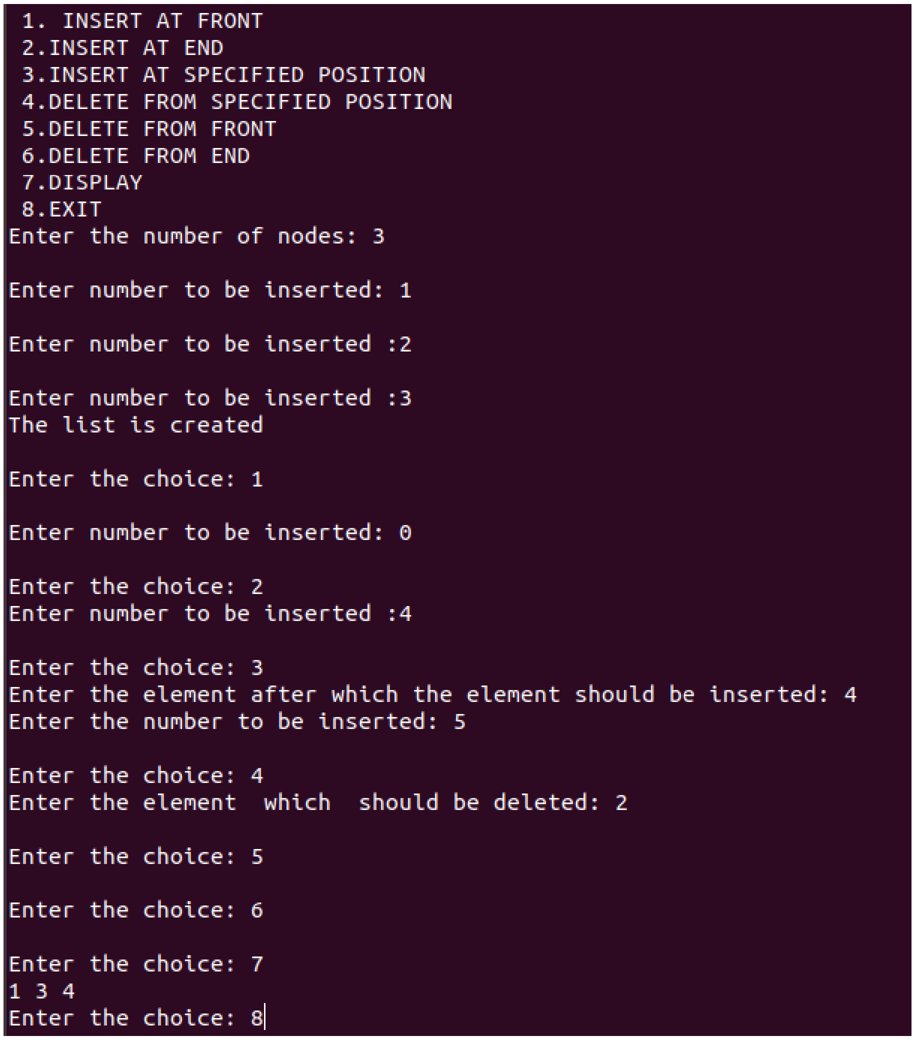
1.2 Program

```
#include <stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node* link;
};
struct node* start = NULL;
void createlist()
{
if(start==NULL)
{
int n;
printf("\nEnter the number of nodes: ");
    scanf("%d",&n);
    if(n!=0)
    {
     int data;
     struct node* newnode;
     struct node* temp;
     newnode= malloc(sizeof(struct node));
     start=newnode;
     temp=start;
     printf("\nEnter number to be inserted: ");
     scanf("%d",&data);
     start->info = data;
     for(int i=2;i<=n;i++)
     {
     newnode = malloc(sizeof(struct node));
     temp->link =newnode;
     printf("\nEnter number to be inserted :");
     scanf("%d",&data);
     newnode->info=data;
     temp=temp->link;

     }
    temp->link=NULL;
    printf("The list is created\n");
    }
   }
  else
   printf("\nThe list is already created\n");
}

void insertfront()
{
int data;
struct node* temp;
temp= malloc(sizeof(struct node));
printf("\nEnter number to be inserted: ");
scanf("%d",&data);
temp->info=data;
temp->link=start;
start=temp;
}

void insertend()
{
int data;
struct node *temp, *head;
temp=malloc(sizeof(struct node));
printf("Enter number to be inserted :");
scanf("%d",&data);
temp->link=NULL;
temp->info=data;
head=start;
while(head->link!=NULL)
head=head->link;
head->link=temp;

}
void insertpos()
{
int data,key;
struct node *temp, *head;
temp=(struct node*)malloc(sizeof(struct node));
printf("Enter the element after which the element should be inserted: ");
scanf("%d",&key);
printf("Enter the number to be inserted: ");
scanf("%d",&data);
head=start;
while(head->info!=key && head->link!=NULL)
head=head->link;
```

```
if(head->link==NULL && head->info!=key)
printf("The key is not available in the list\n");
else
temp->link=head->link;
temp->info=data;
head->link=temp;
}

void deletefront()
{
struct node* temp;
if(start==NULL)
printf("The liked list is empty\n");
else
{
temp=start;
start=start->link;
}

}
void deleteend()
{
struct node *temp, *prevnode;
if(start==NULL)
printf("\nList is empty\n");
else {
temp=start;
while(temp->link!=NULL)
{
prevnode=temp;
temp=temp->link;
}
}
prevnode->link=NULL;
}

void deletepos()
{
struct node *temp, *prev;
int i=1,key;
if(start==NULL)
printf("List is empty\n");
printf("Enter the element  which  should be deleted: ");
scanf("%d",&key);
temp=start;
while(temp->info!=key && temp->link!=NULL)
```

```
{prev=temp;
temp=temp->link;
}
if(temp->link==NULL && temp->info!=key)
printf("The key is not available in the list\n");
else
{
 prev->link=temp->link;
}
}
void display()
{
struct node *temp;
temp=start;
while(temp->link!=NULL)
{printf("%d ",temp->info);
temp=temp->link;
}
printf("%d ",temp->info);
}


void main()
{
  int choice;
   printf("\n 1. INSERT AT FRONT\n 2.INSERT AT END\n 3.INSERT AT SPECIFIED POSITION\n 4.DELE
   printf("\n 5.DELETE FROM FRONT\n 6.DELETE FROM END\n 7.DISPLAY\n 8.EXIT");
  createlist();
  do
  {
   printf("\nEnter the choice: ");
   scanf("%d",&choice);
   switch(choice)
   {
   case 1:insertfront();break;
   case 2:insertend();break;
   case 3:insertpos();break;
   case 4:deletepos();break;
   case 5:deletefront();break;
   case 6:deleteend();break;
   case 7:display();break;
   case 8:return ;break;
   default:printf("INVALID INPUT\n"); break;
   }
  }while(choice!=8);
}
```

1.3 Sample Output



```
 1. INSERT AT FRONT
 2.INSERT AT END
 3.INSERT AT SPECIFIED POSITION
 4.DELETE FROM SPECIFIED POSITION
 5.DELETE FROM FRONT
 6.DELETE FROM END
 7.DISPLAY
 8.EXIT
Enter the number of nodes: 3

Enter number to be inserted: 1

Enter number to be inserted :2

Enter number to be inserted :3
The list is created

Enter the choice: 1

Enter number to be inserted: 0

Enter the choice: 2
Enter number to be inserted :4

Enter the choice: 3
Enter the element after which the element should be inserted: 4
Enter the number to be inserted: 5

Enter the choice: 4
Enter the element  which  should be deleted: 2

Enter the choice: 5

Enter the choice: 6

Enter the choice: 7
1 3 4
Enter the choice: 8
```

Figure 1: Input and Output

1.4 Result

Implemented linked list using a self referential structure with data in it. A function named createlist() creates a linked list with a given number of nodes taken from the user. insertfront(),insertend(),insertpos() inserts elements to the front,end and at given position in the linked list. deletefront(),deleteend(),deletepos() deletes elements from the front,end and from a given position in the linked list.