	SUBJECT TITLE: **DATA STRUCTURES WITH C LAB**		
	SUBJECT CODE:CSL36	**No. of Credits:0:0:1:0**	**No. of Lecture hours per week:2**
	Exam Duration :3 hours	**Exam Marks: 50**	

Course objectives:

The objectives of this course are:

1. To develop skills to design and analyze simple linear and non linear data structures.
2. To Strengthen the ability to identify and apply the suitable data structure for the given real world problem by developing algorithms for manipulating stacks, queues, linked lists, trees.
3. To understand recursion concept.
4. To explore sorting techniques.

1.	**(SORTING) →** Write a C program to sort the given 'n' elements using i) Insertion Sort ii) Bucket Sort
2.	**(SEARCH IN SPARSE MATRIX)→**Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix
3.	**(STACKS)→**Write a C Program to construct a stack of integers and to perform the following operations on it: a. Push b. Pop c. Display The program should print appropriate messages for stack overflow, stack underflow, and stack empty.
4.	**(INFIX TO POSTFIX)→**Design, develop, and execute a program in C to convert a given valid parenthesized infix arithmetic expression to postfix expression and then to print both the expressions. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).
5.	**(EVALUATE A POSTFIX EXPRESSION)→**Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).
6.	**(QUEUE)→**Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display
7.	**(CIRCULAR QUEUE)→**Write a C Program to simulate the working of a circular queue of integers using an array. Provide the following operations: a. Insert b. Delete c. Display
8.	**(STACKS USING SINGLY LINKED LIST)→**Write a C Program using dynamic variables and pointers to construct a stack of integers using singly linked list and to perform the following operations: a. Push b. Pop c. Display The program should print appropriate messages for stack overflow and stack empty.

9.	**(QUEUES USING SINGLY LINKED LIST)→**Write a C program using dynamic variables and pointers to construct a queue of integers using singly linked list and to perform the following operations: a. Insert b. Delete c. Display The program should print appropriate messages for queue full and queue empty.
10.	**(POLYNOMIAL ADDITION USING LINLKED LIST)→**Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.
11.	**(DOUBLY LINKED LIST)→**Design, develop, and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations: i. Create a doubly linked list by adding each node at the front. ii. Insert a new node to the left of the node whose key value is read as an input. iii. Delete the node of a given data if it is found, otherwise display appropriate message. iv. Display the contents of the list. (Note: Only either (a,b and d) or (a, c and d) may be asked in the examination)
12.	**(TREES)→**Write a C Program a. To construct a binary search tree of integers. b. To traverse the tree using all the methods ▪ Inorder, Preorder, Postorder. c. To display the elements in the tree.
13.	**(MAX HEAP CREATION)→**Design, develop, and execute a program in C to create a max heap of integers by accepting one element at a time and by inserting it immediately in to the heap. Use the array representation for the heap. Display the array at the end of insertion phase.
14.	**(RECURSION)→**Write recursive C Programs for a. Searching an element on a given list of integers using the Binary Search method. b. Solving the Towers of Hanoi problem.

Course Outcomes:

At the end of this lab session, the student will

CO1: Be able to design & analyse the appropriate data structure for given problem.
CO2: Be capable to identify the appropriate data structure for given problem.
CO3: Be able to Solve a problem using Recursion.
CO4: Be able to compare different sorting techniques.

Cos	Mapping with POs
CO1	PO1,PO2,PO3,PO4, PO9, PO12
CO2	PO1,PO2,PO3,PO4, PO9, PO12
CO3	PO1,PO2,PO3,PO4, PO9, PO12
CO4	PO1,PO2,PO3,PO4, PO9, PO12

LAB PROGRAM 1: (SORTING) → Write a C program to sort the given 'n' elements using i) Insertion Sort ii) Bucket Sort

// INSERTION SORT PROGRAM

```
#include<stdio.h>

#include<conio.h>


void main()

{

int i, j, num, temp, arr[20];

clrscr();

printf("Enter size of array: ");

scanf("%d", &num);


printf("Enter %d elements in arry: \n", num);

for(i=0; i<num; i++)

{

scanf("%d", &arr[i]);

}


for(i=1; i<num; i++)

{

temp=arr[i];

j=i-1;

while((temp<arr[j])&&(j>=0))
```

```
{
arr[j+1]=arr[j];
j=j-1;
}
arr[j+1]=temp;
}

printf("After Sorting elements: ");
for(i=0; i<num; i++)
{
printf("%d  ", arr[i]);
}
getch();
}
```

OUTPUT

Enter Size of an Array: 5

Enter 5 array Elements

5

4

3

2

1

After sorting elements: 1 2 3 4 5

// BUCKET SORT PROGRAM

```
#include<stdio.h>

#define SIZE 100


void bucketSort(int a[], int n) {

   int i, j, k, buckets[SIZE];


   for(i = 0; i < SIZE; ++i)

         buckets[i] = 0;


   for(i = 0; i < n; ++i)

         ++buckets[a[i]];


   for(i = 0, j = 0; j < SIZE; ++j)

         for(k = buckets[j]; k > 0; --k)

            a[i++] = j;

}


Void  main()

{

   int a[100], i, n;

   clrscr();

   printf("Enter the size of array : ");

   scanf("%d", &n);

   printf("enter array elements\n");

   for(i=0;i<n;i++)
```

```
    scanf("%d",&a[i]);
    printf("Before sorting:\n");
    for(i = 0; i < n; ++i)
    printf("%d ", a[i]);
    bucketSort(a, n);
    printf("\n\nAfter sorting:\n");
    for(i = 0; i < n; ++i)
    printf("%d ", a[i]);
    getch();
}
```

OUTPUT

Enter Size of an Array: 5

Enter 5 array Elements

50

40

30

20

10

After sorting elements: 10 20 30 40 50

LAB PROGRAM 2: (SEARCH IN SPARSE MATRIX)→Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix.

```
#include<stdio.h>

#include<conio.h>

#define MAX_TERMS 101

int k;


typedef struct

{

int row,col,val;

}TERM;


//FUNCTION TO READ SPARSE MATRIX AS A TRIPLE

void read_sparse_matrix(TERM a[],int m,int n)

{

int i,j,item;

a[0].row=m;

a[0].col=n;

k=1;

printf("Enter the elements\n");

for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

{
```

```
scanf("%d",&item);
if(item==0) continue;
a[k].row=i;
a[k].col=j;
a[k].val=item;
printf("Non zero element is stored in location a[%d].val=%d\n",k,a[k].val);
k++;
}
}
a[0].val=k-1;
}

void print_sparse_matrix(TERM a[])
{
int p;
printf("Non zero elements are present in the following location \n");
for(p=1;p<k;p++)
{
printf("row=%d   col=%d   val=%d\n",a[p].row,a[p].col,a[p].val);
}
}
//FUNCTION TO SEARCH FOR AN ITEM IN SPARSE MATRIX
void search(TERM a[],int item)
{
int i,j;
for(i=0;i<k;i++)
```

```
{
if(item==a[i].val)
{
printf("Search is successful, Element found at pos %d\n",i);
getch();
exit(0);
}
}
printf("Search is unsuccessful\n");
}

void main()
{
int m,n,item;
TERM a[MAX_TERMS];
clrscr();
printf("Enter the number of Rows & Columns\n");
scanf("%d %d",&m,&n);
read_sparse_matrix(a,m,n);
print_sparse_matrix(a);
printf("Enter the element to be searched\n");
scanf("%d",&item);
search(a,item);
getch();
}
```

OUTPUT

```
Enter the number of Rows & Columns
3                    3
Enter the elements
1
Non zero element is stored in location a[1].val=1
0
0
0
2
Non zero element is stored in location a[2].val=2
4
Non zero element is stored in location a[3].val=4
0
0
0
Non zero elements are present in the following location
row=0   col=0   val=1
row=1   col=1   val=2
row=1   col=2   val=4
Enter the element to be searched
4
Search is successful, Element found at pos 3
```

LAB PROGRAM 3: (STACKS)→Write a C Program to construct a stack of integers and to perform the following operations on it:

a. Push

b. Pop

c. Display

The program should print appropriate messages for stack overflow, stack underflow, and stack empty.

```
#include<stdio.h>

#include<conio.h>

#define STACK_SIZE 5

int top,s[10],item;

void push()

{

if(top==STACK_SIZE-1)

{

printf("stack full\n");

return;

}

top=top+1;

s[top]=item;

}

int pop()

{

 int item_deleted;

if(top==-1) return 0;

item_deleted=s[top--];
```

```
return item_deleted;
}
void display()
{
int i;
if(top==-1)
{
printf("stack is empty\n");
return;
}
printf("contents of the stack\n");
for(i=top;i>=0;i--)
{
printf("%d\n",s[i]);
}
}
void main()
{
int choice;
top=-1;
clrscr();
for(;;)
{
printf("1:PUSH 2:POP 3:DISPLAY 4:EXIT\n");
scanf("%d",&choice);
switch(choice)
```

```
{
case 1 :printf("enter the item\n");
        scanf("%d",&item);
        push();
        break;
case 2 : item=pop();
        if(item==-1)
        printf("stack is empty\n");
        else
        printf("item deleted =%d\n",item);
        break;
case 3 :display();
        break;
default:exit(0);
}
}
}
```

OUTPUT

```
1:PUSH 2:POP 3:DISPLAY 4:EXIT
1
enter the item
10
1:PUSH 2:POP 3:DISPLAY 4:EXIT
1
enter the item
20
1:PUSH 2:POP 3:DISPLAY 4:EXIT
3
contents of the stack
20
10
1:PUSH 2:POP 3:DISPLAY 4:EXIT
2
item deleted =20
1:PUSH 2:POP 3:DISPLAY 4:EXIT
1
enter the item
50
1:PUSH 2:POP 3:DISPLAY 4:EXIT
3
contents of the stack
 50  10
```

LAB PROGRAM 4: (INFIX TO POSTFIX) →Design, develop, and execute a program in C to convert a given valid parenthesized infix arithmetic expression to postfix expression and then to print both the expressions. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

```
#include<stdio.h>

#include<string.h>

#include<conio.h>

int F(char symbol)

{

switch(symbol)

{

case '+':

case '-':return 2;

case '*':

case '/':return 4;

case '^':

case '$':return 5;

case '(':return 0;

case '#':return -1;

default:return 8;

}

}

int G(char symbol)

{

switch(symbol)
```

```
{
case '+':
case '-':return 1;
case '*':
case '/':return 3;
case '^':
case '$':return 6;
case '(':return 9;
case ')':return 0;
default:return 7;
}
}
void infix_postfix(char infix[],char postfix[])
{
int i,j,top;
char s[30],symbol;
top=-1;
top=top+1;
s[top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
{
postfix[j]=s[top--];
```

```
j=j+1;
}
if(F(s[top])!=G(symbol))
{
top=top+1;
s[top]=symbol;
}
else
top=top-1;
}
while(s[top]!='#')
{
postfix[j++]=s[top--];
}
postfix[j]='\0';
}
void main()
{
char infix[20],postfix[20];
clrscr();
printf("enter a valid infix expression\n");
gets(infix);
infix_postfix(infix,postfix);
printf("postfix expression is \n");
printf("%s\n",postfix);
getch();   }
```

OUTPUT

1. enter a valid infix expression

(A+B)

postfix expression is

AB+

2.enter a valid infix expression

((A+B)*C-(D-E))^(F+G)

postfix expression is

AB+C*DE--FG+^

3.enter a valid infix expression

A^B*C-D+E/F/(G+H)

postfix expression is

AB^C*D-EF/GH+/+

LAB PROGRAM 5: (EVALUATE A POSTFIX EXPRESSION)→Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<ctype.h>

int compute(char symbol,int op1,int op2)

{

switch(symbol)

{

case '+':return op1+op2;

case '-':return op1-op2;

case '*':return op1*op2;

case '/':return op1/op2;

case '$':

case '^':return pow(op1,op2);

}

}

void main()

{

int s[20],op1,op2,res;

int i,top;

char postfix[20],symbol;
```

```
clrscr();
top=-1;
printf("Enter the postfix expression\n");
gets(postfix);
for(i=0;i<strlen(postfix);i++)
{
symbol=postfix[i];
if(isdigit(symbol))
{
top=top+1;
s[top]=symbol-'0';
}
else
{
op2= s[top--];
op1=s[top--];
res=compute(symbol,op1,op2);
top=top+1;
s[top]=res;
}
}
res=s[top--];
printf("the result is %d\n",res);
getch();
}
```

OUTPUT

1. Enter the postfix expression

23+

the result is 5

2.Enter the postfix expression

632-5*+1^7+

the result is 18

3.Enter the postfix expression

123+*321-+*

the result is 20

4.Enter the postfix expression

12+3-21+3^-

the result is -27

LAB PROGRAM 6: (QUEUE)→Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations:

a. Insert b. Delete c. Display

```
#include<stdio.h>

#include<conio.h>

#define QSIZE 5


int choice,item,f,r,q[10];


void insert_rear()

{

if(r==QSIZE-1)

{

printf("Q full\n");

return;

}

r=r+1;

q[r]=item;

}


void delete_front()

{

if(f>r)

{

 printf("Q underflow\n");

 return;
```

```
 }
printf("Element Deleted is %d\n",q[f]);
f++;
if(f>r)
{
f=0;r=-1;
}
}

void display()
{
int i;

if(f>r)
{
printf("Q is empty\n");
return;
}
printf("contents of the queue\n");
for(i=f;i<=r;i++)
{
printf("%d\n",q[i]);
}
}
void main()
{
```

```
f=0;
r=-1;
clrscr();
for(;;)
{
printf("1:INSERT 2:DELETE 3;DISPLAY 4: EXIT\n");
scanf("%d",&choice);
switch(choice)
{
case 1 :printf("enter the item\n");
        scanf("%d",&item);
        insert_rear(item,q,&r);
        break;
case 2 :delete_front(q,&f,&r);
        break;
case 3 :display(q,f,r);
        break;
default:exit(0);
}
}
}
```

OUTPUT

```
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
1
enter the item
10
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
1
enter the item
20
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
3
contents of the queue
10
20
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
2
Element Deleted is 10
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
1
enter the item
50
1:INSERT 2:DELETE 3;DISPLAY 4: EXIT
3
contents of the queue
 20   50
```

LAB PROGRAM 7: (CIRCULAR QUEUE)→Write a C Program to simulate the working of a circular queue of integers using an array. Provide the following operations:

a. Insert

b. Delete

c. Display

```
#include<stdio.h>
#include<process.h>
#define SIZE 3
int item,f,r,count,q[20];

void insert_rear()
{
if(count==SIZE)
{
printf("Element cannot be inserted queue overflow\n");
return;
}
r=(r+1)%SIZE;
q[r]=item;
count=count+1;
}
void delete_front()
{
if(count==0)
```

```
{
printf("queue underflow\n");
return;
}
printf("deleted ele =%d\n",q[f]);
f=(f+1)%SIZE;
count=count-1;
}
void display()
{
int i;
int temp=f;
if(count==0)
{
printf("Q is empty\n");
return;
}
printf("contents of queue\n");
for(i=1;i<=count;i++)
{
printf("value of front==%d and q[%d]=%d\n",temp,temp,q[temp]);
temp=(temp+1)%SIZE;
}
}
```

```
void main()
{
int choice;
clrscr();
f=0;r=-1;
count=0;
for(;;)
{
printf("1:Insert 2:Delete 3: Display 4:exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item to be inserted\n");
scanf("%d",&item);
insert_rear();
break;
case 2: delete_front();
break;
case 3:display();
break;
default:exit(0);
}
}
}
```

OUTPUT

```
1:Insert 2:Delete 3: Display 4:exit
enter the choice
1
enter the item to be inserted
10
1:Insert 2:Delete 3: Display 4:exit
enter the choice
1
enter the item to be inserted
20
1:Insert 2:Delete 3: Display 4:exit
enter the choice
1
enter the item to be inserted
30
1:Insert 2:Delete 3: Display 4:exit
enter the choice
3
contents of queue
value of front==0 and q[0]=10
value of front==1 and q[1]=20
value of front==2 and q[2]=30
1:Insert 2:Delete 3: Display 4:exit
enter the choice
```

```
2
deleted ele =10
1:Insert 2:Delete 3: Display 4:exit
enter the choice
3
contents of queue
value of front==1 and q[1]=20
value of front==2 and q[2]=30
1:Insert 2:Delete 3: Display 4:exit
enter the choice
2
deleted ele =20
1:Insert 2:Delete 3: Display 4:exit
enter the choice
3
contents of queue
value of front==2 and q[2]=30
1:Insert 2:Delete 3: Display 4:exit
enter the choice
2
deleted ele =30
1:Insert 2:Delete 3: Display 4:exit
enter the choice
3
Q is empty
1:Insert 2:Delete 3: Display 4:exit
```

```
enter the choice
1
enter the item to be inserted
50
1:Insert 2:Delete 3: Display 4:exit
enter the choice
1
enter the item to be inserted
60
1:Insert 2:Delete 3: Display 4:exit
enter the choice
1
enter the item to be inserted
70
1:Insert 2:Delete 3: Display 4:exit
enter the choice
3
contents of queue
value of front==0 and q[0]=50
value of front==1 and q[1]=60
value of front==2 and q[2]=70
```

LAB PROGRAM 8: (STACKS USING SINGLY LINKED LIST)→Write a C Program using dynamic variables and pointers to construct a stack of integers using singly linked list and to perform the following operations:

a. Push

b. Pop

c. Display

The program should print appropriate messages for stack overflow and stack empty.

```
//C PROGRAM TO CREATE A LIST AND TO DISPLAY THE CONTENTS OF THE LIST

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<alloc.h>


struct node

{

int info;

struct node *link;

};

typedef struct node *NODE;


NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));
```

```
if(x==NULL)
{
printf("OUT OF MEMORY\n");
getch();
exit(0);
}
return x;
}

NODE insertfront(int item,NODE first)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=first;
return temp;
}
NODE deletefront(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
```

```
temp=temp->link;
printf("item deleted=%d\n",first->info);
free(first);
first=NULL;
return temp;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("List is Empty\n");
return;
}
printf("CONTENTS OF THE SINGLY LINKED LIST\n");
temp=first;
while(temp!=NULL)
{
printf("%d\n",temp->info);
temp=temp->link;
}
printf("\n");
}

void main()
{
```

```
NODE first=NULL;
int choice,item;
clrscr();
for(;;)
{
printf("1:InsertFront 2:DeleteFront 3:Display 4:Quit\n");
printf("Enter ur choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item to be inserted\n");
        scanf("%d",&item);
        first=insertfront(item,first);
        break;
case 2:first=deletefront(first);
        break;
case 3:display(first);
        break;
default:exit(0);
}
}
}
```

OUTPUT

```
1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

10

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

20

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

20

10


1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

2

item deleted=20

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice
```

```
1

Enter the item to be inserted


1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

10

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

20

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

20

10


1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

2

item deleted=20

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice
```

```
1

Enter the item to be inserted

50

1:InsertFront 2:DeleteFront 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

50

10
```

LAB PROGRAM 9: (QUEUES USING SINGLY LINKED LIST)→Write a C program using dynamic variables and pointers to construct a queue of integers using singly linked list and to perform the following operations:

a. Insert

b. Delete

c. Display

The program should print appropriate messages for queue full and queue empty.

```
//C PROGRAM TO CREATE A LIST AND TO DISPLAY THE CONTENTS OF THE LIST


#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<alloc.h>


struct node

{

int info;

struct node *link;

};

typedef struct node *NODE;


NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));
```

```
if(x==NULL)
{
printf("OUT OF MEMORY\n");
getch();
exit(0);
}
return x;
}

NODE insertrear(int item,NODE first)
{
NODE temp;
NODE cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
{
cur=cur->link;
}
cur->link=temp;
return first;
}
```

```
NODE deletefront(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted=%d\n",first->info);
free(first);
first=NULL;
return temp;
}

void display(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("List is Empty\n");
return;
}
printf("CONTENTS OF THE SINGLY LINKED LIST\n");
```

```
temp=first;
while(temp!=NULL)
{
printf("%d\n",temp->info);
temp=temp->link;
}
printf("\n");
}

void main()
{
NODE first=NULL;
int choice,item;
clrscr();
for(;;)
{
printf("1:Insert Rear 2:Delete Front 3:Display 4:Quit\n");
printf("Enter ur choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item to be inserted\n");
        scanf("%d",&item);
        first=insertrear(item,first);
        break;
case 2:first=deletefront(first);
```

```
        break;
case 3:display(first);
        break;
default:exit(0);
}
}
}
```

OUTPUT

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

10

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

20

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

10

20

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

2

item deleted=10

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

20

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

1

Enter the item to be inserted

50

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST

20

50

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

1

```
Enter the item to be inserted

60

1:Insert Rear 2:Delete Front 3:Display 4:Quit

Enter ur choice

3

CONTENTS OF THE SINGLY LINKED LIST   20   50   60
```

LAB PROGRAM 10: (POLYNOMIAL ADDITION USING LINLKED LIST)→Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<alloc.h>

#include<math.h>


//STRUCTURE DEFINITION


struct node

{

int coeff;

int expon;

struct node *link;

};


typedef struct node *NODE;


//CREATING NEW NODE USING DYNAMIC MEMORY ALLOCATION


NODE getnode()

{

NODE x;
```

```
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("OUT OF MEMORY\n");
getch();
exit(0);
}
return x;
}

//FUNCTION TO INSERT NEW NODES i.e ADDING/ATTACHING TERMS TO POLYNOMIAL

NODE attach(int coeff,int expon,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->coeff=coeff;
temp->expon=expon;
cur=head->link;
while(cur->link!=head)
{
cur=cur->link;
}
cur->link=temp;
temp->link=head;
return head;
```

```
}
//FUNCTION TO READ A POLYNOMIAL
NODE read_poly(NODE head)
{
int i;
int coeff,expon;
printf("Enter co-efficient as -999 to end the polynomial\n");
for(i=1;;i++)
{
printf("Enter %d terms\n",i);
printf("Enter co-efficient\n");
scanf("%d",&coeff);
if(coeff==-999)
break;
printf("Enter  power of x\n");
scanf("%d",&expon);
head=attach(coeff,expon,head);
}
return head;
}
//FUNCTION TO ADD 2 POLYNOMIALS
NODE add_poly(NODE h1,NODE h2,NODE h3)
{
NODE a,b;
int coeff,val;
```

```
a=h1->link;
b=h2->link;
while(a!=h1 && b!=h2)
{
if(a->expon==b->expon)
{
val=0;
}
else if(a->expon>b->expon)
{
val=1;
}
else
{
val=2;
}
switch(val)
{
case 0:coeff=a->coeff+b->coeff;
        if(coeff!=0)
        h3=attach(coeff,a->expon,h3);
        a=a->link;
        b=b->link;
        break;
case 1:h3=attach(a->coeff,a->expon,h3);
        a=a->link;
```

```
        break;
default:h3=attach(b->coeff,b->expon,h3);
        b=b->link;
}
}
while(a!=h1)
{
h3=attach(a->coeff,a->expon,h3);
a=a->link;
}
while(b!=h2)
{
h3=attach(b->coeff,b->expon,h3);
b=b->link;
}
return h3;
}

//FUNCTION TO DISPLAY A POLYNOMIAL
void display(NODE head)
{
NODE temp;
if(head->link==head)
{
printf("POLYNOMIAL DOES NOT EXIST\n");
return;
```

```
}
temp=head->link;

while(temp!=head)
{
if(temp->coeff<0)
printf("%dx^%d",temp->coeff,temp->expon);
else
printf("+%dx^%d",temp->coeff,temp->expon);
temp=temp->link;
}
}

//MAIN FUNCTION
void main()
{
NODE h1,h2,h3;
int choice;
clrscr();
h1=getnode();
h2=getnode();
h3=getnode();
h1->link=h1;
h2->link=h2;
h3->link=h3;
printf("Enter the 1st POLYNOMIAL\n");
```

```
h1=read_poly(h1);
printf("Enter the 2nd POLYNOMIAL\n");
h2=read_poly(h2);
h3=add_poly(h1,h2,h3);
printf("The 1st POLYNOMIAL is \n");
display(h1);
printf("\nThe 2ND POLYNOMIAL is \n");
display(h2);
printf("\nThe 3rd POLYNOMIAL is \n");
display(h3);
getch();
}
```

OUTPUT

Enter the 1st POLYNOMIAL

Enter co-efficient as -999 to end the polynomial

Enter 1 terms

Enter co-efficient

5

Enter power of x

4

Enter 2 terms

Enter co-efficient

8

Enter power of x

3

Enter 3 terms

Enter co-efficient

2

Enter power of x

2

Enter 4 terms

Enter co-efficient

-999

Enter the 2nd POLYNOMIAL

Enter co-efficient as -999 to end the polynomial

Enter 1 terms

Enter co-efficient

7

```
Enter  power of x

4

Enter 2 terms

Enter co-efficient

6

Enter  power of x

2

Enter 3 terms

Enter co-efficient

-999

The 1st POLYNOMIAL is

+5x^4+8x^3+2x^2

The 2ND POLYNOMIAL is

+7x^4+6x^2

The 3rd POLYNOMIAL is

+12x^4+8x^3+8x^2
```

LAB PROGRAM 11: (DOUBLY LINKED LIST)→Design, develop, and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations:

i. **Create a doubly linked list by adding each node at the front.**

ii. **Insert a new node to the left of the node whose key value is read as an input.**

iii. **Delete the node of a given data if it is found, otherwise display appropriate message.**

iv. **Display the contents of the list.**

(Note: Only either (a,b and d) or (a, c and d) may be asked in the examination)

```
//C PROGRAM TO CREATE A CIRCULAR DOUBLY LINKED LIST AND TO DISPLAY THE
CONTENTS OF THE LIST

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<alloc.h>


//STRUCTURE DEFINITION

struct node

{

int info;

struct node *llink;

struct node *rlink;

};

typedef struct node *NODE;


//FUNCTION TO CREATE NEW NODES DYNAMICALLY
```

```
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("OUT OF MEMORY\n");
getch();
exit(0);
}
return x;
}

//FUNCTION TO INSERT NEW NODES FROM FRONT END
NODE insertfront(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
```

```
//FUNCTION TO INSERT NEW NODES BEFORE THE KEY ELEMENT SPECIFIED
NODE insertleft(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("List is empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
printf("Key not found\n");
return head;
}
prev=cur->llink;
printf("Enter the item to insert towards left of %d=\n",item);
scanf("%d",&item);
temp=getnode();
temp->info=item;
prev->rlink=temp;
```

```
temp->llink=prev;
cur->llink=temp;
temp->rlink=cur;
return head;
}
//FUNCTION TO IDELTE THE KEY ELEMENT SPECIFIED
NODE deleteitem(int item,NODE head)
{
NODE cur,prev,next;
if(head->rlink==head)
{
printf("list is empty cannot delete\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
printf("item not found\n");
return head;
}
prev=cur->llink;
```

```
next=cur->rlink;
prev->rlink=next;
next->llink=prev;
free(cur);
return head;
}

//FUNCTION TO DISPLAY THE CONTENTS OF THE DLL
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("List is Empty\n");
return;
}
printf("CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\n",temp->info);
temp=temp->rlink;
}
}
//MAIN FUNCTION
void main()
```

```
{
NODE head;
int choice,item;
clrscr();
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
printf("1:Insertfront 2:InsertLeftKey\n3:DeleteKey 4:Display 5:Quit\n");
printf("Enter ur choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item to be inserted\n");
        scanf("%d",&item);
        head=insertfront(item,head);
        break;
case 2:printf("Enter the key element\n");
        scanf("%d",&item);
        head=insertleft(item,head);
        break;
 case 3:printf("Enter the item to be deleted\n");
        scanf("%d",&item);
        head=deleteitem(item,head);
        break;
```

```
case 4:display(head);
        break;
default:exit(0);
}
}
}
```

OUTPUT

```
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
1
Enter the item to be inserted
10
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
1
Enter the item to be inserted
20
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
1
Enter the item to be inserted
30
1:Insertfront 2:InsertLeftKey
```

```
3:DeleteKey 4:Display 5:Quit

Enter ur choice

4

CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST

30

20

10

1:Insertfront 2:InsertLeftKey

3:DeleteKey 4:Display 5:Quit

Enter ur choice

2

Enter the key element

20

Enter the item to insert towards left of 20=

50

1:Insertfront 2:InsertLeftKey

3:DeleteKey 4:Display 5:Quit

Enter ur choice

4

CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST

30

50

20

10

1:Insertfront 2:InsertLeftKey

3:DeleteKey 4:Display 5:Quit
```

```
Enter ur choice
20
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
1
Enter the item to be inserted
30
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
4
CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST
30
20
10
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
Enter ur choice
2
Enter the key element
20
Enter the item to insert towards left of 20=
50
1:Insertfront 2:InsertLeftKey
3:DeleteKey 4:Display 5:Quit
```

```
Enter ur choice

4

CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST

30

50

20

10

1:Insertfront 2:InsertLeftKey

3:DeleteKey 4:Display 5:Quit

Enter ur choice

3

Enter the item to be deleted

10

1:Insertfront 2:InsertLeftKey

3:DeleteKey 4:Display 5:Quit

Enter ur choice

4

CONTENTS OF THE CIRCULAR DOUBLY LINKED LIST

30

50

20
```

LAB PROGRAM 12: (TREES)→Write a C Program

a. To construct a binary search tree of integers.

b. To traverse the tree using all the methods

- **Inorder,**
- **Preorder**
- **Postorder.**

c. To display the elements in the tree.

```
#include<stdio.h>

#include<conio.h>


struct node

{

int info;

struct node *llink;

struct node *rlink;

};

typedef struct node *NODE;


NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{
```

```
printf("OUT OF MEMORY\n");
getch();
exit(0);
}
return x;
}
//Function to insert items into a binary search tree
NODE insert(int item,NODE root)
{
NODE temp,cur,prev;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
if(root==NULL)
return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
if(item==cur->info)
{
printf("Duplicate items not allowed\n");
free(temp);
return root;
```

```
}
if(item<cur->info)
cur=cur->llink;
else
cur=cur->rlink;
}
if(item<prev->info)
prev->llink=temp;
else
prev->rlink=temp;
return root;
}

//Fuction to display the tree
void preorder(NODE root)
{
if(root==NULL)return;
printf("%d  ",root->info);
preorder(root->llink);
preorder(root->rlink);
}
void inorder(NODE root)
{
if(root==NULL)return;
inorder(root->llink);
printf("%d  ",root->info);
```

```
inorder(root->rlink);
}
void postorder(NODE root)
{
if(root==NULL)return;
postorder(root->llink);
postorder(root->rlink);
printf("%d  ",root->info);
}
void main()
{
NODE root;
int choice,item,x;
clrscr();
root=NULL;
for(;;)
{
printf("Enter your choice\n1:Insert 2:Display, PREORDER+INORDER+POSTORDER
3:Exit\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item to be inserted\n");
        scanf("%d",&item);
        root=insert(item,root);
        break;
```

```
case 2:if(root==NULL)
        printf("Tree is empty\n");
        else
        {
        printf("PREOREDER TRAVERSAL\n");
        preorder(root);
        printf("\n");
        printf("INOREDER TRAVERSAL\n");
        inorder(root);
        printf("\n");
        printf("POSTOREDER TRAVERSAL\n");
        postorder(root);
        printf("\n");
        }
        break;
default:exit(0);
}
}
}
```

OUTPUT 1

```
Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

10

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

20

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

30

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

40

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

2
```

PREOREDER TRAVERSAL

10 20 30 40

INOREDER TRAVERSAL

10 20 30 40

POSTOREDER TRAVERSAL

40 30 20 10

<u>OUTPUT2</u>

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

100

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

50

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

200

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

```
Enter the item to be inserted
25
Enter your choice
1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit
1
Enter the item to be inserted
90
Enter your choice
1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit
1
Enter the item to be inserted
150
Enter your choice
1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit
1
Enter the item to be inserted
300
Enter your choice
1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit
1
Enter the item to be inserted
80
Enter your choice
1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit
1
Enter the item to be inserted
```

```
180

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit



Enter the item to be inserted

25

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

90

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

150

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

300

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

80
```

```
Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

1

Enter the item to be inserted

180

Enter your choice

1:Insert 2:Display, PREORDER+INORDER+POSTORDER 3:Exit

2

PREOREDER TRAVERSAL

100  50  25  90  80  200  150  180  300

INOREDER TRAVERSAL

25  50  80  90  100  150  180  200  300

POSTOREDER TRAVERSAL

25  80  90  50  180  150  300  200  100
```

LAB PROGRAM 13: (MAX HEAP CREATION)→Design, develop, and execute a program in C to create a max heap of integers by accepting one element at a time and by inserting it immediately in to the heap. Use the array representation for the heap. Display the array at the end of insertion phase.

```
#include<stdio.h>

#include<conio.h>

#define MAXSIZE 20


//FUNCTION TO CREATE A MAXHEAP


int maxheap(int item,int a[],int n)

{

int c,p;

if(n==MAXSIZE-1)

{

printf("HEAP IS FULL\n");

return n-1;

}

c=n;

p=c/2;

while(p!=0 && item>a[p])

{

a[c]=a[p];
```

```
c=p;
p=c/2;
}
a[c]=item;
return n;
}

void display(int a[],int n)
{
int i;
if(n==0)
{
printf("Heap is empty\n");
return;
}
printf("CONTENTS OF THE HEAP\n");
for(i=1;i<=n;i++)
printf("%d\n",a[i]);
}

//MAIN FUNCTION
void main()
{
int a[100],n=0,choice,item;
clrscr();
for(;;)
```

```
{
printf("1:INSERT   2:DISPLAY    3:EXIT\n");
printf("Enter UR CHOICE\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter item\n");
      scanf("%d",&item);
      n=maxheap(item,a,n+1);
      break;
case 2:display(a,n);
            break;
default:exit(0);
}
}
getch();
}
```

OUTPUT

```
1:INSERT   2:DISPLAY   3:EXIT

Enter UR CHOICE

1

Enter item

50

1:INSERT   2:DISPLAY   3:EXIT

Enter UR CHOICE

1

Enter item

25

1:INSERT   2:DISPLAY   3:EXIT

Enter UR CHOICE

1

Enter item

30

1:INSERT   2:DISPLAY   3:EXIT

Enter UR CHOICE

1

Enter item

75

1:INSERT   2:DISPLAY   3:EXIT

Enter UR CHOICE
```

```
1
Enter item
100
1:INSERT   2:DISPLAY   3:EXIT
Enter UR CHOICE
1
Enter item
45
1:INSERT   2:DISPLAY   3:EXIT
Enter UR CHOICE
1
Enter item
80
1:INSERT   2:DISPLAY   3:EXIT
Enter UR CHOICE
2
CONTENTS OF THE HEAP
100  75 80 25 50 30 45
```

LAB PROGRAM 14: (RECURSION)→Write recursive C Programs for

a. Searching an element on a given list of integers using the Binary Search method.

```
#include<stdio.h>

#include<conio.h>

//FUNCTION TO SEARCH AN ELEMENT USING BINARY SEARCH LOGIC

int search(int key,int a[],int low,int high)

{

int mid;

if(low>high) return -1;

mid=(low+high)/2;

if(key==a[mid])

return mid;

if(key<a[mid])

search(key,a,low,mid-1);

else

search(key,a,mid+1,high);

}

void main()

{

int n,i,a[20],key,pos;

clrscr();

printf("enter the number of elements\n");

scanf("%d",&n);

printf("Enter the elements\n");
```

```
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the key ele to be searched\n");
scanf("%d",&key);
pos=search(key,a,0,n-1);
if(pos==-1)
printf("Key ele not found\n");
else
printf("Ele found at pos %d\n",pos);
getch();
}
```

OUTPUT

enter the number of elements

5

Enter the elements

5

6

7

8

9

Enter the key ele to be searched

7

Ele found at pos 2

b. Solving the Towers of Hanoi problem.

```
#include<stdio.h>
#include<conio.h>
void tower(int n,char src,char temp,char des)
{
if(n==1)
{
printf("move disc %d from %c to %c\n",n,src,des);
return;
}
tower(n-1,src,des,temp);
printf("move disc %d from %c to %c\n",n,src,des);
tower(n-1,temp,src,des);
}
void main()
{
int n;
clrscr();
printf("Enter the number of discs\n");
scanf("%d",&n);
tower(n,'S','T','D');
getch();
}
```

OUTPUT1

Enter the number of discs

3

move disc 1 from S to D

move disc 2 from S to T

move disc 1 from D to T

move disc 3 from S to D

move disc 1 from T to S

move disc 2 from T to D

move disc 1 from S to D

OUTPUT2

Enter the number of discs

4

move disc 1 from S to T

move disc 2 from S to D

move disc 1 from T to D

move disc 3 from S to T

move disc 1 from D to S

move disc 2 from D to T

move disc 1 from S to T

move disc 4 from S to D

move disc 1 from T to D

move disc 2 from T to S move disc 1 from D to S

move disc 3 from T to D move disc 1 from S to T

move disc 2 from S to D move disc 1 from T to D