C BYREGOWDA INSTITUTE OF TECHNOLOGY

Affiliated to Visvesvaraya Technological University

Approved by AICTE New Delhi & Govt. of Karnataka

Laboratory Manual

DATA STRUCTURES LABORATORY (BCSL305)

III Semester (CBCS Scheme)

Prepared by

VASUDEVA R
Asst. Prof, Dept. of CSE

KAVITHA N
Asst. Prof, Dept. of CSE

ASHOK BABU A
Asst. Prof, Dept. of AIML

Department of Computer Science and Engineering
Department of Artificial Intelligence and Machine Learning

C BYREGOWDA INSTITUTE OF TECHNOLOGY

An ISO 9001:2015 certified Institute

Kolar-Srinivaspur Road, Kolar- 563101

2023-24

Course objectives:

This laboratory course enable students to get practical experience in design, develop, implement, analyze and evaluation/testing of

- Dynamic memory management.
- Linear data structures and their applications such as stacks, queues and lists.
- Non-Linear data structures and their applications such as trees and graphs.

Course outcomes:

On the completion of this laboratory course, the students will be able to:

- Analyze various linear and non-linear data structures.
- Demonstrate the working nature of different types of data structures and their applications.
- Use appropriate searching and sorting algorithms for the give scenario.
- Apply the appropriate data structure for solving real world problems.

Conduction of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution (Need to change in accordance with university regulations)
  - ✓ For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  - ✓ For laboratories having PART A and PART B i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

List of Programs

1 Develop a Program in C for the following:
   a. Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), the second field is the date of the Day (An integer), and the third field is the description of the activity for a particular day (A dynamically allocated String).
   b. Write functions create (), read () and display (); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

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.

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.

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**.**

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

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.

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.

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.

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

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
    d. Exit.

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.

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**.

Introduction to Data Structure

Data is the basic entity or fact that is used in calculation or manipulation process. There are two types of data such as numerical and alphanumerical data. Integer and floating-point numbers are of numerical data type and strings are of alphanumeric data type. Data may be single or a set of values and it is to be organized in a particular fashion. This organization or structuring of data will have profound impact on the efficiency of the program.

Data structure is the structural representation of logical relationships between elements of data. In other words a data structure is a way of organizing data items by considering its relationship to each other. Data structure affects the design of both the structural and functional aspects of a program. Data structures are the building blocks of a program; here the selection of a particular data structure will help the programmer to design more efficient programs as the complexity and volume of the problems solved by the computer is steadily increasing day by day. The programmers have to strive hard to solve these problems.

Definition

Data Structure can be defined as logical or mathematical model of particular organization of data is called data structure.

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently.

It is clear from the above discussion that the data structure and the operations on organized data items can integrally solve the problem using a computer.

Classification of Data Structure

Data structure are Divided into two main categories.

1. **Primitive data structure**
2. **Non-primitive data structure**

Primitive data structure: The primitive data structures are defined that can be manipulated or operated by the machine instruction.

For example- the integers, floating-point numbers, pointers, string constants, characters etc.

Non-primitive data structure-The non-primitive data structures are data structure that cannot be manipulated or operated directly by the machine instructions. These are more sophisticated data structures. These are derived from the primitive data structure**.**

For example-Arrays, structure, stack, queues, linked list etc.

The non-primitive data structures are classified into two categories:

Linear data structure

When the elements are accessed in contiguous memory location then data structure is known as linear data structure.

Example - Stack, queue and linked list.

Non Linear data Structure

In non-linear data structure element are not accessed in contiguous memory allocation

Example-Non-linear data structure are tree graph, sets etc.

Stack Data Structure

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.
- **isEmpty:** Returns true if stack is empty, else false.

Applications of stack: Balancing of symbols

- Infix to Postfix /Prefix conversion
- Redo-undo features at many places like editors, Photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.
- Other applications can be Backtracking, Knight tour problem, rat in a maze, N queen problem and Sudoku solver.

Queue Data Structure

Queue is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **in First out** (FIFO). A good example of queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Operations on Queue:

Mainly the following four basic operations are performed on queue:

- **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
- **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
  Front: Get the front item from queue.
  Rear: Get the last item from queue.

Applications of Queue:

Queue is used when things don't have to be processed immediately, but have to be processed in **F**irst **I**n First **O**ut order .This property of Queue makes it also useful in following kind of scenarios.

- When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.
- When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

Linked List Data Structure

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

Why Linked List?

Arrays can be used to store linear data of similar types, but arrays have following limitations.

- The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Generally, the allocated memory is equal to the upper limit irrespective of the usage.
- Inserting a new element in an array of elements is expensive; because room has to be created for the new elements and to create room existing elements have to shift.

- **Advantages over arrays**
  1) Dynamic size
  2) Ease of insertion/deletion

A tree is a data structure made up of nodes or vertices and edges without having any cycle. The tree with no nodes is called the **null** or **empty** tree. A tree that is not empty consists of a root node and potentially many levels of additional nodes that form a hierarchy.

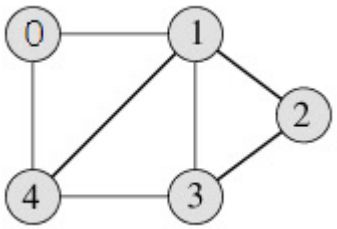
1. **Root:** The top node in a tree.
2. **Child**: A node directly connected to another node when moving away from the Root.
3. **Parent:** The converse notion of a child.
4. **Siblings:** A group of nodes with the same parent.
5. **Descendant:** A node reachable by repeated proceeding from parent to child.
6. **Ancestor:** A node reachable by repeated proceeding from child to parent.
7. **Leaf:** A node with no children.
8. **Branch:** A node with at least one child.
9. **Degree:** The number of sub trees of a node.
10. **Edge:**The connection between one node and another.
11. **Path:** A sequence of nodes and edges connecting a node with a descendant.
12. **Level:** The level of a node is defined by 1 + (the number of connections between the node and the root).

13. **Height of node:** The height of a node is the number of edges on the longest path between that node and a leaf.
14. **Height of tree:** The height of a tree is the height of its root node.
15. **Depth:** The depth of a node is the number of edges from the tree's root node to the node.
16. **Forest:** A forest is a set of $n \geq 0$ disjoint trees.

Graph Data Structure

Graph is a data structure that consists of following two components:
1. A finite set of vertices also called as nodes.
2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph (di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook.
Following is an example undirected graph with 5 vertices.



- Following two are the most commonly used representations of graph.
  Adjacency Matrix
  Adjacency List
-

Adjacency Matrix:

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

1. Develop a Program in C for the following:

a. Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), the second field is the date of the Day (An integer), and the third field is the description of the activity for a particular day (A dynamically allocated String).

b. Write functions create (), read () and display (); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

```
#include <stdio.h>
#include<stdlib.h>
#include <string.h>                    // Define a structure to represent a day
struct Day
{
        char dayname[20];
        int date;
        char activity[100];
};
void create(struct Day calendar[7])          // Function to create the calendar
{
        printf(" Day Intialization Done \n ");
        strcpy(calendar[0].dayname, "Monday");
        strcpy(calendar[1].dayname, "Tuesday");
        strcpy(calendar[2].dayname, "Wednesday");
        strcpy(calendar[3].dayname, "Thursday");
        strcpy(calendar[4].dayname, "Friday");
        strcpy(calendar[5].dayname, "Saturday");
        strcpy(calendar[6].dayname, "Sunday");
}
void read(struct Day calendar[7])       // Function to read data from the keyboard
{
        int i;
        for (i = 0; i < 7; i++)
        {
                printf("\n Enter details for %s:\n", calendar[i].dayname);
                printf(" Date and Activity :  ");
                scanf("%d", &calendar[i].date);
                gets(calendar[i].activity);
        }
}
void display(struct Day calendar[7])
{
        printf("Calendar for the week:\n\n");
        int i;
        for (i = 0; i < 7; i++)
        {
                printf("%s (Date: %d):  ", calendar[i].dayname, calendar[i].date);
                puts(calendar[i].activity);
                printf("\n");
        }
}
```

```
void main()
{
        struct Day *calendar;
        calendar=(struct Day*)calloc(7,sizeof(struct Day));
        int choice;
        while(1)
        {
                printf("\n*********calendar Program*********\n");
                printf("1. Create Calendar\n");
                printf("2. Read Calendar Data\n");
                printf("3. Display Calendar Daywise Activity\n");
                printf("4. Exit \n");
                printf("************************************\n");
                printf("Enter your choice: ");
                scanf("%d", &choice);
                switch (choice)
                {
                        case 1: create(calendar);
                                break;
                        case 2: read(calendar);
                                break;
                        case 3: display(calendar);
                                break;
                        case 4: exit(0);
                }
        }

}
```

OUTPUT

```
*********Calendar Program*********

1. Create Calendar
2. Read Calendar Data
3. Display Calendar Daywise Activity
4. Exit
Enter your choice: 1
Day Intialization Done

*********calendar Program*********

1. Create Calendar
2. Read Calendar Data
3. Display Calendar Daywise Activity
4. Exit

Enter your choice: 2

 Enter details for Monday:

 Date and Activity :  1 class and lab work

 Enter details for Tuesday:
```

Date and Activity : 2 class work

Enter details for Wednesday:
Date and Activity : 3 theory work

Enter details for Thursday:
Date and Activity : 4 exam work

Enter details for Friday:
Date and Activity : 5 practical exams

Enter details for Saturday:
Date and Activity : 6 main exams

Enter details for Sunday:
Date and Activity : 7 NSS activity

*********calendar Program*********

1. Create Calendar
2. Read Calendar Data
3. Display Calendar Daywise Activity
4. Exit

Enter your choice: 3

Calendar for the week:

Monday (Date: 1): class and lab exam
Tuesday (Date: 2): class work
Wednesday (Date: 3): theory work
Thursday (Date: 4): exam work
Friday (Date: 5): practical exams
Saturday (Date: 6): main exams
Sunday (Date: 7): NSS activity

*********calendar Program*********

1. Create Calendar
2. Read Calendar Data
3. Display Calendar Daywise Activity
4. Exit

Enter your choice: 4

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.

Algorithm for String operations

//**input**: Given String, Pattern string and replacement string.
//**ouput**: Resulted string after replacement.
1. Read String.
2. Read pattern String.
3. Read Replace string.
4. Search a pattern in the given string.
    If found
    then
        print "Pattern is found in given string".
        goto step 5.
    Otherwise
        print " Pattern is is not found in given string".
        goto step 6.
    end.
5. Replace all occurrences of Pattern in String with Replace string.
6. Stop.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char str[50],pat[10],rep[10];
void readstring();
void readpattren();
void readreplacestring();
void findandreplace();
void main()
{
        printf("program to find and replace a given string\n");
        readstring();
        readpattren();
        readreplacestring();
        findandreplace();

}
void readstring()
{
        printf("Enter  string\n");
        scanf("%s",str);
}
void readpattren()
{
        printf("enter pattren\n");
```

```
        scanf("%s",pat);
}

void readreplacestring()
{
        printf("enter replacment string\n");
        scanf("%s",rep);
}
void findandreplace()
{
        int i=0,j=0,k;
        while(str[i]!='\0' && pat[j]!='\0')
        {
                if(str[i]!=pat[j])
                        i++;
                else{
                        i++;
                        j++;
                    }
        }
if(pat[j]=='\0')
{
        printf("The given Pattren found in a string\n");
        int pcount=0,rcount=0,scount=0;
        int pos=i, space=0;
          /* Procedure to find the length of each string*/
        for(k=0;pat[k]!='\0';k++)
                pcount++;

        for(k=0;rep[k]!='\0';k++)
                rcount++;

        for(k=0;str[k]!='\0';k++)
                scount++;

        if(rcount>pcount)  //Procedure to create extra space in case replace string is greater than pattern string
        {
                space=rcount-pcount;
                for(i=scount+space-1;i>pos;i--)
                {
                        str[i]=str[i-space];
                }
        }

        k=0;
        for(i=pos-pcount; rep[k]!='\0'; i++) //Procedure to replace a pattern by replace string
        {
                str[i]=rep[k];
                k++;
        }

str[scount+space]='\0';
}
```

```
        else{
                printf("not found\n");

              }
printf("After replacment the string =  %s \n",str);

}
```

OUTPUT 1

```
program to find and replace a given string
Enter  string
cbitkolar
enter pattren
kolar
enter replacment string
bangalore
The given Pattren found in a string
After replacment the string =  cbitbangalore
```

OUTPUT 2:

```
program to find and replace a given string
Enter  string
GoodMorning
enter pattren
bad
enter replacment string
student
not found
 After replacment the string =  GoodMorning
```

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.

Algorithm for PUSH Operation

Step 1: if stack is full **then**

Write "Stack overflow"

Step 2: top ← top + 1

Step3: stack [top] ← data

Algorithm for Pop Operation

Step1: if stack is empty **then**

Write "Stack is Underflow"

Step2: Return Stack [top]

Step3: top ← top -1

Algorithm to check Palindrome using Stack (number)

Step 1: Read Number
Step 2: m ← number
Step 3: while m not equal **to 0**
do
digit ← m%10
n ← n + 1
a[n] ← digit
m ← m / 10
done
Step 4: [pushing elements to stack]
for i ←0 to n in step **of 1**
do
push an element a[i]
done
Step 5: [pop elements]

while top not equal **to -1**
do
digit ← pop()
rev ← rev * 10 + digit
done

Step 6: Stop

```
#include<stdio.h>
#define STACKSIZE 5
#include<stdlib.h>
int choice,top=-1;
int stack[10],item,element;
void push(int);
int pop();
void display();
void palindrome();
void main()
{
	while(1)
	{
		printf("\n************************MENU*************************************\n");
		printf("\n1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit\n");
		printf("\n********************************************************************* *\n");
		printf("\nEnter your choice: ");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1: printf("Enter the item to be inserted: ");
				scanf("%d",&item);
				push(item);
				break;
			case 2: element=pop();
				if(element)
					printf("The item deleted is: %d",element);
				break;
			case 3: printf("Demonstration of how stack can be used to check palindrome\n");
				if(top==-1)
					palindrome();
				else
					printf("Make first stack empty\n");
					break;

			case 4:  printf("*********The status of STACK********\n");
				printf("Total number of elements can be inserted into stack: %d\n",STACKSIZE);
				printf("The total number of elements present in stack is:  %d\n",top+1);
				printf("Number of locations free in Stack: %d\n",(STACKSIZE-top-1));
				break;

			case 5: display();
				break;
			case 6: exit(0);
```

```
                default: printf("Unknown choice\n");
                         break;
                }
        }
}
void push(int item)
{
        if(top==STACKSIZE-1)
        {
                printf("STACK over flow\n");
                return;
        }
        stack[++top]=item;
 }

int pop()
{
        if(top==-1)
        {
                printf("STACK underflow\n");
                return(0);
        }

        return(stack[top--]);
}

void palindrome()
{
        int digit,rev=0,m,lenth,num,a[5],n=0,i;
        printf("Enter the element: ");
        scanf("%d",&num);
        m=num;
        while(m!=0)   //procedure to divide number and to store all digits in array
        {
                digit=m%10;
                 n++;
                 a[n]=digit;
                 m=m/10;
        }
       //Pushing elements into a stack
        for(i=n;i>0;i--)
                push(a[i]);

        while(top!=-1)
        {
                digit=pop();    // poping elements from stack
                rev=rev*10+digit;
        }

         if(num==rev)
                printf("\n The Given Number is palindrome\n");
         else
                printf("\n Given Number is Not a palindrome\n");
}
```

```
void display()
{
        int i;
        if(top==-1)
        {
                printf("STACK is empty\n");
                return;
        }
        printf("contents of Stack are\n");
        for(i=top;i>=0;i--)
        {
                printf("%d\n",stack[i]);
        }
}
```

OUTPUT

```
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 1
Enter the item to be inserted: 10
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 1
Enter the item to be inserted: 20
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 1
Enter the item to be inserted: 56
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 5
contents of Stack are
56
20
10
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 2
The item deleted is: 56
*************************MENU*********************************
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**************************************************************
Enter your choice: 4
*********The status of STACK********
Total number of elements can be inserted into stack: 5
The total number of elements present in stack is:  2
Number of locations free in Stack: 3
```

```
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 3
Demonstration of how stack can be used to check palindrome
Make first stack empty
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 2
The item deleted is: 20
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 2
The item deleted is: 10
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 2
STACK underflow
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 3
Demonstration of how stack can be used to check palindrome
Enter the element: 121

 The Given Number is palindrome
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 3
Demonstration of how stack can be used to check palindrome
Enter the element: 1234

 Given Number is Not a palindrome
**********MENU**********
1: PUSH 2: POP 3:Palindrome work 4: Status of STACK 5: Dispaly 6:exit
**********
Enter your choice: 6
```

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.

Algorithm

```
opstk = the empty stack;
while(not end of input)
{
        symb=next input character;
        if(symb is an operand)
                add symb to the postfix string;
        else
           {
                while(!empty(opstk) && prcd(stacktop(opstk),symb))
                {
                        topsymb=pop(opstk);
                        add topsymb to the postfix string;
                }
                push(opstk,symb);

            }
}
while(!empty(opstk))
{
        topsymb= pop(opstk);
        add topsymb to the postfix string;
}
```

Program

```
#include<stdio.h>
#include<string.h>
char infix[50],postfix[50];
char s[10];
int top=-1;
void main()
{
        void push(char);
        char pop();
        void convert();
        int pri(char ch);
        printf("enter the infix expression\n");
        scanf("%s",infix);
        convert();

 }
 void push(char ch)
{
        s[++top]=ch;
}
```

```
char pop()
{
        return(s[top--]);
 }

 int pri(char ch)
 {
        if(ch=='(' || ch=='#')
                return 1;

        if(ch=='+' || ch=='-')
                return 2;

        if(ch=='*' || ch=='/' || ch=='%')
                return 3;

        if(ch=='^')
                return 4;
  return 0;
}


void convert()
{
         int i,j=0;
         push('#');
        for(i=0;infix[i]!='\0';i++)
        {
                if(isalnum(infix[i]))
                        postfix[j++]=infix[i];

                else if(infix[i]=='(')
                        push(infix[i]);

                else if(infix[i]==')')
                {
                        while(s[top]!='(')
                                postfix[j++]=pop();
                        pop();
                }
                else
                {
                        while(pri(s[top])>=pri(infix[i]))
                                postfix[j++]=pop();
                        push(infix[i]);
                }
         }
        while(s[top]!='#')
                postfix[j++]=pop();
        postfix[j]='\0';
        printf("postfix expression====> %s",postfix);
}
```

OUTPUT 1:

enter the infix expression
a+b*c
postfix expression====> abc*+

OUTPUT 2:

enter the infix expression
(a+b)-(c*d)
postfix expression====> ab+cd*-

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: +, -, *, /, %, ^.

Algorithm

```
opndstk=the empty stack;
while(not end of input)
{
        symb=next input character;

        if(symb is an operand)
                push(opndstk,symb);
        else
        {
                opnd2=pop(opndstk);
                opnd1=pop(opndstk);
                value=result of applying symb to opnd1 and opnd2;
                push(opndstk,value);
        }
}
return(pop(opndstk));
```

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define TRUE 1
#define FALSE 0
int top=-1;
double stack[25];
char postfix[25];
void main()
{
        double eval();
        double result;
        printf("enter a valid postfix expression");
        scanf("%s",postfix);
        result=eval();
        printf("\n the result is %f",result);
}
double eval()
{
        void push(double);
        double pop();
        int is_digit(char);
        double oper(char,double,double);
        double opnd1,opnd2,value;
        int i=0;
        while(postfix[i]!='\0')
        {
                if(isdigit(postfix[i]))
```

```
                        push(postfix[i]-'0');
                else
                {
                        opnd2=pop();
                        opnd1=pop();
                        value=oper(postfix[i],opnd1,opnd2);
                        push(value);
                }
                i++;
        }
  return(pop());
}
 double oper(char opr,double opnd1,double opnd2)
 {
        double res=1;
        switch(opr)
        {
                case '+':return(opnd1+opnd2);
                case '-':return(opnd1-opnd2);
                case '*':return(opnd1*opnd2);
                case '/':return(opnd1/opnd2);
                case '^':return(pow(opnd1,opnd2));
                case '%':return((int)opnd1%(int)opnd2);
                default:printf("illegal operation");
                exit(1);
        }
 }
void push(double item)
 {
        stack[++top]=item;
 }

 double pop()
 {
        double item;
        item=stack[top--];
        return(item);
 }

 int is_digit(char symb)
 {
        if(symb>='0' && symb<='9')
                return(TRUE);
        else
                return(FALSE);
 }
```

OUTPUT

enter a valid postfix expression : 23+
the result is 5

enter a valid postfix expression : 23+4*
the result is 20

b. Solving Tower of Hanoi problem with n disks.

```
#include <stdio.h>
void towers(int, char, char, char);
int 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, 'A', 'C', 'B');
        printf("\n\n");
        return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
        if (num == 1)
        {
                printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
                return;
        }
        towers(num - 1, frompeg, auxpeg, topeg);

        printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
        towers(num - 1, auxpeg, topeg, frompeg);
}
```

OUTPUT 1

```
Enter the number of disks : 1
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from peg A to peg C
```

OUTPUT 2:

```
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
```

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.

Algorithm for INSERT Operation

Step 1: **if** count is equal to **Queue Size then**

Write "Queue overflow"

Step 2: rear ← (rear + 1) % QSIZE;
Step 3: queue [rear] ← item;
Step 4: count ← count + 1;
Step 5: Stop

Algorithm for DELETE Operation

Step 1: **if** count is equal to 0 **then**

Write "Queue Underflow"

Step 2: **Print** Element
Step 3: front ← (front + 1) % QSIZE
Step 4: count ← count - 1
Step 5: Stop

```
#include<stdio.h>
#define QSIZE 5
#include<stdlib.h>
#include<string.h>
int choice,rear,front,count=0;
char queue[10],item;
void insert();
void del();
void display();
void main()
{
        front=0,rear=-1;
        count=0;
        for(;;)
        {
                printf("\n****************MENU*******************\n");
                printf("\n1: insert 2: delete 3: display 4: exit\n");
                printf("\n****************************************\n");
                printf("\nEnter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
```

```
                        case 1: printf("Enter the item to be inserted: ");
                                scanf("\n%c",&item);
                                insert();
                                break;
                        case 2: del();
                                break;
                        case 3: display();
                                break;
                        default: exit(0);
                }
        }
}
void insert()
{
        if(count==QSIZE)
        {
                printf("queue over flow\n");
                return;
        }
        rear=(rear+1)%QSIZE;
        queue[rear]=item;
        count+=1;
}

void del()
{
        if(count==0)
        {
                printf("queue underflow\n");
                return;
        }
        printf("%c is deleted\n",queue[front]);
        front=(front+1)%QSIZE;
        count - = 1;
}

void display()
{
        int i,m;
        m=front;
        if(count==0)
        {
                printf("queue is empty\n");
                return;
        }
        printf("contents of queue\n");
        for(i=1;i<=count;i++)
        {
                printf("%c\t",queue[m]);
                m=(m+1)%QSIZE;
        }
}
```

OUTPUT

```
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 1
Enter the item to be inserted: c
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 1
Enter the item to be inserted: s
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 1
Enter the item to be inserted: e
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 3
contents of queue
c       s       e
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 2
c is deleted
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 2
s is deleted
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 2
e is deleted
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 2
queue underflow
**********MENU**********
1: insert 2: delete 3: display 4: exit
****************************************
Enter your choice: 4
```

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.**

ALGORITHM:

Step 1: Start.

Step 2: Read the value of N. (N student's information)

Step 2: Create a singly linked list (SLL) for N students by front insertion.

Step 3: Display the status of SLL.

Step 4: Count the number of nodes.

Step 5: Perform insertion at front of list.

Step 6: Perform deletion at the front of the list.

Step 7: Perform insertion at end of the list.

Step 8: Perform deletion at the end of the list.

Step 9: Demonstrate how singly linked list can be used as stack.

Step 10: Demonstrate how singly linked list can be used as queue.

Step 11: Stop.


```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct node
{
        int sem;
        long long int phno;
        char name[20],branch[10],usn[20];
        struct node *next;
};
struct node *first=NULL,*last=NULL,*temp=NULL,*prev=NULL;
void readdata()
{
        int sem;
        long long int 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 %lld", 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 create()
{
        if (first == NULL)
        {
                first = temp;
        }
        else
        {
                temp->next = first;
                first = temp;
        }
}

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

void insert_atend()
{
        if(first==NULL)
        {
                first = temp;
        }
        else
        {
                last=first;
                while(last->next!=NULL)
                {
                        last=last->next;
                }
                last->next=temp;
        }
}
```

```
void delete_atend()
{
        last=first;
        if(first==NULL)
        {
                printf("LIST IS EMPTY\n");
                return;
        }
        else if(last->next==NULL)
        {
                printf("%s %s %s %d %lld\n", last->usn, last->name,last->branch,last->sem, last->phno );
                free(last);
                first=NULL;
        }
        else
        {
                while(last->next!=NULL)
                {
                        prev=last;
                        last=last->next;
                }
                printf("%s %s %s %d %lld\n", last->usn, last->name,last->branch,last->sem, last->phno );
                free(last);
                prev->next=NULL;
        }
        count--;
}

void insert_atfront()
{
        if (first == NULL)
        {
                first = temp;
        }
        else
        {
                temp->next = first;
                first = temp;
        }
}
void delete_atfront()
{
        temp=first;
        if(first==NULL)
        {
                printf("LIST IS EMPTY\n");
                return;
        }
        else if(temp->next==NULL)
        {
             printf("%s %s %s %d %lld", temp->usn, temp->name,temp->branch,temp->sem, temp->phno );
             free(temp);
             first=NULL;
        }
```

```
        else
        {
            printf("%s %s %s %d %lld", temp->usn, temp->name,temp->branch,temp->sem, temp->phno );
            first=temp->next;
            free(temp);
        }
        count--;
}

void main()
{
        int choice,n,i;
        while (1)
        {
                printf("-----------------MENU----------------------\n");
                printf("\n 1   To Create a SLL of N Students Data by using front insertion");
                printf("\n 2 - Display the status of SLL and count the number of nodes in it");
                printf("\n 3 - Insertion at End of SLL");
                printf("\n 4 - Deletion at End of SLL");
                printf("\n 5 - Insertion at Front of SLL(Demonstration of stack)");
                printf("\n 6 - Deletion at Front of SLL(Demonstration of stack)");
                printf("\n 7 - exit\n");
                printf("-------------------------------------------\n");
                printf("\n Enter choice : ");
                scanf("%d", &choice);
                switch (choice)
                {
                        case 1: printf("\n Enter no of students : ");
                                scanf("%d", &n);
                                for(i=0;i<n;i++)
                                {
                                        readdata();
                                        create();
                                }
                                break;
                        case 2: display();
                                break;
                        case 3: readdata();
                                insert_atend();
                                break;
                        case 4: delete_atend();
                                break;
                        case 5: readdata();
                                insert_atfront();
                                break;
                        case 6: delete_atfront();
                                break;
                        case 7: exit(0);
                        default:  printf("wrong choice\n");
                }
        }
}
```

OUTPUT

```
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
-----------------------------------------
 Enter choice : 1
 Enter no of students : 2
 Enter usn,name, branch, sem, phno of student : 1ck15cs001 sowmya cse 3 9446545789
 Enter usn,name, branch, sem, phno of student : 1ckcs15005 Deepthi cse 3 8876547869
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
-----------------------------------------
 Enter choice : 3
 Enter usn,name, branch, sem, phno of student : 1ckcs15043 Divya cse 3 9900675879
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
-----------------------------------------
 Enter choice : 2
 Linked list elements from begining :

 1ckcs15005 Deepthi cse 3 8876547869
 1ck15cs001 sowmya cse 3 9446545789
 1ckcs15043 Divya cse 3 9900675879 No of students = 3
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
-----------------------------------------
 Enter choice : 4
1ckcs15043 Divya cse 3 9900675879
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
```

```
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
----------------------------------------
 Enter choice : 5
 Enter usn,name, branch, sem, phno of student : 1ck15cs005 Manjunath cse 3 9987765897
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
----------------------------------------
 Enter choice : 2
 Linked list elements from begining :

 1ck15cs005 Manjunath cse 3 9987765897
 1ckcs15005 Deepthi cse 3 8876547869
 1ck15cs001 sowmya cse 3 9446545789 No of students = 3
 ----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
----------------------------------------
 Enter choice : 6
1ck15cs005 Manjunath cse 3 998776589
----------MENU----------
 1   To Create a SLL of N Students Data by using front insertion
 2 - Display the status of SLL and count the number of nodes in it
 3 - Insertion at End of SLL
 4 - Deletion at End of SLL
 5 - Insertion at Front of SLL(Demonstration of stack)
 6 - Deletion at Front of SLL(Demonstration of stack)
 7 - exit
----------------------------------------
 Enter choice : 7
```

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.**

ALGORITHM:

Step 1: Start.

Step 2: Read the value of N. (N Employee's information)

Step 3: Create a doubly linked list for N Employee's by End insertion. (DLL)

Step 4: Display the status of DLL.

Step 5: Count the number of nodes.

Step 6: Perform insertion at front of list.

Step 7: Perform deletion at the front of the list.

Step 8: Perform insertion at end of the list.

Step 9: Perform deletion at the end of the list.

Step 10: Demonstrate how doubly linked list can be used as double ended queue.

Step 11: Stop.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct dlist
{
        struct dlist *llink;
        char ssn[10],name[20],dept[10],designation[10];
        int sal;
        long long int phno;
        struct dlist *rlink;
};
typedef struct dlist dnode;
dnode *first,*temp,*prev,*last;
int count=0;
void readdata();
void create();
void insert_atfront();
void insert_atend();
void delete_atfront();
void delete_atend();
void display();
void double_ended_queue();
```

```
void main()
{
        int choice,item,key,n,i;
        first=NULL;
        while(1)
        {
                printf("\n**********************MENU*********************************");
                printf("\n Choice 1 to Create a DLL of N Employees Data by using end insertion.");
                printf("\n Choice 2 to Display the status of DLL and count the number of nodes in it");
                printf("\n Choice 3 to Perform Insertion at End of DLL");
                printf("\n Choice 4 to Perform Deletion at End of DLL ");
                printf("\n Choice 5 to Perform Insertion at Front of DLL");
                printf("\n Choice 6 to Perform Deletion at Front of DLL");
                printf("\n Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue");
                printf("\n Choice 8 to exit");
                printf("\n*****************************************************************");
                printf("\n enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("\n Enter no of Employees data to read : ");
                                scanf("%d", &n);
                                for(i=0;i<n;i++)
                                {
                                        readdata();
                                        create();
                                }
                                break;
                        case 2: display();
                                break;
                        case 3:  readdata();
                                insert_atend();
                                break;
                        case 4: delete_atend();
                                break;
                        case 5: readdata();
                                insert_atfront();
                                break;
                        case 6:delete_atfront();
                                break;
                        case 7:double_ended_queue();
                                break;
                        case 8:exit(0);
                }
        }
}

void readdata()
{
        char ssn[10],name[20],dept[10],designation[10];
        int sal;
        long long int phno;
        temp=(struct dlist*)malloc(sizeof(struct dlist));
        printf("\n Enter SSN ,Name, Dept, Designation, Sal & Phno of a Employee: ");
```

```
        scanf("%s%s%s%s%d%lld",ssn,name,dept,designation,&sal,&phno);
        strcpy(temp->ssn,ssn);
        strcpy(temp->name,name);
        strcpy(temp->dept,dept);
        strcpy(temp->designation,designation);
        temp->sal = sal;
        temp->phno=phno;
        temp->llink = temp->rlink = NULL;
        count++;
}
void create()
{

        insert_atend();
}
void insert_atend()
{
        if(first==NULL)
                first=temp;
        else
        {
                last=first;
                while(last->rlink!=NULL)
                {
                        last=last->rlink;
                }
                last->rlink=temp;
                temp->llink=last;
        }
}
void insert_atfront()
{
        if(first==NULL)
                first=temp;
        else
        {
                temp->rlink=first;
                first->llink=temp;
                first=temp;
        }
}

void delete_atfront()
{
        temp=first;
        if(first==NULL)
        {
                printf("LIST IS EMPTY\n");
                return;
        }
        else if(temp->rlink==NULL)
        {
                printf("%s %s %s %s %d %lld", temp->ssn, temp->name, temp->dept, temp->designation,
                                                                        temp->sal, temp->phno);
```

```
                free(temp);
                first=NULL;
        }
        else
        {
                printf("%s %s %s %s %d %lld", temp->ssn, temp->name, temp->dept, temp->designation,
                                                                        temp->sal, temp->phno);
                first=first->rlink;
                free(temp);
        }
        count--;
}

void delete_atend()
{
        temp=first;
        if(first==NULL)
        {
                printf("LIST IS EMPTY\n");
                return;
        }
        else if(temp->rlink==NULL)
        {
                printf("%s %s %s %s %d %lld", temp->ssn, temp->name, temp->dept, temp->designation,
                                                                        temp->sal,temp->phno);
                free(temp);
                first=NULL;
        }
        else
        {
                while(temp->rlink!=NULL)
                {
                        prev=temp;
                        temp=temp->rlink;
                }
                printf("%s %s %s %s %d %lld", temp->ssn, temp->name, temp->dept, temp->designation,
                                                                        temp->sal, temp->phno);
                free(temp);
                prev->rlink=NULL;
        }
        count--;
}
void display()
{
        temp=first;
        if(temp==NULL)
                printf("LIST IS EMPTY\n");
        else
          {
                printf("The Employee Data is\n");
                 while(temp)
                {
                        printf("%s\t %s\t %s\t %s\t %d\t %lld\n",temp->ssn,temp->name,temp->dept,
                                                        temp->designation,temp->sal,temp->phno);
```

```
                        temp=temp->rlink;
                }
        }
}
void double_ended_queue()
{
        printf("\n Demonstration of DLL as Double Ended Queue i,e both ends you can insert and delete");
        char option;
        while(1)
        {
                printf("\n************Double Ended Queue Operation**************");
                printf("\n Option A-> Perform Insertion at Rear End of DLL");
                printf("\n Option B-> Perform Deletion at Rear End of DLL ");
                printf("\n Option C-> Perform Insertion at Front End of DLL");
                printf("\n Option D-> Perform Deletion at Front End of DLL");
                printf("\n Option E-> Display elements in Double Ended Queue");
                printf("\n Option F-> To return to main program");
                printf("\n*****************************************************");
                printf("\nEnter your option: ");
                scanf("\n%c",&option);
                switch(option)
                {
                        case 'A':readdata();
                                 insert_atend();
                                 break;
                        case 'B':delete_atend();
                                 break;
                        case 'C':readdata();
                                 insert_atfront();
                                 break;
                        case 'D':delete_atfront();
                                 break;
                        case 'E':display();
                                 break;
                        case 'F':return;
                }
        }
}
```

OUTPUT

```
**********MENU**********
Choice 1 to Create a DLL of N Employees Data by using end insertion.
Choice 2 to Display the status of DLL and count the number ofnodesin it
Choice 3 to Perform Insertion at End of DLL
Choice 4 to Perform Deletion at End of DLL
Choice 5 to Perform Insertion at Front of DLL
Choice 6 to Perform Deletion at Front of DLL
Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
Choice 8 to exit
*****************************************************************
enter your choice: 1
Enter no of Employees data to read : 2
Enter SSN ,Name, Dept, Designation, Sal & Phno of a Employee: 1
Sona cse Engineer 30000 9900342567
Enter SSN ,Name, Dept, Designation, Sal & Phno of a Employee: 2
```

```
Deepa cse Engineer 40000 8722345436
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 2
The Employee Data is
1       Sona    cse  Engineer      30000  9900342567
2       Deepa   cse  Engineer      40000  8722345436
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 3
 Enter SSN ,Name, Dept, Designation, Sal & Phno of a Employee: 3
Hamsa cse Engineer 50000 9988765897
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 2
The Employee Data is
1       Sona    cse  Engineer      30000  9900342567
2       Deepa   cse  Engineer      40000  8722345436
3       Hamsa   cse  Engineer      50000  9988765897
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 4
```

```
3 Hamsa cse Engineer 50000 9988765897
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 2
The Employee Data is
1       Sona      cse  Engineer       30000  9900342567
2       Deepa     cse  Engineer       40000  8722345436
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 5
 Enter SSN ,Name, Dept, Designation, Sal & Phno of a Employee: 4
Seema cse Engineer 45000  8745298765
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice: 2
The Employee Data is
4       Seema     cse  Engineer       45000  8745298765
1       Sona      cse  Engineer       30000  9900342567
2       Deepa     cse  Engineer       40000  8722345436
 **********MENU**********
 Choice 1 to Create a DLL of N Employees Data by using end insertion.
 Choice 2 to Display the status of DLL and count the number nodes in it
 Choice 3 to Perform Insertion at End of DLL
 Choice 4 to Perform Deletion at End of DLL
 Choice 5 to Perform Insertion at Front of DLL
 Choice 6 to Perform Deletion at Front of DLL
 Choice 7 to Demonstrate how this DLL can be used as Double Ended Queue
 Choice 8 to exit
**********
 enter your choice:7
```

9. Design, Develop and Implement a Program in C for the following operations on 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.

ALGORITHM:

Step 1: Start.

Step 2: Read a polynomial poly(x,y,z) of N terms.

Step 3: Represent the polynomial using singly circular linked list with header Nodes.

Step 4: Evaluate the given polynomial.

value = eval(p(x,y,z));

Step 5: print "**value**".

Step 6: Read two polynomials poly1(x,y,z) and poly2(x,y,z).

Step 7: find the sum of the polynomials.

polysum(x,y,z) = poly1(x,y,z) + poly2(x,y,z).

Step 8: print "polysum(x,y,z)".

Step 9: Stop.

```
#include<stdio.h>
#include<malloc.h>
#include<math.h>
#include<stdlib.h>
struct polynode
{
        float coef;
        int xexpo,yexpo,zexpo;
        int status;
        char sign;
        struct polynode *link;
};

struct polynode *polysum(struct polynode *,struct polynode *);
struct polynode *readdata(struct polynode *,struct polynode *,int);
struct polynode *insert(struct polynode *,struct polynode *,char,float,int,int,int);
float evaluate(struct polynode *,int);
void display(struct polynode *);

int p,q; //variables to strore the number of terms
struct polynode *head4,*head1,*head2,*head3,*head;

void main( )
{
        struct polynode *p_start=NULL,*p1_start=NULL,*p2_start=NULL,*p3_start=NULL;
        //pointer to each list
        float value;
        int choice;
```

```
        head=(struct polynode *)malloc(sizeof(struct polynode));
        head1=(struct polynode *)malloc(sizeof(struct polynode));
        head2=(struct polynode *)malloc(sizeof(struct polynode));
        head3=(struct polynode *)malloc(sizeof(struct polynode));
        head4=(struct polynode *)malloc(sizeof(struct polynode));
        head->link=head1->link=head2->link=head3->link=head4->link=NULL;
        while(1)
        {
                printf("\n **********MENU**********")
                printf("\n Choice 1 to Represent and Evaluate a Polynomial P(x,y,z).");
                printf("\n Choice 2 to Find the sum of two polynomials");
                printf("\n Choice 3 to exit");
                printf("\n**********");
                printf("\n enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("Enter the No of terms for Polynomial:  ");
                                scanf("%d",&p);
                                p_start=readdata(p_start,head4,p);
                                printf("\n Polynomial is :\n");
                                display(head4);
                                value=evaluate(head4,p);
                                printf("After evaluation poly(x,y,z)= %f",value);
                                break;

                        case 2: printf("Enter the No of terms for Polynomial 1:  ");
                                scanf("%d",&p);
                                printf("Polynomial 1 :\n");
                                        p1_start=readdata(p1_start,head1,p);
                                printf("\nEnter the No of terms for Polynomial 2:  ");
                                scanf("%d",&q);
                                printf("Polynomial 2 : \n");
                                        p2_start=readdata(p2_start,head2,q);
                                printf("\n Polynomial 1 is :  ");
                                        display(head1);
                                printf("\n Polynomial 2 is :  ");
                                        display(head2);
                                p3_start=polysum(head1,head2);
                                printf("\n Added polynomial is :  ");
                                display(head3);
                                break;

                        case 3:exit(0);
                }
        }
}
struct polynode *readdata(struct polynode *start,struct polynode *head,int n)
{
        int i,xexponent,yexponent,zexponent;
        char sign;
        float coeficient;
        for(i=1;i<=n;i++)
        {
```

```
                printf("Enter sign, coeficient, x, y & z exponent for term %d : ",i);
                scanf("\n%c%f%d%d%d",&sign,&coeficient,&xexponent,&yexponent,&zexponent);
                start=insert(start,head,sign,coeficient,xexponent,yexponent,zexponent);
        }
   return start;
}

struct polynode *insert(struct polynode *start,struct polynode *head,char sign,float co,int xex,int yex,int zex)
{
        struct polynode *temp;
        temp=(struct polynode *)malloc(sizeof(struct polynode));
        temp->coef=co;
        temp->xexpo=xex;
        temp->yexpo=yex;
        temp->zexpo=zex;
        temp->status=1;
        temp->sign=sign;
        temp->link=NULL;
        if(start==NULL)
        {
                start=temp;
                temp->link=head;
                head->link=temp;
        }
      else{
                temp->link=start->link;
                start->link=temp;
                start=temp;
           }
 return start;
}

float evaluate(struct polynode *head4,int n)
{
        int x,y,z,i,sum=0;
        struct polynode *temp;
        temp=head4->link;
        printf("\nEnter the value of x, y, z: ");
        scanf("%d%d%d",&x,&y,&z);
        while(temp!=head4)
        {
              if(temp->sign=='+')
                 sum=sum+(temp->coef*pow(x,temp->xexpo)*pow(y,temp->yexpo)*pow(z,temp->zexpo));
              else
                 sum=sum-(temp->coef*pow(x,temp->xexpo)*pow(y,temp->yexpo)*pow(z,temp->zexpo));
                 temp=temp->link;
        }
return sum;
}

struct polynode *polysum(struct polynode *head1,struct polynode *head2)
{
        struct polynode *p3_start,*p1,*p2;
        p3_start=NULL;
```

```
        p1=head1->link;
        while(p1!=head1)
        {
              p2=head2->link;
              while(p2!=head2)
              {
                    if(p1->xexpo == p2->xexpo && p1->yexpo == p2->yexpo && p1->zexpo == p2->zexpo)
                   {
                          if(p1->sign==p2->sign)
                          {
                                p3_start=insert(p3_start,head3,p1->sign,(p1->coef+p2->coef),p1->xexpo,
                                                                                    p1->yexpo,p1->zexpo);
                                      p1->status=0;
                                      p2->status=0;
                                 }
                                 else{
                            p3_start=insert(p3_start,head3,p1->sign,(p1->coef-p2->coef)
                                                                                    ,p1->xexpo,p1->yexpo,p1->zexpo);
                             p1->status=0;
                             p2->status=0;
                                       }
                          }
                          p2=p2->link;
                    }
                    p1=p1->link;
        }

    p1=head1->link;
    while(p1!=head1)
    {
          if(p1->status==1)
          {
                p3_start=insert(p3_start,head3,p1->sign,p1->coef,p1->xexpo,p1->yexpo,p1->zexpo);
          }
          p1=p1->link;
    }
    p2=head2->link;
    while(p2!=head2)
    {
          if(p2->status==1)
          {
                p3_start=insert(p3_start,head3,p2->sign,p2->coef,p2->xexpo,p2->yexpo,p2->zexpo);
          }
          p2=p2->link;
    }
 return p3_start;
}

 void display(struct polynode *head)
 {
        struct polynode *temp;
        if(head->link==NULL)
        {
                printf("Empty\n");
```

```
                return;
        }
        temp=head->link;
        while(temp!=head)
        {
                printf(" %c %.1f x^%dy^%dz^%d  ",temp->sign,temp->coef,temp->xexpo,temp->yexpo,
                                                                                temp->zexpo);
                temp=temp->link;
        }

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

OUTPUT

```
**********MENU**********
 Choice 1 to Represent and Evaluate a Polynomial P(x,y,z).
 Choice 2 to Find the sum of two polynomials
 Choice 3 to exit
*****************************************************************
 enter your choice: 1
Enter the No of terms for Polynomial:  5
Enter sign, coeficient, x, y & z exponent for term 1 : + 6 2 2 1
Enter sign, coeficient, x, y & z exponent for term 2 : - 4 0 1 5
Enter sign, coeficient, x, y & z exponent for term 3 : + 3 3 1 1
Enter sign, coeficient, x, y & z exponent for term 4 : + 2 1 5 1
Enter sign, coeficient, x, y & z exponent for term 5 : - 2 1 1 3
 Polynomial is :
 + 6.0 x^2y^2z^1   - 4.0 x^0y^1z^5   + 3.0 x^3y^1z^1   + 2.0 x^1y^5z^1   - 2.0 x^1y^1z^3

Enter the value of x, y, z: 1 2 3
After evaluation poly(x,y,z)= -1770.000000
 **********MENU**********
 Choice 1 to Represent and Evaluate a Polynomial P(x,y,z).
 Choice 2 to Find the sum of two polynomials
 Choice 3 to exit
*****************************************************************
 enter your choice: 2
Enter the No of terms for Polynomial 1:  2
Polynomial 1 :
Enter sign, coeficient, x, y & z exponent for term 1 : + 4 3 2 1
Enter sign, coeficient, x, y & z exponent for term 2 : - 6 2 1 5

Enter the No of terms for Polynomial 2:  3
Polynomial 2 :
Enter sign, coeficient, x, y & z exponent for term 1 : + 4 3 2 1
Enter sign, coeficient, x, y & z exponent for term 2 : - 5 3 1 1
Enter sign, coeficient, x, y & z exponent for term 3 : + 7 8 5 2

 Polynomial 1 is :   + 4.0 x^3y^2z^1   - 6.0 x^2y^1z^5

 Polynomial 2 is :   + 4.0 x^3y^2z^1   - 5.0 x^3y^1z^1   + 7.0 x^8y^5z^2

 Added polynomial is :   + 8.0 x^3y^2z^1   - 6.0 x^2y^1z^5   - 5.0 x^3y^1z^1   + 7.0 x^8y^5z^2
```

**********MENU**********
Choice 1 to Represent and Evaluate a Polynomial P(x,y,z).
Choice 2 to Find the sum of two polynomials
Choice 3 to exit
****

enter your choice: 3

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**
d. **Exit.**

Algorithm for BST(Binary Search Tree)

```
Tree = maktree(number);
while(there are numbers left in the input)
{
        number = read the next number;
        p=q=tree;
        while(number != info(p) && q != NULL)
        {
                p=q;
                if(number < info(p))
                        q=left(p);
                else
                        q=right(p);
        }

        if(number < info(p))
                setleft(p, number);
        else
                setright(p, number);
}
```

Algorithms for Tree Traversals

Preorder

1. Visit the Root.
2. Traverse the left subtree in preorder.
3. Traverse the right subtree in preorder.

Inorder

1. Traverse the left subtree in inorder.
2. Visit the Root.
3. 3.Traverse the right subtree in inorder.

Postorder

1. Traverse the left subtree in postorder.
2. Traverse the right subtree in postorder.
3. Visit the Root.

```
#include<stdio.h>
#include<stdlib.h>
struct btree
{
        struct btree *left;
        int info;
        struct btree *right;
};
typedef struct btree node;
node *root;
void main()
{
        int choice,item,key;
        void create(int);
        void preorder(node *);
        void inorder(node *);
        void postorder(node *);
        void search(node *,int);
        root=NULL;
        while(1)
        {
            printf("\n**********MENU*********");
            printf("\n 1-> create\n 2-> preorder \n 3-> inorder \n 4-> postorder \n 5-> Search \n 6-> exit\n");
            printf("*************************");
                printf("\n enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("\n enter the elements of the tree to insert(-1 to terminate):");
                                scanf("%d",&item);
                                while(item!=-1)
                                {
                                        create(item);
                                        scanf("%d",&item);
                                }
                                break;

                        case 2:printf("\npreorder traversal of binary search tree is");
                                preorder(root);
                                break;

                        case 3:printf("\n preorder traversal of binary search tree is");
                                inorder(root);
                                break;

                        case 4:printf("\n preorder traversal of binary search tree is");
                                postorder(root);
                                break;

                        case 5:printf("\nenter the key element to search:");
                                scanf("%d",&key);
                                search(root,key);
                                break;
```

```
                        case 6:exit(0);
                                break;
                }
        }
}
void create(int item)
{
        node *temp,*p,*q;
        temp=(node *)malloc(sizeof(node));
        temp->info=item;
        temp->left=temp->right=NULL;
        if(root==NULL)
                root=temp;
        else
        {
                q=root;
                while(q!=NULL)
                {
                        p=q;
                        if(item<q->info)
                                q=q->left;
                        else
                                q=q->right;
                }
                if(item<p->info)
                        p->left=temp;
                else
                        p->right=temp;
        }
 }
void preorder(node *tree)
{
        if(tree!=NULL)
        {
                printf("%d\n",tree->info);
                preorder(tree->left);
                preorder(tree->right);
        }
}
void inorder(node *tree)
{
        if(tree!=NULL)
        {
                inorder(tree->left);
                printf("%d\n",tree->info);
                inorder(tree->right);
        }
}
void postorder(node *tree)
{
        if(tree!=NULL)
        {
                postorder(tree->left);
                postorder(tree->right);
```

```
                printf("%d\n",tree->info);
        }
}
void search(node *tree, int key)
{
        while(tree)
        {
                if(tree->info==key)
                {
                        printf("\nElement is found in Tree: %d",tree->info);
                        return;
                }
                else if(key < tree->info)

                        tree=tree->left;
                else
                        tree=tree->right;
        }
  printf("\nElement Not found in Tree");
}
```

OUTPUT

```
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 1
 enter the elements of the tree to insert(-1 to terminate):14 15 4 9 7 18 3 5 16 4 20 17 9 5 -1
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 2
preorder traversal of binary search tree is
14      4       3       9       7       5       4       5       9       15      18      16      17
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 3
 preorder traversal of binary search tree is
3       4       4       5       5       7       9       9       14      15      16      17      18
```

```
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 4
 preorder traversal of binary search tree is
3       4       5       5       7       9       9       4       17      16      20      18      15      14
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 5
enter the key element to search: 7
Element is found in Tree: 7
**********MENU*********
 1-> create
 2-> preorder
 3-> inorder
 4-> postorder
 5-> Search
 6-> exit
************************
 enter your choice: 6
```

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.

ALGORITHM:

BFS (G)

//Input: Graph G = < V , E >

//Output: Graph G, with its vertices marked with consecutive integers in

//the order they have been visited by the BFS traversal.

Step 1: mark each vertex in V with 0 as a mark of being "unvisited"

Step 2: count = 0

Step 3: for each vertex v in V do

Step 4: if v is marked with 0 then BFS (v)

//end for

BFS (v)

//visits recursively all the unvisited vertices connected to vertex v and

//assigns them the numbers in that order they are visited via global //variable count

Step 5: count = count + 1

Step 6: mark v with count and initialize a queue with v

Step 7: While the queue is not empty do {

Step 8: for each vertex w in V adjacent to the front's vertex v do {

Step 9: if w is marked with 0 then {

Step 10:count = count + 1

Step 11:add w to the queue

} // end if //end for

Step 12:remove vertex v from the front of the queue

} // end while

```
#include<stdio.h>
#include<stdlib.h>
int a[10][10], n, m, i, j, source, s[10], b[10];
int visited[10];
void create()
{
        printf("\nEnter the number of vertices of the digraph: ");
        scanf("%d", &n);
        printf("\nEnter the adjacency matrix of the graph:\n");
        for(i=1; i<=n; i++)
                for(j=1; j<=n; j++)
                        scanf("%d", &a[i][j]);
}
```

```
void bfs()
{
	int q[10], u, front=0, rear=-1;
	printf("\nEnter the source vertex to find other nodes reachable or not: ");
	scanf("%d", &source);
	q[++rear] = source;
	visited[source] = 1;
	printf("\nThe reachable vertices are: ");
	while(front<=rear)
	{
		u = q[front++];
		for(i=1; i<=n; i++)
		{
			if(a[u][i] == 1 && visited[i] == 0)
			{
				q[++rear] = i;
				visited[i] = 1;
				printf("\n%d", i);
			}
		}
	}
}
void dfs(int source)
{
	int v, top = -1;
	s[++top] = 1;
	b[source] = 1;
	for(v=1; v<=n; v++)
	{
		if(a[source][v] == 1 && b[v] == 0)
		{
			printf("\n%d -> %d", source, v);
			dfs(v);
		}
	}
}
void main()
{
	int ch;
	while(1)
	{
		printf("\n1.Create Graph\n2.BFS\n3.Check graph connected or not(DFS)\n4.Exit");
		printf("\nEnter your choice: ");
		scanf("%d", &ch);
		switch(ch)
		{
			case 1: create();
				break;
			case 2: bfs();
				for(i=1;i<=n;i++)
					if(visited[i]==0)
				printf("\nThe vertex that is not rechable %d" ,i);
				break;
			case 3: printf("\nEnter the source vertex to find the connectivity: ");
```

```
                                scanf("%d",&source);
                                m=1;
                                dfs(source);
                                for(i=1;i<=n;i++)
                                {
                                        if(b[i]==0)
                                                m=0;
                                }
                                if(m==1)
                                        printf("\nGraph is Connected");
                                else
                                        printf("\nGraph is not Connected");
                                break;
                        default: exit(0);
                }
        }
}
```

OUTPUT

```
1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 1

Enter the number of vertices of the digraph: 3

Enter the adjacency matrix of the graph:
0 1 0
1 0 0
0 0 0

1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 2

Enter the source vertex to find other nodes reachable or not: 1

The reachable vertices are:
2
The vertex that is not rechable 3
1.Create Graph
2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 3

Enter the source vertex to find the connectivity: 1

1 -> 2
Graph is not Connected
1.Create Graph
```

2.BFS
3.Check graph connected or not(DFS)
4.Exit
Enter your choice: 4

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.

ALGORITHM:

Step 1: Start.

Step 2: Given N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F.

Step 3: 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.

Step 3: Let the keys in K and addresses in L are Integers

Step 4: Hash function H: K → L as H(K)=K **mod** m (remainder method)

Step 5: Hashing as to map a given key K to the address space L, Resolve the collision (if any) is using linear probing.

Step6: Stop.

```
#include<stdio.h>
#include<stdlib.h>
#define LIMIT 5
enum record_status {EMPTY, DELETED, OCCUPIED};
struct Employee
{
        int employee_id, employee_age;
        char employee_name[30];
};

struct Record
{
        struct Employee info;
        enum record_status status;
};

int hash_function(int key)
{
        return (key % LIMIT);
}

int search_records(int key, struct Record htable[])
{
        int count, temp, pos;
        temp = hash_function(key);
        pos = temp;
        for(count = 1; count != LIMIT + 1; count++)
        {
```

```
                if(htable[pos].status == EMPTY)
                {
                        return -1;
                }
                if( htable[pos].info.employee_id == key)
                {
                        return pos;
                }
                pos = (temp + count) % LIMIT;
        }
        return -1;
}
void insert_records(struct Employee emprec, struct Record htable[])
{
        int count, pos, temp,probing=0;
        int key = emprec.employee_id;
        temp = hash_function(key);
        pos = temp;
        // Collision Detection and Resolving using Linear Probing
            for(count = 1; count != LIMIT+1; count++)
        {
                if(htable[pos].status == EMPTY || htable[pos].status == DELETED)
                {
                        htable[pos].info = emprec;
                        htable[pos].status = OCCUPIED;
                        printf("\nRecord Inserted into Hash Table\n");
                        if(probing==1)
                        printf("\nCollision Detected and resolved using Linear Probing\n");
                        return;
                }
                if(htable[pos].info.employee_id == key)
                {
                        printf("\nDuplicate Record cannot be Inserted\n");
                        return;
                }
                probing=1;
        pos = (temp + count) % LIMIT;
        }
        printf("\nHash Table Limit Exceeded\n");
}

void display_records(struct Record htable[])
{
        int count;
        printf("\nHash Table\n");
        for(count = 0; count < LIMIT; count++)
        {
                printf("[%d]:\t", count);
                if(htable[count].status == OCCUPIED)
                {
                   printf("Occupied-ID: %d Name: %s Age: %d\n",htable [count].info.employee_id,
                                htable[count].info.employee_name, htable[count].info.employee_age);
                }
                else if(htable[count].status == DELETED)
```

```
                {
                        printf("\nRecord is Deleted\n");
                }
                else
                {
                        printf("\nHash Table is Empty\n");
                }
        }
}
void delete_records(int key, struct Record htable[])
{
        int pos = search_records(key, htable);
        if(pos == -1)
        {
                printf("\nKey Not Found\n");
        }
        else
        {
                htable[pos].status = DELETED;
        }
}
void main()
{
        int count, key, option;
        struct Record htable[LIMIT];
        struct Employee emprec;
        for(count = 0; count <= LIMIT - 1; count++)
        {
                htable[count].status = EMPTY;
        }
        while(1)
        {
                printf("\n1. Insert a Record\n");
                printf("2. Delete a Record\n");
                printf("3. Search a Record\n");
                printf("4. Display All Records\n");
                printf("5. Exit\n");
                printf("Enter Your Option:  ");
                scanf("%d", &option);
                switch(option)
                {
                        case 1: printf("\nEnter Employee ID: ");
                                scanf("%d", &emprec.employee_id);
                                printf("Enter Employee Name: ");
                                scanf("%s", emprec.employee_name);
                                printf("Enter Employee Age: ");
                                scanf("%d", &emprec.employee_age);
                                insert_records(emprec, htable);
                                break;

                        case 2: printf("\nEnter the Key to Delete:\t");
                                scanf("%d", &key);
                                delete_records(key, htable);
                                break;
```

```
                case 3: printf("\nEnter the Key to Search:\t");
                        scanf("%d", &key);
                        count = search_records(key, htable);
                        if(count == -1)
                        {
                                printf("\nRecord Not Found\n");
                        }
                        else
                        {
                        printf("\nRecord Found at Index Position:\t%d\n", count);
                        }
                        break;


                case 4:  display_records(htable);
                        break;

                case 5:  exit(1);
                }
        }

}
```

OUTPUT:

```
1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  1
Enter Employee ID: 1111
Enter Employee Name: anil
Enter Employee Age: 34

Record Inserted into Hash Table

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  1
Enter Employee ID: 3333
Enter Employee Name: kumar
Enter Employee Age: 25

Record Inserted into Hash Table

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  1
```

Enter Employee ID: 1201
Enter Employee Name: vinu
Enter Employee Age: 32

Record Inserted into Hash Table

Collision Detected and resolved using Linear Probing

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option: 1

Enter Employee ID: 1234
Enter Employee Name: lokesh
Enter Employee Age: 31

Record Inserted into Hash Table

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option: 4

Hash Table
[0]:
Hash Table is Empty
[1] Occupied-ID: 1111 Name: anil Age: 34
[2] Occupied-ID: 1201 Name: vinu Age: 32
[3] Occupied-ID: 3333 Name: kumar Age: 25
[4] Occupied-ID: 1234 Name: lokesh Age: 31

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option: 1

Enter Employee ID: 1111
Enter Employee Name: manu
Enter Employee Age: 35

Duplicate Record cannot be Inserted

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records

```
5. Exit
Enter Your Option:  1

Enter Employee ID: 2004
Enter Employee Name: guru
Enter Employee Age: 45

Record Inserted into Hash Table

Collision Detected and resolved using Linear Probing

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  4

Hash Table
[0]:    Occupied-ID: 2004 Name: guru Age: 45
[1]:    Occupied-ID: 1111 Name: anil Age: 34
[2]:    Occupied-ID: 1201 Name: vinu Age: 32
[3]:    Occupied-ID: 3333 Name: kumar Age: 25
[4]:    Occupied-ID: 1234 Name: lokesh Age: 31

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  1

Enter Employee ID: 1321
Enter Employee Name: vinay
Enter Employee Age: 29

Hash Table Limit Exceeded

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option:  3

Enter the Key to Search:        1234

Record Found at Index Position: 4

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
```

Enter Your Option: 2

Enter the Key to Delete: 1201

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option: 4

Hash Table
[0]: Occupied-ID: 2004 Name: guru Age: 45
[1]: Occupied-ID: 1111 Name: anil Age: 34
[2]:
Record is Deleted
[3]: Occupied-ID: 3333 Name: kumar Age: 25
[4]: Occupied-ID: 1234 Name: lokesh Age: 31

1. Insert a Record
2. Delete a Record
3. Search a Record
4. Display All Records
5. Exit
Enter Your Option: 5

VIVA QUESTIONS

1. **What is data structure?**
2. **When is a binary search best applied?**
3. **Define linked list?**
4. **How do you reference all the elements in a one-dimension array?**
5. **In what areas do data structures applied?**
6. **What is LIFO?**
7. **What is a queue?**
8. **What are binary trees?**
9. **Which data structures are applied when dealing with a recursive function?**
10. **What is a stack?**
11. **Explain Binary Search Tree**
12. **What are multidimensional arrays?**
13. **How does dynamic memory allocation help in managing data?**
14. **Explain the merge sort logic?**
15. **What is a linear search?**
16. **What is Data abstraction?**
17. **How do you insert a new item in a binary search tree?**
18. **How does a selection sort work for an array?**
19. **What is the minimum number of nodes that a binary tree can have?**
20. **What are ARRAYs?**
21. **Differentiate STACK from ARRAY.**
22. **What is a dequeue?**
23. **What is a bubble sort and how do you perform it?**
24. **Differentiate linear from non linear data structure.**
25. **What are doubly linked lists?**
26. **How do you search for a target key in a linked list?**
27. What do you mean by overflow and underflow?
28. What are the disadvantages array implementations of linked list?
29. What is a priority queue?
30. What are the disadvantages of representing a stack or queue by a linked list?
31. What is Stack and where it can be used?
32. What is a Queue, how it is different from stack?
33. Which data structures are used for BFS and DFS of a graph?
34. Explain different types of Tree Traversal.
35. What are limitations of array?
36. What is the operations of array?
37. What are the applications of stack?
38. What are the operations performed on stack?
39. What is recursion?
40. What is Circular Queue?
41. What are the different types of linked list?
42. What are the advantages of linked list?
43. What is null pointer?
44. What are advantages of circular linked list
45. What is a dangling pointer?
46. What is a tree?
47. What do you mean by degree of a tree?

48. What are the different types of sorts.
49. How does quick sort works?
50. Define the complexity of binary search?
51. What do you mean by directed acyclic graph?
52. What is a connected graph?
53. What is a weighted graph?
54. Give postfix form for (A+B)*C/D A7 AB+C*D/
55. Give postfix form for A+B/C-D A8 ABC/+D
56. Give prefix form for A/B^C+D A9 +/A^BCD
57. Give prefix form for A*B+C A10 +*ABC