PROGRAM 5

Design, develop and implement a Program inC for the following StackApplications

a. Evaluation of Suffixexpression withsingle digit operandsand operators: +, -, *, /,%, ^

b. Solving Tower of Hanoi problem with n disks

Program objective :

- Understand different polish notation.
- Understand the methodology of evaluating suffix expression.
- Get the knowledge of operator precedence and associativity.

Algorithm

Step 1: Read the suffix/postfix expression

Step 2: Scan the postfix expression from left to right character by character

Step 3: if scanned symbol is operand push data into stack.

If scanned symbol is operator pop two elements from stack Evaluate result and result is pushed onto stack

Step 4: Repeat step 2-3 until all symbols are scanned completely

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define MAX 50
char post[MAX];
int stack[MAX],top=-1,i;
void pushstack(int);
void calculator(char);
main()
{
 printf("enter suffix expression\n");
 gets(post);
 for(i=0; i<strlen(post); i++)
 {
  if(post[i]>'0'&& post[i]<='9')
  pushstack(i);
  else
   calculator(post[i]);
  }
   printf("result=%d\n",stack[top]);
 }
void pushstack(int i)
 {
   top=top+1;
   stack[top]=(int)(post[i]-48);
 }
void calculator(char c)
 {
  int a,b,ans;
  b=stack[top--];
  a=stack[top--];
```

```
 switch(c)
 {
  case '+':ans=a+b;break;
  case '-':ans=a-b;break;
  case '*':ans=a*b;break;
  case '/':ans=a/b;break;
  case '%':ans=a%b;break;
  case '^':ans=pow(a,b);break;
  default :printf("wrong input\n");
  exit(0);
 }
  top++;
  stack[top]=ans;
}
```

Output1

enter suffix expression:
23+
The result is 5

Output2

enter suffix expression:
123-4*+
The result is -3.

Output3

enter suffix expression:

623+-382/+*2$3+

The result is 52

Program outcome:

- Identify the applications of suffix expression.
- Familiarized with the methodology of suffix evaluation.
- Familiarized the operator precedence and associativity.

Viva Questions

- What is Suffix Expression?

5 b. Solving Tower of Hanoi problem with n disks

Program objective:

- Understand tower of Hanoi problem.
- Understand recursive functions and its disadvantages.

Algorithm:

MAIN FUNCTION ()

Step 1: Read No of disks called n from keyboard.

Step 2: Check if n is not zero or a negative no. if yes display suitable message else go to step3.

Step 3: Call tower of Hanoi function with n asparameter,

Step 4: Stop

TOWERS OF HANOI FUNCTION TO MOVE DISKS FROM A TO CUSING B ()

Step 1: If n is equal to 1 then move the single disk from A to C and stop

Step 2: Move the top n

-

Step 1 disks from A to B using c as auxiliary.

Step 3: Move the remaining disk from A to C.

Step 4: Move the n-1 disks from B to C using as auxiliary.

THEORY

The **Tower of Hanoi** is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

The program objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smallerdisk.

With three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks

PROGRAM:

```
#include<stdio.h>
Void tower(int n,char frompeg,char  topeg,char  auxpeg);  int
n;
void main()
{
 printf("Enter the no. of discs: \n");
 scanf("%d",&n);
 printf("the number of moves in tower of henoi problem\n");
 tower(n,'A','C','B');
}
void  tower(int  n,char  frompeg,char  topeg,char auxpeg)
{
 if(n==1)
  {
   printf("move disk1 from %C to %C\n ",frompeg,topeg);
   return;
  }
 tower(n-1,frompeg,auxpeg,topeg);
 printf("move disk%d from %C to %C\n",n,frompeg,topeg);
 tower(n-1,auxpeg,topeg,frompeg);
}
```

Output

```
Enter the no. of discs:
3
the number of moves in  tower of henoi problem
Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C
```

Program outcome:

- Identify the application of Tower of Hanoi problem.
- Implement the methodology to solve Tower of Hanoi problem.
- Implement the given problem using recursive function.

Viva Questions

- How do you solve the problem of the Tower of Hanoi usingrecursion?
- What is recursion? And what is tower of Hanoi problem?

PROGRAM 6

Design, develop and implement a menu driven Program in C for thefollowing operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate *Overflow* and *Underflow* situations on Circular QUEUE

d.Display the status of Circular QUEUE

e. Exit

Support the program with appropriate functions for each of the aboveoperations

Program objective:

- Understand the working of circularqueue
- Know the advantages of circular queue over liner queue.
- Understand the insertion and deletion operation oncircular queue.
- Understand overflow and underflow conditions incircularqueue.

ALGORITHM:

Step1: Initialize front and rear pointer and also count
front->0,count<-0,rear<- -1

Step2: Insert an element into queue before check overflow condition
Count=max
Insert an element rear<-(rear+1) %max
q[rear]<-item and count=count+1

Step3: Delete an element from queue .check underflow condition
Count=0 underflow condition. Count<-count-1
Item<-q[front]Deleted element

Step4: Display contents of queue. Number of elements represents count.
Check empty queue condition before displaying an element

THEORY

Circular queue is a linear data structure. It follows FIFO principle. In **circular queue** the last node is connected back to the first node to make a **circle**.

It is also called FIFO structure. Elements are added at the rear end and the elements are deleted at front end of the **queue**. The queue is considered as a circular queue when the positions 0 and MAX-1 are adjacent.

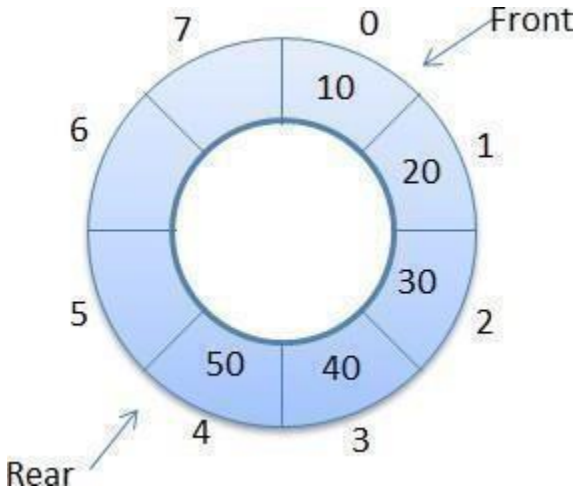

Fig6-circular queue

The **limitation** of **simple queue** is that even if there is a free memory space available in the simple queue we cannot use that free memory space to insert element. **Circular Queue is designed to overcome the limitation of Simple Queue.**

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
   Char q[MAX],item;
int f=0,r=-1,count=0;
void insert();
void  delete();
void  display();
main()
{
int ch;
while(1)
{
printf("1.insert 2.delete 3.display 4.exit \n");
printf("enter choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:getchar();insert();
      break;
case 2:delete();
      break;
case 3:display();
      break;
case  4:exit(0);
default :printf("Invalid choice\n");
        break;
```

```
}
}
}
void insert()
{
if(count==MAX)
printf("queue overflow\n");
else
{
printf("enter the item to be inserted\n");
scanf("%c",&item );
r=(r+1)%MAX;
q[r]=item;
count++;
}
}
void delete()
{
if(count==0)
printf("queue underflow\n");
else
{
printf("deleted item is %c\n",q[f]);
f=(f+1)%MAX;
count--;
}
}
void display()
{
int j=f,i;
```

```
if(count==0)
printf("queue is empty\n");
else
{
printf("contents of circular queue\n");
for(i=1;i<=count;i++)
{
printf("%c\t",q[j]);
j=(j+1)%MAX;
}
printf("total number of items=%d\n",count);
}
}
```

Output

```
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the item to be inserted: A
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the item to be inserted: B
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the item to be inserted: C
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the item to be inserted: D
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 3
Contents of Queue is:
A B C D
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the character / item to be inserted: F
Queue is Full
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 2
Deleted item is: A
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 2
Deleted item is: B
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 3
Contents of Queue is:
C D
1. Insert 2. Delete 3. Display 4. Exit
Enter the choice: 1
Enter the item to be inserted: K
1. Insert 2. Delete
Enter the choice: 3
```

Contents of Queue is:
C D K
1. Insert 2. Delete 3. Display4.Exit
Enter the choice: 4

Program outcome:

- Identify the applications of circular queue.
- Implement insert and delete operations on circular queue.

Viva Questions:

- What is a queue ?what are applications of queue?
- What is Circular Queue? What is the difference between a Stack and a Queue?

PROGRAM 7

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Branch, Sem, PhNo*

a **Create a SLL of N Students Data by using *front insertion*.**
b **Display the status of SLL and count the number of nodes in it**
c **Perform Insertion and Deletion at End of SLL**
d **Perform Insertion and Deletion at Front of SLL**
e **Demonstrate how this SLL can be used as STACKandQUEUE**
f **Exit**

Program objective:

- Understand the Singly Linked List (SLL) data structures.
- Understand the methodology to insert and delete the element at the frontof SLL.
- Understand the methodology to insert and delete the element at theendof SLL.
- Get the knowledge of how SLL and be used as both stack and queue.

Algorithm

Step 1: declare structure of node createemptylist
head->null
Step2: **Insert at front end**
head<-null
returntemp
if list is empty
temp->link=head
return head
Step 3**:Insert at rear end**
head=null
returntemp
if list is empty
cur->head
while(cur!=null)
cur=cur->link
cur->link=temp;
return head

Step 4**: Delete at front end**

head->link=null;

return null

if list has only one node

cur=head

head=head->link

free(cur)

Step 5:**Delete at Rear end**

head->link=null

return null

if only one node

cur<-head

while(cur!=null)

prev<-cur, cur=cur<-link;

free(cur);

THEORY

Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts. Each node consists of its own data and the address of the next node and forms a chain. Linked Lists are used to create trees and graphs.

In any single linked list, the individual element is called as "Node". Every "Node" contains two fields, data and next. The data field is used to store actual value of that node and next field is used to store the address of the next node in the sequence.

The graphical representation of a node in a single linked list is as follows...

Fig-7 Graphical Representation of Linked List

In a single linked list, the address of the first node is always stored in a reference node known as "front" (Some times it is also known as "head").Always next part (reference part) of the last node must be NULL.

They are a dynamic in nature which allocates the memory when required.

- Insertion and deletion operations can be easilyimplemented.
- Stacks and queues can be easily executed.
- Linked List reduces the access time.
- Linked lists are used to implement stacks, queues, graphs, etc.
- Linked lists let you insert elements at the beginning and end of the list.
- In Linked Lists we don't need to know the size in advance.

Advantages over arrays

1) Dynamic size
2) Ease of insertion/deletion

Drawbacks:

1) Random access is not allowed. We have to access elements sequentially starting fromthe first node. So we cannot do binary search with linked lists.
2) Extra memory space for a pointer is required with each element of the list.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void create();
void insert_front();
void insert_rear();
void  display();
void delete_front();
void delete_rear();
int count=0;
struct  node{
char usn[20],name[50],branch[10];
int sem;
unsigned long long int phno;
struct node *link;
};
struct node *first=NULL,*last=NULL,*temp=NULL,*p;
void main()
{
int ch,n,i;
while(1)
{
printf("1.create SLL 2.insert at front 3.insert at rear 4.display 5.delete at front 6.delete at rear 7.exit\n");
printf("enter choice\n");
scanf("%d",&ch);
switch(ch)
{
```

```
case 1:printf("enter the no.of students\n");
scanf("%d",&n);
 for(i=1;i<=n;i++)
     insert_front();
     break;
case 2:insert_front();
break;
case 3:insert_rear();break;
case 4:display();break;
case 5:delete_front();break;
case 6:delete_rear();break;
case 7:exit(0);
default:printf("invalid choice\n");break;
}
}
}
void create()
{
char usn[20],name[50],branch[10];
int sem;
unsigned long long int phno;
temp=(struct node*)malloc(sizeof(struct node));
printf("enter usn,name,branch,sem,phno\n");
scanf("%s%s%s%d%llu",usn,name,branch,&sem,&phno);
strcpy(temp->usn,usn);
strcpy(temp->name,name);
strcpy(temp->branch,branch);
temp->sem=sem;
temp->phno=phno;
```

```
count++;
}
void insert_front()
{
if(first==NULL)
{
create();
temp->link=NULL;
first=temp;
last=temp;
}
else
{
create();
temp->link=first;
first=temp;
}
}
void insert_rear()
{
if(first==NULL)
{
create();
temp->link=NULL;
first=temp;
last=temp;
}
else
```

```
{


create();
temp->link=NULL;
last->link=temp;
last=temp;
}
}
void display()
{
if(first==NULL)
{
printf("list is empty\n");
}
else
{
p=first;
printf("content of list is\n");
while(p!=NULL)
{
printf("%s\t%s\t%s\t%d\t%llu\n",p->usn,p->name,p->branch,p->sem,p->phno);
p=p->link;
}
printf("total no.of students %d\n",count);
}
}
void delete_front()
{
p=first;
```

```
if(first==NULL)
{


printf("list is empty\n");
}
else if(p->link==NULL)
{
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p->name,p->branch,p->sem,p->phno);
free(p);
first=NULL;
count--;
}
else
{
first=p->link;
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p->name,p->branch,p->sem,p->phno);
free(p);
count--;
}
}
void delete_rear()
{
p=first;
if(first==NULL)
{
printf("list is empty\n");
}
else if(p->link==NULL)
{
```

```
printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",p->usn,p->name,p->branch,p->sem,p->phno);
free(p);
first=NULL;
count--;
}
else
{
while(p->link!=last)
p=p->link;
    printf("deleted node is %s\t%s\t%s\t%d\t%llu\n",last->usn,last->name,last->branch,last->sem,last->phno);
free(last);
p->link=NULL;
last=p;
count--;
}
}
```

Output

```
_____MENU-_____

1  create a SLL of n emp
2  - Display from beginning
3 - Insert at end
4 - delete at end
5 -  Insert at beg
6 - delete at beg
7 - exit
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Enter choice :

2
List empty to display


_____MENU-_____

1  create a SLL of n emp
2  - Display frombeginning
3 - Insert at end
4 - delete at end
5 -  Insert at beg
6 - delete at beg
7 - exit
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Enter choice :1

Enter no of students :
2
```

Enter usn,name, branch, sem, phno of student :
4ad16cs022 harsha cs 3 9912367789

Enter usn,name, branch, sem, phno of student :
4ad16cs024 deepak cs 3 9538218822

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
2

Linked list elements from begining :
4ad16cs024 deepak cs 3 9538218822
4ad16cs022 harsha cs 3 9912367789

No of students = 2

MENU-

1 create a SLL of n emp
2 - Displayfrombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter usn,name, branch, sem, phno of student :
4ad16cs011 bharath cs 3 9912698467

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
2

Linked list elements from begining :

4ad16cs024	deepak	cs	3	9538218822
4ad16cs022	harsha	cs	3	9912367789
4ad16cs011	bharath	cs	3	9912698467

No of students = 3

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

– – – – – – – – – – – – – – – – – – –

Enter choice :
5

Enter usn,name, branch, sem, phno of student :
4ad16cs033 jayakumar cs 3 8903478345

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
2

Linked list elements from begining :

4ad16cs033	jayakumar	cs	3	8903478345
4ad16cs024	deepak	cs	3	9538218822
4ad16cs022	harsha	cs	3	9912367789
4ad16cs011	bharath	cs	3	9912698467

No of students = 4

MENU-

1 create a SLL of n emp
2 - Displayfrombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
4
4ad16cs011 bharath cs 3 9912698467

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
2

Linked list elements from begining :

4ad16cs033	jayakumar	cs	3	8903478345
4ad16cs024	deepak	cs	3	9538218822
4ad16cs022	harsha	cs	3	9912367789

No of students = 3

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

– – – – – – – – – – – – – – – – – – –

Enter choice :
6
4ad16cs033 jayakumar cs 3 8903478345

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
2

Linked list elements from begining :
4ad16cs024 deepak cs 3 9538218822
4ad16cs022 harsha cs 3 9912367789
No of students = 2

MENU-

1 create a SLL of n emp
2 - Displayfrombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
8
wrong choice

MENU-

1 create a SLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
– – – – – – – – – – – – – – – – – – –

Enter choice :
7

Program outcome :

- Implement Singly Linked List.
- Implement insertion at the front and end of SLL.
- Implement deletion at the front and end of SLL.
- Identify the applications of SLL.
- Familiarized how SLL can be used as both stack and queue.

Viva Questions :

- What is a Linked List and What are its types? What is a node?
- What are the parts of a linked list? What are the advantages of linked list?
- Mention what is traversal in linked lists?

PROGRAM 8

Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept, Designation,Sal, PhNo*

a. Create a **DLL** of **N** Employees Data by using ***end insertion***.
b. Display the status of **DLL** and count the number of nodes in it
c. Perform Insertion and Deletion at End of **DLL**
d. Perform Insertion and Deletion at Front of **DLL**
e. Demonstrate how this **DLL** can be used as **Double Ended Queue**
f. Exit

Program objective:

- Understand the Doubly Linked List (DLL) data structures.
- Understand the methodology to insert and delete the element at the frontof DLL.
- Understand the methodology to insert and delete the element at the endof DLL.
- Get the knowledge of how DLL and be used as double ended queue.

Algorithm

Insertion at front end of list.

Step 1: Allocate memory for temp node and assign values to node
Step2: if list is empty, temp is attached to list directly
head=null
returntemp
if list is not empty
temp->rlink=head
head->llink=temp
return head

Insertion at rear end of list.

Step1: Read node information and allocate memory for temp node
Step2: traverse the cur node upto to end of list then attach node cur to temp
cur->rlink=temp;
temp->llink=cur
Step 3:return starting address of list

return head;

Delete from front end of list.

Step 1: check if list has only one node
head=NULL;
return null;
if list is empty
head->rlink=NULL
return NULL;
if list has o0nly one node
Step 2:otherwise first node address is shifted to next node
cur=head
head=head->rlink
free(cur)
Step 3: return starting address of list
return head

Delete node from rear end

Step 1: two pointers requires one is cur and prev
Cur is one which points, node to be deleted.

Step 2: Traverse the cur node upto end of list before updating current pointer save the
Address to prev pointer.

```
While(cur->rlink!=null)
{
prev=cur;
cur=cur->rlink;
}
prev->rlink=null;
cur->llink=null;
free(cur);
```

Step3: return starting address of the list

THEORY

- In computer science, a doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes.

- Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes' previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.

- A doubly linked list whose nodes contain three fields: an integer value, the link to the next node, and the link to the previous node.

- The two node links allow traversal of the list in either direction. While adding or removing a node in a doubly linked list requires changing more links than the same operations on a singly linked list, the operations are simpler and potentially more efficient (for nodes other than first nodes) because there is no need to keep track of the previous node during traversal or no need to traverse the list to find the previous node, so that its link can be modified.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void create();
void insert_front();
void insert_rear();
void  display();
void delete_front();
void delete_rear();
int count=0;
struct  node
{
     int ssn;
     char name[50],dept[20],desg[20];
     float sal;
     unsigned long long int phno;
     struct node *llink,*rlink;
};
struct node *first=NULL,*last=NULL,*temp;
main()
{
int ch,n,i;
while(1)
{
printf("1.create\n 2.insert_front\n 3.insert_rear\n 4.display\n 5.delete_front\n 6.delete_rear\n
7.exit\n");
printf("enter  choice\n");
```

```
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter the number of employee\n");
       scanf("%d",&n);
       for(i=0;i<n;i++)
       insert_rear();
       break;
case 2:insert_front();break;
case 3:insert_rear();break;
case  4:display();break;
case 5:delete_front();break;
case 6:delete_rear();break;
case 7:exit(0);
default:printf("invalid   choice\n");break;
}
}
}
void create()
{
       int ssn;
       char name[50],dept[20],desg[20];
       float sal;
 unsigned long long int phno;
       temp=(struct node*)malloc(sizeof(struct node));
       temp->llink=temp->rlink=NULL;
printf("enter ssn,name,dept,desg,salaryand phno\n");
       scanf("%d%s%s%s%f%llu",&ssn,name,dept,desg,&sal,&phno);
       temp->ssn=ssn;
```

```
    strcpy(temp->name,name);
   strcpy(temp->dept,dept);
   strcpy(temp->desg,desg);
   temp->sal=sal;
    temp->phno=phno;
    count++;
}
void insert_front()
{
    if(first==NULL)
    {
            create();
            first=temp;
            last=temp;
    }
    else
    {
            create();
   temp->rlink=first;
            first->llink=temp;
            first=temp;
    }
}
void insert_rear()
    {
     if(first==NULL)
            {
            create();
            first=temp;
```

```
            last=temp;
            }
    else
    {

            create();
            last->rlink=temp;
            temp->llink=last;
            temp->rlink=NULL;
            last=temp;
    }
}
void display()
{
struct node *p;
if(first==NULL)
{
printf("list is empty\n");
return;
}
p=first;
printf("contents of list\n");
while(p!=NULL)
{
printf("%d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p->name,p->dept,p->desg,p->sal,p->phno);
p=p->rlink;
}
printf("total no. of employee %d\n",count);
}
```

```
void delete_front()
{
struct node *p;
if(first==NULL)
{
printf("list is empty,cannot delete\n");
}
else if(first->rlink==NULL)
{
printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",first->ssn,first->name,first->dept,first->desg,first->sal,first->phno);
first=NULL;
free(first);
count--;
}
else
{
p=first;
first=p->rlink;
printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p->name,p->dept,p->desg,p->sal,p->phno);
free(p);
count--;
}
}
void delete_rear()
{
  struct node*p;
  if(first==NULL)
            {
```

```
            printf("list is empty,cannot delete\n");
            }
    else if(first->rlink==NULL)
            {
            printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",first->ssn,first->name,first->dept,first->desg,first->sal,first->phno);
            first=NULL;
            free(first);
            count--;
            }
    else
    {
    p=last;
    last=p->llink;
printf("deleted data is %d\t%s\t%s\t%s\t%f\t%llu\n",p->ssn,p->name,p->dept,p->desg,p->sal,p->phno);
    free(p);
    last->rlink=NULL;
    count--;
    }
}
```

Output

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

-

Enter choice :
2
List empty to display

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

-

Enter choice :
1

Enter no of employees :
2
Enter ssn,name,department, designation, salary and phno ofemployee :
120 harsha cs instructor 14000 9912378956

Enter ssn,name,department, designation, salary and phno ofemployee :
121 sanjay cs programmer 15000 9538215567

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

Enter choice :
2

Linked list elements from begining :
120 harsha cs instructor 14000.000000 9912378956
121 sanjay cs programmer 15000.000000 9538215567
No of employees = 2

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at
end 5 - Insert at
beg 6 - delete
at beg 7 - exit

Enter choice :
3

Enter ssn,name,department, designation, salary and phno ofemployee :
123 deepak cs instructor 14000 9534567812

_____MENU-_____

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
_____-

Enter choice :
2

Linked list elements from begining :
120 harsha cs instructor 14000.000000 9912378956
121 sanjay cs programmer 15000.000000 9538215567
123 deepak cs instructor 14000.000000 9534567812
No of employees = 3
_____MENU-_____

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
_____-

Enter choice :
5

```
Enter ssn,name,department, designation, salary and phno ofemployee :
124 lohith cs lecturer 20000 9967834578

_____MENU-_____

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
_____-

Enter choice :
2

Linked list elements from begining :
124 lohith cs lecturer 20000.000000 9967834578
120 harsha cs instructor 14000.000000 9912378956
121 sanjay cs programmer 15000.000000 9538215567
123 deepak cs instructor 14000.000000 9534567812
 No of employees = 4

_____MENU-_____

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
_____-
Enter choice :
4
123 deepak cs instructor 14000.000000 9534567812
```

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
-

Enter choice :
2

Linked list elements from begining :
124 lohith cs lecturer 20000.000000 9967834578
120 harsha cs instructor 14000.000000 9912378956
121 sanjay cs programmer 15000.000000 9538215567
No of employees = 3

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit
-

Enter choice :
6

124lohith cs lecturer 20000.000000 9967834578

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

-

Enter choice :
2

Linked list elements from begining :
120 harsha cs instructor 14000.000000 9912378956
121 sanjay cs programmer 15000.000000 9538215567
No of employees = 2

MENU-

1- create a DLL of n emp
2 - Display frombeginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

-

Enter choice :
8
wrong choice

MENU-

1- create a DLL of n emp

2 - Display from beginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

Enter choice :
7
$

Program outcome:

- Implement Doubly Linked List.
- Implement insertion at the front and end of DLL.
- Implement deletion at the front and end of DLL.
- Identify the applications of DLL.
- Familiarized how DLL can be used as double ended queue.

Viva Questions:

- What are doubly linked lists?
- What is the difference between singly and doubly linked lists?
- What are the advantages of double linked list over single linked list?