LAB PROGRAM 1

Design, Develop and Implement a menu driven Program in C for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements with Suitable Headings

c. Inserting an Element (**ELEM**) at a given valid Position (**POS**) `

d. Deleting an Element at a given valid Position (**POS**)

e. Exit.

Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
int a[MAX];
int *src,*des; //used for left shift and right shift
int n,i;  // n- gives the count of elements in array

//Function to create an array and insert elements into the array
void Create_Array()
{
printf("enter the number of array elements to be created with in MAX= %d size \n",MAX);
scanf("%d",&n);
for(i=0;i<n;i++)
        {
                printf("enter the %d element into the array\n",i+1);
                scanf("%d",&a[i]);
        }
}//end of function Create_Array

//Function to display array elements
void Display_Array()
{
if(n==0)
{
        printf("NO ELEMENTS TO DISPLAY \n");
        return;
}
printf("Elements of the array are: \n");
for(i=0;i<n;i++)
        {
                printf("%d\n",a[i]);
        }
```

```
}//end of function Display_Array

//Function to insert an element at a valid position
void Insert_Array()
{
int ELEM,pos;
if(n==MAX)
{
printf("ARRAY FULL INSERTION NOT POSSIBLE\n");
return;
}
else
{
printf("Enter a valid position for insertion \n");
scanf("%d",&pos);
if(pos>n+1)
{
printf("**Invalid Postion** give a value within %d \n",n+1);
return;
}
else
{
        printf("Enter the element to be inserted into the array\n");
        scanf("%d",&ELEM);
        des=&a[n];
        src=&a[n-1];
        for(i=n+1;i>pos;i--)
        {

                *des=*src;
                src--;
                des--;
        }
        *des=ELEM;
        n++;
}
printf("INSERTION SUCCESSFUL\n");
}
}//end of function Insert_Array

//Function to delete an element from a valid position
void Delete_Array()
{
int pos;
if(n==0)
{
```

```
printf("ARRAY EMPTY DELETION NOT POSSIBLE\n");
return;
}
else
{
printf("Enter the position of deletion\n");
scanf("%d",&pos);
if(pos>n)
{
printf("INVALID POSITION\n");
return;
}
else
{
        src=&a[pos];
        des=&a[pos-1];
        for(i=0;i<=n-pos;i++)
        {

                *des=*src;
                des++;
                src++;
        }
        n--;

}
printf("DELETION SUCCESSFUL\n");
}
}//end of function Delete_Array

void main()
{
        int ch;
        while(1)
        {
                printf("*************** MENU ***************\n");
                printf("1. CREATE-ONLY DONE INITIALLY\n");
                printf("2. DISPALY\n");
                printf("3. INSERT\n");
                printf("4. DELETE \n");
                printf("5. EXIT\n");
                printf("Enter your choice \n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: Create_Array();
```

```
                        break;
                        case 2: Display_Array();
                        break;
                        case 3: Insert_Array();
                        break;
                        case 4: Delete_Array();
                        break;
                        case 5: exit(0);
                        break;
                        default:printf("Enter a valid choice\n");
                }
        }

}
```

OUTPUT:

```
********** MENU **********

1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
1
enter the number of array elements to be created with in MAX= 5 size
3
enter the 1 element into the array
1
enter the 2 element into the array
2
enter the 3 element into the array
3
********** MENU **********
1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
2
Elements of the array are:
Element 1
Element 2
Element 3
```

********** MENU **********
1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
3
Enter the element to be inserted into the array
4
Enter a valid position for insertion
1
INSERTION SUCCESSFUL
********** MENU **********
1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
2
Elements of the array are:
Element 4
Element 1
Element 2
Element 3
********** MENU **********
1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
4
Enter the position of deletion
4
DELETION SUCCESSFUL
********** MENU **********
1. CREATE-ONLY DONE INITIALLY
2. DISPALY
3. INSERT
4. DELETE
5. EXIT
Enter your choice
2
Elements of the array are:

Element 4
Element 1
Element 2

LABPROGRAM 2

Design, Develop and Implement a Program in C for the following operations on **Strings**

a. Read a main String (**STR),** a Pattern String (**PAT**) and a Replace String (**REP**)

b. Perform Pattern Matching Operation: Find and Replace all occurrences of **PAT** in **STR** with **REP** if **PAT** exists in **STR.** Report suitable messages in case **PAT** does not exist in **STR**

Support the program with functions for each of the above operations. Don't use Built-in functions.

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

char STR[100],PAT[100],REP[100];   //string array

//Function to find the length of a string
int length(char t[])
{
        char *p;
        int c=0;
        for(p=t;*p!='\0';p++)
        {
                c++;
        }
        return c;
}


//Function to compare two given string of same length
int strcompare(char pat[],char temp[])
{
        int i,flag=0;
        for(i=0;i<length(PAT);i++)
        {
                if(pat[i]==temp[i])
                flag++;

        }
        if(flag==length(pat))
        return 0;
        else
        return 1;
}

void Find_Replace()
```

```
{
        char temp[100];
        int n=length(STR);
        int m=length(PAT);
        int i,j,s,k,start_pos,count=0;
        printf("\nEnter the replace string REP:\n");
        printf("\n\t(REP string must be of same length as the PAT string %d:)\n" ,length(PAT));
        gets(REP);
        for(s=0;s<=n;s++)
        {
                for(j=s,k=0;j<m;j++,k++)
                        temp[k]=STR[s+k];
                temp[k]='\0';
                if(strcompare(PAT,temp)==0)
                {
                        printf("\n Pattern Occurs With Shift : %d \n",s);
                        start_pos = s;
                        for(i=start_pos,j=0;j<length(REP);i++,j++)
                        {
                                STR[i]=REP[j];
                        }
                        count++;
                }
                m++;
        }
        if(count== 0)
        printf("\n THE PATTERN DOES NOT OCCUR ** CANNOT REPLACE**\n");
        else
        {
        printf("\n THE FINAL STRING IS: ");
        puts(STR);
        }
}


void main()
{
        printf("\n ENTER THE TEXT    : ");
        gets(STR);
        printf("\n ENTER THE PATTERN : ");
        gets(PAT);
        Find_Replace();
}
```

OUTPUT:

ENTER THE TEXT : hello how are you

ENTER THE PATTERN : how

Enter the replace string REP:

(REP string must be of same length as the PAT string 3:)
now

Pattern Occurs With Shift : 6

THE FINAL STRING IS : hello now are you

LAB PROGRAM 3

Design, Develop and Implement a menu driven Program in C for the following operations on **STACK** of Integers (Array Implementation of Stack with maximum size **MAX**)

a. ***Push*** an Element on to Stack

b. ***Pop*** an Element from Stack

c. Demonstrate how Stack can be used to check ***Palindrome***

d. Demonstrate ***Overflow*** and ***Underflow*** situations on Stack

e. Display the status of Stack

f. Exit

Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAX 5

// Function to push elements into the stack
void push(int *s,int ELEM,int *top)
{
        if(*top==MAX-1)
        printf("STACK OVERFLOW \n");
        else
        s[++(*top)]=ELEM;
}

//Function to pop elements from stack
int pop(int *s,int *top)
{
        int ELEM;
        if(*top==-1)
        {
        printf("STACK UNDERFLOW\n");
        return -1;
        }
        else
        {
        ELEM = s[(*top)--];
        return ELEM;
        }
}
```

```
//Function to display elements of the stack
void display(int *s,int *top)
{
	int i;
	if(*top==-1)
	printf("NOTHING TO DIAPLAY \n");
	else
	{
	printf("The elements in stack are:\n");
	for(i=*top;i>=0;i--)
		printf("%d\n",s[i]);
	}
}


//Function to check for palindrome
void check_palindrome(int *s,int *top)
{
	int temp[MAX],rev[MAX];
	int flag=0,i,n=*top;
	for(i=0;i<=n;i++) //copy the contents of the
		temp[i]=s[i];    //stack to another temperory stack
	for(i=0;i<=*top;i++) //pop the elements from teperory stack
	{			//store it an array called reverse
		if(s[i]==(rev[i]=pop(temp,&n)))
		flag=1 ;
		else
		{
		flag=0;
		break;
		}
	}
	printf("The number ** : ");
	for(i=0;i<=*top;i++)
	{
		printf("%d\t",s[i]);
	}
	printf("is :");
	if(flag==1)
	printf("PALNDROME \n");
	else
	printf("NOT PALINDROME \n");
}
```

```
//Main function
void main()
{
	int ch;
	int ELEM;
	int
	top=-1;
	int s[MAX];
	while(1)
	{
		printf("********** STACK IMPLEMENTAITON MENU ********** \n");
		printf(" 1. PUSH \n");
		printf(" 2. POP \n");
		printf(" 3. CHECK PALINDROME \n");
		printf(" 4. Display \n");
		printf(" 5. EXIT \n");
		printf("enter the chioce: \n");
		scanf("%d",&ch);
		switch(ch)
		{
		case 1: printf("enter the element to be inserted: Let ELEM be single digit + ve
integer \n");
			scanf("%d",&ELEM);
			push(s,ELEM,&top);
			break;
		case 2: ELEM=pop(s,&top);
			if(ELEM==-1)
			printf("NOTHING TO DELETE\n");
			else
			printf("The deleted elemnt is :  %d\n",ELEM);
			break;
		case 3: check_palindrome(s,&top);
			break;
		case 4: display(s,&top);
			break;
		case 5: exit(0);
			break;
		default: printf("wrong choice\n");
			break;

		}
	}
}
```

OUTPUT:

********** STACK IMPLEMENTAITON MENU **********

1. PUSH
2. POP
3. CHECK PALINDROME
4. Display
5. EXIT

enter the chioce:

1

enter the element to be inserted:Let ELEM be single digit integer

1

********** STACK IMPLEMENTAITON MENU **********

1. PUSH
2. POP
3. CHECK PALINDROME
4. Display
5. EXIT

enter the chioce:

1

enter the element to be inserted:Let ELEM be single digit integer

2

********** STACK IMPLEMENTAITON MENU **********

1. PUSH
2. POP
3. CHECK PALINDROME
4. Display
5. EXIT

enter the chioce:

1

enter the element to be inserted:Let ELEM be single digit integer

3

********** STACK IMPLEMENTAITON MENU **********

1. PUSH
2. POP
3. CHECK PALINDROME
4. Display
5. EXIT

enter the chioce:

4

The elements in stack are:

3

2

1

********** STACK IMPLEMENTAITON MENU **********

1. PUSH
2. POP

```
 3. CHECK PALINDROME
 4. Display
 5. EXIT
enter the chioce:
3
The number ** : 1       2       3       is :NOT PALINDROME
********** STACK IMPLEMENTAITON MENU **********
 1. PUSH
 2. POP
 3. CHECK PALINDROME
 4. Display
 5. EXIT
enter the chioce:
2
The deleted elemnt is :  3
********** STACK IMPLEMENTAITON MENU **********
 1. PUSH
 2. POP
 3. CHECK PALINDROME
 4. Display
 5. EXIT
enter the chioce:
1
enter the element to be inserted:Let ELEM be single digit integer
1
********** STACK IMPLEMENTAITON MENU **********
 1. PUSH
 2. POP
 3. CHECK PALINDROME
 4. Display
 5. EXIT
enter the chioce:
3
The number ** : 1       2       1       is :PALNDROME
```

LAB PROGRAM 4

Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: **+, -, *, /, %(Remainder), ^(Power)** and **alphanumeric** operands**.**

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define stack_size 10

//Function to push elemets on to the stack
void push(char item,int *top, char s[])
{
        if(*top==stack_size-1)
          {
        printf("stack overflow\n");
        return;
          }
        s[++(*top)]=item;
}

//Function to pop the elements from the stack
char pop(int *top,char s[])
{
char item_d;
        if(*top==-1)
        {
        printf("Stack underflow\n");
        return 0;
        }

        item_d=s[(*top)--];
        return item_d;
}

//Functions to consider the operator precedence and priority
int g(char symbol)
{
switch(symbol)//infix array/expression
   {
        case '+':
        case '-':return 1;
        case '*':
```

```
        case '/':
        case '%':return 3;
        case '^':return 6;
        case '(':return 9;
        case ')':return 0;
        default :return 7;
    }
}
int f(char symbol)//stack
{
switch(symbol)
     {
        case '+':
        case '-':return 2;
        case '*':
        case '/':
        case '%':return 4;
        case '^':return 5;
        case '(':return 0;
        case '#':return -1;
        default :return 8;
        }
}


//Function to convert infix expression to postfix
void infix_postfix(char infix[],char postfix[])
{
        int top=-1;
        char symbol;
        char s[10];
        char item;
        int i;
        int j=0;
        push('#',&top,s);
        for(i=0;i<strlen(infix);i++)
        {
                symbol=infix[i];
                while(f(s[top])>g(symbol))
                {
                        postfix[j++]=pop(&top,s);
                }
                if(f(s[top])!=g(symbol))
                {
                        push(symbol,&top,s);
                }
```

```
                else
                pop(&top,s);
        }
        while(s[top]!='#')
        {
                postfix[j++]=pop(&top,s);
        }
        postfix[j]='\0';
}

//Main Function
void main()
{
        char infix[10];
        char postfix[10];
        printf("************************************************\n");
        printf("CONVERSION FROM INFIX TO POSTFIX\n");
        printf("************************************************\n");
        printf("Enter the valid infix expression\n");
        scanf("%s",infix);
        infix_postfix(infix,postfix);
        printf("The equivalent postfix expression\n");
        printf("%s",postfix);
}
```

OUTPUT:

```
**********************************************

CONVERSION FROM INFIX TO POSTFIX

**********************************************

Enter the valid infix expression

3-1+6%3^7

The equivalent postfix expression

31-637^%+

**********************************************

CONVERSION FROM INFIX TO POSTFIX

**********************************************

Enter the valid infix expression
```

3-(1+6)%3^7

The equivalent postfix expression

316+37^%-

LAB PROGRAM 5

Design, Develop and Implement a Program in C for the following Stack Applications:

a) Evaluation of **Suffix expression** with single digit operands and operators: **+, -, *, /, %,** ^

b) Solving **Tower of Hanoi** problem with **n** disks

LAB PROGRAM 5A

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
double op(char symbol,double op1,double op2)
{
        switch(symbol)
         {
            case '+':return op1+op2;
            case '-':return op1-op2;
            case '*':return op1*op2;
            case '/':return op1/op2;
            case '%':return op1%op2;
            case '^': return pow(op1,op2);
         }
        return 0;
}
void push(double item,int *top,double s[])
  {
     *top=*top+1;
     s[*top]=item;
  }

double pop(int *top,double s[])
{
    double item;
    item=s[(*top)];
    *top=*top-1;
    return item;
}
 int isdigit(char symbol)
 {
    return(symbol>='0'&& symbol<='9');
```

```
}
void main()
{
     double op1,op2,res;
     double s[10];
     char postfix[10];
     int top=-1;
     char symbol;
     int i;
printf("Enter the valid postfix expression\n");
scanf("%s",postfix);
for(i=0;i<strlen(postfix);i++)
{
          symbol=postfix[i];
          if(isdigit(symbol))
  {
         push(symbol-'0',&top,s);
  }
else
   {
    op2=pop(&top,s);
    op1=pop(&top,s);
    res=op(symbol,op1,op2);
    push(res,&top,s);
    }
}
   res=pop(&top,s);
   printf("The result is=%f\n",res);
}
```

OUTPUT:

Enter the valid postfix expression

3-1+6%3^7

The result is=7.000000

LAB PROGRAM 5B

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

void towers(int, char, char, char);
void main()
{
        int num;
        printf("Enter the number of disks : ");
        scanf("%d", &num);
        printf("The sequence of moves involved in the Tower of Hanoi are :\n");
        towers(num, 'S', 'T', 'D');
}
void towers(int num, char S, char T, char D)
{
        if (num == 1)
        {
                printf("\n Move disk 1 from peg %c to peg %c", S, D);
                return;
        }
        towers(num-1, S, D, T);
        printf("\n Move disk %d from peg %c to peg %c", num, S, D);
        towers(num-1, T, S, D);
}
```

OUTPUT:

Enter the number of disks:

3

The sequence of moves involved in the Tower of Hanoi are:

Move disk 1 from peg A to peg C

Move disk 2 from peg A to peg B

Move disk 1 from peg C to peg B

Move disk 3 from peg A to peg C

Move disk 1 from peg B to peg A

Move disk 2 from peg B to peg C

Move disk 1 from peg A to peg C

LAB PROGRAM 6

Design, Develop and Implement a menu driven Program in C for the following 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 above operations

```
#include<stdio.h>
#include<stdlib.h>
#define CQSIZE 5
int rear=-1,front=-1;
char item;
char cq[CQSIZE];
void main()
{
  int ch;
  while(1)
  {  printf("*****************************************\n");
     printf("CIRCULAR QUEUE OF CHARACTER IMPLENTATION\n");
     printf("*****************************************\n");
     printf("\n1.CQINSERT\n");
     printf("2.CQDELETE\n");
     printf("3.CQDISPLAY\n");
     printf("4.QUIT\n");
     printf("\Enter your choice\n");
     scanf("%d",&ch);
     switch(ch)
          {
            case 1: cqinsert();
                    break;
            case 2: cqdelete();
                    break;
```

```
            case 3: cqdisplay();
                    break;
            case 4: exit(0);
                    break;
            default: printf("wrong choice\n");
          }
    }
}

int cqinsert()
{
  if((rear+1)%CQSIZE==front)
  {
        printf("\nCIRCULAR QUEUE overflow\n");
        return;
  }
  else
  {
     printf("enter the item to be inserted\n");
     fflush(stdin);
     item=getchar();
     if(front==-1)
         front=rear=0;
     else
         rear=(rear+1)%CQSIZE;
     cq[rear]=item;
  }
  return;
}

int cqdelete()
{
  if(front==-1) //when queue is empty
  printf("\nCIRCULAR QUEUE underflow\n");
  else
  {
     printf("the item deleted is : %c \n",cq[front]);
     if(front==rear)//if the element is the last element in Q
         front=rear=-1;
     else
         front=(front+1)%CQSIZE;
  }
  return;
}
```

```
int cqdisplay()
{
  int i;
  if(front==-1)
  printf("\nCIRCULAR QUEUE is empty\n");
  else
  {
        if(front<=rear)
        {
              printf("\nThe items in the QUEUE are\n");
              for(i=front;i<=rear;i++)
                    printf("%c\t",cq[i]);
        }
        if(front>rear)
        {
              for(i=front;i<=CQSIZE-1;i++)
                    printf("%c\t",cq[i]);
              for(i=0;i<=rear;i++)
                    printf("%c\t",cq[i]);
        }

  }
  printf("\n");
  return;
}
```

OUTPUT:

```
************************************
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
************************************

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
1
enter the item to be inserted
a
************************************
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
************************************

1.CQINSERT
```

2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
1
enter the item to be inserted
b
****
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
****

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
1
enter the item to be inserted
c
****
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
****

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
3

the items in the QUEUE are
a b c
****
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
****

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
2

```
the item deleted is a:
************************************
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
************************************

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
2
the item deleted is b:
************************************
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
************************************

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
2
the item deleted is c:
************************************
CIRCULAR QUEUE OF CHARACTER IMPLENTATION
************************************

1.CQINSERT
2.CQDELETE
3.CQDISPLAY
4.QUIT

enter your choice
2

CIRCULAR QUEUE underflow
```

LAB 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 / Deletion at End of **SLL**

d. Perform Insertion / Deletion at Front of **SLL(Demonstration of stack)**

e. Exit

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct node
{
        int sem,phno;
        char name[20],branch[10],usn[20];
        struct node *next;
}*first=NULL,*last=NULL,*temp=NULL, *temp1=NULL;

void create()
{
        int sem,phno;
        char name[20],branch[10],usn[20];
        temp=(struct node*)malloc(sizeof(struct node));
        printf("\n Enter usn,name, branch, sem, phno of student : ");
        scanf("%s %s %s %d %d", usn, name,branch, &sem,&phno);
        strcpy(temp->usn,usn);
        strcpy(temp->name,name);
        strcpy(temp->branch,branch);
        temp->sem = sem;
        temp->phno = phno;
        temp->next=NULL;
        count++;
}
void insert_atfirst()
```

```
{
	if (first == NULL)
	{
		create();
		first = temp;
		last = first;
	}
	else
	{
		create();
		temp->next = first;
		first = temp;
	}
}
void insert_atlast()
{
	if(first==NULL)
	{
		create();
		first = temp;
		last = first;
}
	else
	{
		create();
		last->next = temp;
		last = temp;
	}
}
void display()
{
	temp1=first;
	if(temp1 == NULL)
	{
		printf("List empty to display \n");
		return;
	}
	printf("\n Linked list elements from begining : \n");
	while (temp1!= NULL)
	{
		printf("\t %s %s %s %d %d\n", temp1->usn, temp1->name,temp1-
>branch,temp1->sem,temp1->phno );
		temp1 = temp1->next;
	}
	printf(" No of students = %d ", count);
}
```

```
int deleteend()
{
	struct node *temp;
	temp=first;
	if(temp->next==NULL)
	{
		free(temp);
		first=NULL;
	}
	else
	{
		while(temp->next!=last)
		temp=temp->next;
		printf("The deleted element is: ");
		printf("%s %s %s %d %d\n", last->usn, last->name,last->branch,last->sem, last->phno );
		free(last);
		temp->next=NULL;
		last=temp;
	}
	count--;
	return 0;
}
int deletefront()
{
	struct node *temp;
	temp=first;
	if(temp->next==NULL)
	{
		free(temp);
		first=NULL;
		return 0;
	}
	else
	{
		first=temp->next;
		printf("The element deleted is: ");
		printf("%s %s %s %d %d", temp->usn, temp->name,temp->branch,temp->sem, temp->phno );
		free(temp);
	}
	count--;
	return 0;
}
void main()
```

```
{
        int ch,n,i;
        first=NULL;
        temp = temp1 = NULL;
        printf("**********SLL IMPLEMENTATION**********\n");
        printf("\n 1 - Create a SLL of n student");
        printf("\n 2 - Display from beginning along with Count");
        printf("\n 3 - Insert at end-used for demonstrating stack");
        printf("\n 4 - delete at end-used for demonstrating stack");
        printf("\n 5 - Insert at beg");
        printf("\n 6 - delete at beg");
        printf("\n 7 - exit\n");
        while (1)
        {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
                case 1:
                        printf("\n Enter no of students : ");
                        scanf("%d", &n);
                        for(i=0;i<n;i++)
                        insert_atfirst();
                        break;
                case 2:
                        display();
                        break;
                case 3:
                        insert_atlast();
                        break;
                case 4:
                        deleteend();
                        break;
                case 5:
                        insert_atfirst();
                        break;
                case 6:
                        deletefront();
                        break;
                case 7:
                        exit(0);
                default: printf("wrong choice\n");
        }
        }
}
```

OUTPUT

**********SLL IMPLEMENTATION**********

1 - Create a SLL of n student
2 - Display from beginning along with Count
3 - Insert at end-used for demonstrating stack
4 - delete at end-used for demonstrating stack
5 - Insert at beg
6 - delete at beg
7 - exit

Enter choice : 1

Enter no of students : 3

Enter usn,name, branch, sem, phno of student : 1 john CSE 5 123

Enter usn,name, branch, sem, phno of student : 2 mac ISE 3 345

Enter usn,name, branch, sem, phno of student : 3 mary EEE 5 567

Enter choice : 2

Linked list elements from begining :
3 mary EEE 5 567
2 mac ISE 3 345
1 john CSE 5 123
No of students = 3
Enter choice : 5

Enter usn,name, branch, sem, phno of student : 4 zen ECE 3 666

Enter choice : 2

Linked list elements from begining :
4 zen ECE 3 666
3 mary EEE 5 567
2 mac ISE 3 345
1 john CSE 5 123
No of students = 4
Enter choice : 4
Deleted element is : 1 john CSE 5 123

Enter choice : 2

Linked list elements from begining :
4 zen ECE 3 666
3 mary EEE 5 567
2 mac ISE 3 345
No of students = 3

LAB 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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct node
{
        struct node *prev;
        int ssn,phno;
        float sal;
        char name[20],dept[10],desg[20];
        struct node *next;
}*h,*temp,*temp1,*temp2;
void create()
{
        int ssn,phno;
        float sal;
        char name[20],dept[10],desg[20];
        temp =(struct node *)malloc(sizeof(struct node));
        temp->prev = NULL;
        temp->next = NULL;
        printf("\n Enter ssn,name,department, designation, salary and phno of employee : ");
        scanf("%d %s %s %s %f %d", &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 insertbeg()
{
        if (h == NULL)
        {
                create();
                h = temp;
                temp1 = h;
        }
        else
        {
                create();
                temp->next = h;
                h->prev = temp;
                h = temp;
        }
}
void insertend()
{
        if(h==NULL)
        {
                create();
                h = temp;
                temp1 = h;
        }
        else
        {
                create();
                temp1->next = temp;
                temp->prev = temp1;
                temp1 = temp;
        }
}
void displaybeg()
{
        temp2 =h;
        if(temp2 == NULL)
        {
                printf("List empty to display \n");
                return;
```

```
	}
	printf("\n Linked list elements from begining : \n");
	while (temp2!= NULL)
	{
		printf("%d %s %s %s %f %d\n", temp2->ssn, temp2->name,temp2->dept,temp2-
>desg,temp2->sal, temp2->phno );
		temp2 = temp2->next;
	}
	printf(" No of employees = %d ", count);
}
int deleteend()
{
	struct node *temp;
	temp=h;
	if(temp->next==NULL)
	{
		free(temp);
		h=NULL;
		return 0;
	}
	else
	{
		temp2=temp1->prev;
		temp2->next=NULL;
		printf("%d %s %s %s %f %d\n", temp1->ssn, temp1->name,temp1->dept,
		temp1->desg,temp1->sal, temp1->phno );
		free(temp1);
	}
	count--;
	return 0;
}
int deletebeg()
{
	struct node *temp;
	temp=h;
	if(temp->next==NULL)
	{
		free(temp);
		h=NULL;
	}
	else
	{
		h=h->next;
		printf("%d %s %s %s %f %d", temp->ssn, temp->name,temp->dept,
		temp->desg,temp->sal, temp->phno );
		free(temp);
```

```
                h->prev=NULL;
        }
        count--;
        return 0;
}
void main()
{
        int ch,n,i;
        h=NULL;
        temp = temp1 = NULL;
        printf("************ DLL IMPLEMENTATION ************\n");
        printf("\n 1 - Create a DLL of n emp- end insertion");
        printf("\n 2 - Display from beginning with count ");
        printf("\n 3 - Insert at end - used to demonstrate DQUEUE");
        printf("\n 4 - delete at end - used to demonstrate DQUEUE");
        printf("\n 5 - Insert at beg - used to demonstrate DQUEUE");
        printf("\n 6 - Delete at beg - used to demonstrate DQUEUE");
        printf("\n 7 – Exit\n");
        while (1)
        {
                printf("\n Enter choice : ");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1:
                                printf("\n Enter no of employees : ");
                                scanf("%d", &n);
                                for(i=0;i<n;i++)
                                insertend();
                        break;
                        case 2:
                                displaybeg();
                        break;
                        case 3:
                                insertend();
                        break;
                        case 4:
                                deleteend();
                        break;
                        case 5:
                                insertbeg();
                        break;
                        case 6:
                                deletebeg();
                        break;
                        case 7:
```

```
                                exit(0);
                        default: printf("wrong choice\n");
                }
        }
}
```

OUTPUT:

********** DLL IMPLEMENTATION **********

1 - create a DLL of n emp- end insertion

2 - Display from beginning with count

3 - Insert at end - used to demonstrate DQUEUE

4 - delete at end - used to demonstrate DQUEUE

5 - Insert at beg - used to demonstrate DQUEUE

6 - delete at beg - used to demonstrate DQUEUE

7 - exit

Enter choice : 1

Enter no of employees : 2

Enter ssn,name,department, designation, salary and phno of employee :

1 tim Development SSE 10000 123

Enter ssn,name,department, designation, salary and phno of employee : 2 mark de

sign SDE 12000 234

Enter choice : 2

Linked list elements from begining :

1 tim DevelopmenSSE SSE 10000.000000 123

2 mark design SDE 12000.000000 234

No of employees = 2

Enter choice : 3

Enter ssn,name,department, designation, salary and phno of employee : 3 xel des
ign SDE 12000 567

Enter choice : 2

Linked list elements from begining :

1 tim DevelopmenSSE SSE 10000.000000 123

2 mark design SDE 12000.000000 234

3 xel design SDE 12000.000000 567

No of employees = 3

LAB PROGRAM 9

Design, Develop and Implement a Program in C for the following operationson **Singly Circular Linked List (SCLL)** with header nodes

a. Represent and Evaluate a Polynomial **P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3**

b. Find the sum of two polynomials **POLY1(x,y,z)** and **POLY2(x,y,z)** and store the result in **POLYSUM(x,y,z)**

Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct poly
{

int coef;
int expon;
struct poly *link;
};

typedef struct poly polyNode;

polyNode *a,*last_a,*b,*last_b,*c,*last_c,*node;

polyNode* getNode()
{
        /*provide a node for use */
        polyNode *node;
        node=(polyNode*)malloc(sizeof(polyNode));
        if(!node)
        {
                printf("INSUFFICIENT MEMEORY\n");
                exit(0);
        }
        return node;
}

void attach(float coefficient,int exponent, polyNode **ptr)
```

```
{
	/*create a new node with coeff=coefficient and expon=exponent, attach it to the node pointed to by ptr.ptr is updated to point to this new node. */
	polyNode *temp;
	temp=getNode();
	temp->coef=coefficient;
	temp->expon=exponent;
	(*ptr)->link=temp;
	*ptr=temp;
}

int COMPARE(int a, int b)
{
	if(a==b)
	return 0;
	else if(a>b)
	return 1;
	else
	return -1;
}


void cpadd(polyNode *a, polyNode *b)
{
 //polynomials a and b are singly linked circular lists with a header node. Return a polynomial which is a sum of a and b
	int sum;
	while(a!=last_a->link && b!= last_b->link)
	{
		switch(COMPARE(a->expon,b->expon))
		{
			case -1:
				//a->expon < b->expon
				attach(b->coef,b->expon,&last_c);
				b=b->link;
				break;

			case 0:
				//a->expon = b->expon
			sum=a->coef+b->coef;
			if(sum) attach(sum,a->expon,&last_c);
			a=a->link;
				b=b->link;
			break;

			case 1:
```

```
                                //a->expon > b-expon
                                attach(a->coef,a->expon,&last_c);
                                a=a->link;
                        }
                }

                for(;a!=last_a->link;a=a->link)//remaining terms in a
                        attach(a->coef,a->expon,&last_c);

                for(;b!=last_b->link;b=b->link)//remaining terms in b
                        attach(a->coef,a->expon,&last_c);

                last_c->link=c;//link the last node of c to first node

}


void display_poly(polyNode *temp,polyNode *last)
{
        while(temp!=last->link)
        {
                printf("%dX^%d + ",temp->coef,temp->expon);
                temp=temp->link;
        }

}

void peval(polyNode *temp,polyNode *last)
{
        int eval=0,x;
        printf("Enter the value of x\n");
        scanf("%d",&x);
        while(temp!=last->link)
        {
                eval=eval+temp->coef*pow(x,temp->expon);
                temp=temp->link;
        }
        printf("The result of polynomial evaluation is: %d\n",eval);
}

void main()
{
        int co,ex,n,m,i;
        printf("******POLYNOMIAL ADDITION *******\n");
        printf("Enter the number of terms in 1st polynomial\n");
        scanf("%d",&m);
```

```
printf("Enter the number of terms in 2nd polynomial \n");
scanf("%d",&n);

a=(polyNode*)malloc(sizeof(polyNode));//headernode of a
last_a=a;
last_a->link=last_a;

b=(polyNode*)malloc(sizeof(polyNode));//headernode of b
last_b=b;
last_b->link=last_b;

//read 1st polynomial-a
for(i=1;i<=m;i++)
{
        printf("Enter the %d term (coef and expon) of 1st polynomial\n",i);
        scanf("%d%d",&co,&ex);
        attach(co,ex,&last_a);
        last_a->link=a;//circular list
}


//read 2nd polynomial -b
for(i=1;i<=n;i++)
{
        printf("Enter the %d term (coef and expon) of 2nd polynomial\n",i);
        scanf("%d%d",&co,&ex);
        attach(co,ex,&last_b);
        last_b->link=b;//circular list
}

//display a and b alon with result c

printf("*** 1st Polynomial a(x):");
display_poly(a->link,last_a);
printf("\n");
printf("*** 2nd Polynomial b(x):");
display_poly(b->link,last_b);
printf("\n");

c=(polyNode*)malloc(sizeof(polyNode));//headernode of a
last_c=c;
cpadd(a->link,b->link);
printf("*** Result c(x)       :");
display_poly(c->link,last_c);
printf("\n");
printf("\n---------------------------------\n");
```

```
        //evaluation of th result c
        printf("******POLYNOMIAL EVALUATION*******\n");
        peval(c->link,last_c);

}
```

LAB PROGRAM 10

Design, Develop and Implement a menu driven Program in C for the following operations on **Binary Search Tree (BST)** of Integers

a. Create a BST of **N** Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (**KEY**) and report the appropriate message

e. Exit

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

struct node
{
        int info;
        struct node *llink;
        struct node *rlink;
};

typedef struct node *NODE;

NODE create(int item,NODE root)
{
        NODE temp,cur,prev;
        temp=(struct node*)malloc(sizeof(struct node));
        temp->info=item;
        temp->llink=NULL;
        temp->rlink=NULL;


        if(root ==NULL)
        return temp;

        prev=NULL;//intially
        cur=root;

        while(cur !=NULL)
        {

                prev=cur;
                if(item<cur->info)
                        cur=cur->llink;
```

```
            else
                cur=cur->rlink;
        }


        if(item<prev->info)
            prev->llink=temp;
        else
            prev->rlink=temp;

        return root;
}


NODE search(int item, NODE root)
{

        NODE cur;
        if(root ==NULL)
            return NULL;

        cur=root;
        while(cur!=NULL)
        {
            if(item == cur->info) return cur;

            if(item<cur->info)
                cur=cur->llink;
            else
                cur=cur->rlink;
        }
        return NULL;
}


void inorder(NODE root)
{

        if(root==NULL) return;
        inorder(root->llink);//traverse L
        printf("%d", root->info);//traverse N
        inorder(root->rlink);//traverse R

}
```

```
void preorder(NODE root)
{

        if(root==NULL) return;
        printf("%d", root->info);//traverse N
        preorder(root->llink);//traverse L
        preorder(root->rlink);//traverse R

}



void postorder(NODE root)
{

        if(root==NULL) return;
        postorder(root->llink);//travrsev
        postorder(root->rlink);
        printf("%d", root->info);
}




void main()
{
        NODE root,cur;
        int ch,item;

        root=NULL;
        for(;;)
        {
                printf("\n******* BINARY SEARCH TREE *******\n");
                printf("1. CREATE\t\t2. INORDER\n");
                printf("3. PREORDER\t\t4. POSTORDER\n");
                printf("5. SEARCH\t\t6. EXIT\n");
                printf("ENTER THE CHOICE\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter the item to be inserted \n");
                                scanf("%d",&item);
                                root=create(item,root);
                        break;
```

```
case 2:
        if(root==NULL)
        {
                printf("Tree empty \n");
                break;
        }
        printf("Inorder traversal is: ");
        inorder(root);
break;



case 3:
        if(root==NULL)
        {
                printf("Tree empty \n");
                break;
        }
        printf("Preorder traversal is: ");
        preorder(root);
break;

case 4:
        if(root==NULL)
        {
                printf("Tree empty \n");
                break;
        }
        printf("Postorder traversal is: ");
        postorder(root);
break;


case 5:
        printf("Enter the item to be searched \n");
        scanf("%d",&item);

        cur= search(item,root);

        if(cur==NULL)
                printf("Item not found \n");
        else
                printf("Item found \n");
```

```
                    break;
                    default:exit(0);
                }//end switch
        }//end infinte for
}//end main
```

LAB PROGRAM 11

Design, Develop and Implement a Program in C for the following operations on **Graph(G)** of Cities

a. Create a Graph of **N** cities using Adjacency Matrix.

b. Print all the nodes **reachable** from a given starting node in a digraph using DFS/BFS method

```
#include<stdio.h>
#include<stdlib.h>
void create(int a[10][10],int n)
{
	int i,j;
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
			scanf("%d",&a[i][j]);
		}
	}
}

void bfs(int a[10][10],int n,int u)
{
	int f,r,q[10],v;
	int s[10]={0};
	printf("Nodes visited from %d are:",u);
	f=0,r=-1;
	q[++r]=u; //insert u into q
	s[u]=1;
	printf("%d",u);
	while(f<=r)
	{
		u=q[f++];
		for(v=0;v<n;v++)
		{
			if(a[u][v]==1)
			{
				if(s[v]==0)
				{
					printf("%d",v);
					s[v]=1;
					q[++r]=v;
				}//end if
			}//end if
```

```
                }//end for
        }//end while
        printf("\n");
}//end function

void main()
{
        int n,a[10][10],src,i,j;

        printf("Enter the number of cities :");
        scanf("%d",&n);
        printf("Enter the adjacency matrix \n");
        create(a,n);

        for(src=0;src<n;src++)
                bfs(a,n,src);
}//end main
```

LAB PROGRAM 12

Given a File of **N** employee records with a set **K** of Keys(4-digit) which uniquely determine the records in file **F**. Assume that file **F** is maintained in memory by a Hash Table(HT) of **m** memory locations with **L** as the set of memory addresses (2-digit) of locations in HT. Let the keys in **K** and addresses in **L** are Integers. Design and develop a Program in C that uses Hash function **H: K->L** as H(**K**)=**K** mod **m (remainder** method), and implement hashing technique to map a given key **K** to the address space **L.** Resolve the collision (if any) using **linear probing**.

```
#include<stdio.h>
#include<stdio.h>
#include<stdlib.h>
#define m 5
int a[10],l;
void init_hashtable()
{
        int i;
        for(i=0;i<m;i++)//m is hash size
                a[i]=0;
}

void display()
{
        int i;
        printf("******Hash table ******\n");
        printf("Locat :");
        for(i=0;i<m;i++)//print location
        {
                printf(" %d\t",i);

        }
        printf("\n");
        printf("Value :");
        for(i=0;i<m;i++)//print the values
        {
                printf(" %d\t",a[i]);

        }
}
```

```
int compute_hashvalue(int k)
{
	return k%m;
}

void main()
{
	int i,k,ch,c=0;
	init_hashtable();//initialize hash values to 0
	display();// display the hash table

	printf("\n1.Insert key\n2.Resolve collision-Linear Probing\n3.Exit \n");
	while(1)
	{
		printf("\nEnter your choice: ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				printf("Enter the key k which is a four digit integer: ");
				scanf("%d",&k);
				l=compute_hashvalue(k);
				printf("\nThe item is: %d \nThe position is: %d\n",k,l);

				if(a[l]!=0)//detect collision
				{
					printf("!!! COLLISION HAS OCCURED !!!\n");
					break;
				}
				else
				{
					a[l]=k;
					display();
				}
			break;

			case 2:
				c=0;
				while(a[l]!=0 && c<m)
				{
					l=(l+1)%m;
					c++;//find empty slot using linear probing
				}
				if(c==m)//insert if empty slot found
				{
					printf("Hash table full \n");
```

```
                    }
                    else
                    {
                        a[l]=k;
                        display();
                    }

                break;

                case 3:exit(0);
                break;
                default: printf("WRONG CHOICE \n");
                break;
            }
        }
}
```