Karnataka ReddyJana Sangha (R)
VEMANA INSTITUTE OF TECHNOLOGY
Approved by AICTE-New Delhi, Affiliated to VTU-Belagavi, Recognized by Govt. of Karnataka
#1,Mahayogi Vemana Road, 3rd Block, Koramangala, Bengaluru - 34.
Accredited by NBA
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Academic Year 2023 – 2024 (ODD Semester)

LABMANUAL

BCSL305 DATA STRUCTURES LABORATORY

III Semester CS

Vision and Mission of the Institute

Vision

To become a leading institute for quality technical education and research with ethical values.

Mission

M1: To continually improve quality education system that produces thinking engineers having good technical capabilities with human values.

M2: To nurture a good eco-system that encourages faculty and students to engage in meaningful research and development.

M3: To strengthen industry institute interface for promoting team work, internship and entrepreneurship.

M4: To enhance educational opportunities to the rural and weaker sections of the society to equip with practical skills to face the challenges of life.

Vision and Mission of the Department

Vision

To become a leading department engaged in quality education and research in the field of computer science and engineering.

Mission

M1: To nurture a positive environment with state of art facilities conducive for deep learning and meaningful research and development.

M2: To enhance interaction with industry for promoting collaborative research in emerging technologies.

M3: To strengthen the learning experiences enabling the students to become ethical professionals with good interpersonal skills, capable of working effectively in multi-disciplinary teams.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Successful and ethical professionals in IT and ITES industries contributing to societal progress.

PEO 2: Engaged in life-long learning, adapting to changing technological scenarios.

PEO 3: Communicate and work effectively in diverse teams and exhibit leadership qualities.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Analyze, design, implement and test innovative application software systems to meet the specified requirements.

PSO 2: Understand and use systems software packages.

PSO 3: Understand the organization and architecture of digital computers, embedded systems and computer networks.

PROGRAM OUTCOMES (POs)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental consideration.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DATA STRUCTURES LABORATORY
SEMESTER – III

Course Code	**BCSL305**	**CIE Marks**	50
Number of Contact Hours/Week	0:0:2	**SEE Marks**	50
Total Number of Lab Contact Hours	28	**Exam Hours**	03

Credits – 1

Course Learning Objectives:

This laboratory course enables 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

Descriptions (if any):

- Implement all the programs in "C " Programming Language and Linux OS.

Programs List:

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 (A integer), 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.	Develop 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.	Develop 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.	Develop 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.	Develop 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.	Develop 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.	Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Programme, 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.	Develop 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.	Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial P(x,y,z) = $6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$ 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.	Develop 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.	Develop 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. 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.

Laboratory Outcomes: The student should 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

Conduct 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)*
  c) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  d) 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

Table of contents

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 (A integer), 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.**

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
// Structure to represent a day in the calendar
struct Day
{
char *name;
int date;
char *activity;
};
// for dynamically allocating memory for name
// for dynamically allocating memory for name
// Function to allocate memory Dynamically for each variable of structure
void create(struct Day calendar[], int size)
{
for (int i = 0; i < size; i++)
{
calendar[i].name = (char *)malloc(20 * sizeof(char));
calendar[i].activity = (char *)malloc(20 * sizeof(char));
   }
}

// Function to read data from the keyboard
void read(struct Day calendar[], int size)
{
   for (int i = 0; i < size; i++)
   {
      printf("Enter details for day %d:\n", i + 1);
      printf("Enter day name: ");
      scanf("%s", calendar[i].name);
      printf("Enter date: ");
      scanf("%d", &calendar[i].date);
```

```
        printf("Enter activity: ");
        scanf(" %[^\n]s", calendar[i].activity); //Space before %[^\n]s
    }
}

// Function to display the calendar
void display(struct Day calendar[], int size)
{
    printf("\n\nWeek's Activity Details Report:\n");
    for (int i = 0; i < size; i++)
    {
        printf("Day %d:\t", i + 1);
        printf("Day Name: %s\t", calendar[i].name);
        printf("Date: %d\t", calendar[i].date);
        printf("Activity: %s\n", calendar[i].activity);
    }
}

// Function to free memory allocated for the calendar
void freeMemory(struct Day calendar[], int size)
{
    for (int i = 0; i < size; i++)
    {
        free(calendar[i].name);
        free(calendar[i].activity);
    }
}

int main()
{
    int size;
    printf("Enter the number of days in the calendar: ");
    scanf("%d", &size);
   struct Day calendar[size];     // Declare array of structure as required by user
    create(calendar, size);     // Allocate Memory
    if (calendar == NULL)      //check if memory allocation failed
    {
       printf("Memory allocation failed.\n");
        return 1;
}
read(calendar,size);
display(calendar, size);
freeMemory(calendar, size);
return 0;
}
```

OUTPUT:

```
Enter the number of days in the calendar: 7
Enter details for day 1:
Enter day name: Monday
Enter date: 1
Enter activity: Reading
Enter details for day 2:
Enter day name: Tuesday
Enter date: 2
Enter activity: Writing
Enter details for day 3:
Enter day name: Wednesday
Enter date: 3
Enter activity: Revision
Enter details for day 4:
Enter day name: Thursday
Enter date: 4
Enter activity: Assignment
Enter details for day 5:
Enter day name: Friday
Enter date: 5
Enter activity: Quiz
Enter details for day 6:
Enter day name: Saturday
Enter date: 6
Enter activity: Weekend
Enter details for day 7:
Enter day name: Sunday
Enter date: 7
Enter activity: Holiday
```

Week's Activity Details Report:

Day 1:	Day Name: Monday	Date: 1	Activity: Reading
Day 2:	Day Name: Tuesday	Date: 2	Activity: Writing
Day 3:	Day Name: Wednesday	Date: 3	Activity: Revision
Day 4:	Day Name: Thursday	Date: 4	Activity: Assignment
Day 5:	Day Name: Friday	Date: 5	Activity: Quiz
Day 6:	Day Name: Saturday	Date: 6	Activity: Weekend
Day 7:	Day Name: Sunday	Date: 7	Activity: Holiday

2. Develop 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.

PROGRAM:

```
#include<stdio.h>
char str[100], pat[50], rep[50], ans[100];
int i, j, c, m, k, flag=0;
void stringmatch()
{
i = m = c = j = 0;
while(str[c]! = '\0')
{
if(str[m] = = pat[i]) // ..... matching
{
i++; m++;
if(pat[i] = = '\0')
//.....found occurrences.
{
flag = 1;
//.... copy replace string in ans string.
for(k = 0; rep[k] != '\0'; k++, j++)
ans[j] = rep[k];
i = 0;
c = m;
}
}
// if ends.
else
//... mismatch
{
ans[j] = str[c];
j++;
c++;
m = c;
i = 0;
}//else ends
```

```
} //end of while
ans[j] = '\0';
} //end stringmatch()
void main()
{
printf("\nEnter a main string \n");
gets(str);
printf("\nEnter a pattern string \n");

gets(pat);
printf("\nEnter a replace string \n");
gets(rep);
stringmatch();
if(flag = = 1)
printf("\nThe resultant string is\n %s" , ans);
else
printf("\nPattern string NOT found\n");

}
// end of main
```

OUTPUT:

RUN 1: Enter a main string:
Test
Enter a pattern string:
St
Enter a replace string:
ar
The resultant string is:
tear

RUN 2:
Enter a main string:
Vemana
Enter a pattern string:
ka
Enter a replace string:
Ta
Pattern string NOT found

3. Develop 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.

PROGRAM:

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

int s[MAX];
int top = -1;

void push(int item);
int pop();
void palindrome();
void display();

void main()
{
        int choice, item;
        while(1)
        {
                printf("\n\n\n\n~~~~~~Menu~~~~~~~ : ");
                printf("\n=>1.Push an Element to Stack and Overflow demo ");
                printf("\n=>2.Pop an Element from Stack and Underflow demo");
                printf("\n=>3.Palindrome demo ");
                printf("\n=>4.Display ");
                printf("\n=>5.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:         printf("\nEnter an element to be pushed: ");
                                        scanf("%d", &item);
                                        push(item);
                                        break;
```

```
                        case 2:     item = pop();
                                    if(item != -1)
                                        printf("\nElement popped is: %d", item);
                                    break;
                        case 3:     palindrome();
                                    break;
                        case 4:     display();
                                    break;
                        case 5:     exit(1);
                        default:    printf("\nPlease enter valid choice ") ;
                                    break;
                }
        }
}

void push(int item)
{
        if(top == MAX-1)
        {
                printf("\n~~~~Stack overflow~~~~");
                return;
        }

        top = top + 1 ;
        s[top] = item;
}

int pop()
{
        int item;
        if(top == -1)
        {
                printf("\n~~~~Stack underflow~~~~");
                return -1;
        }
        item = s[top];
        top = top - 1;
        return item;
}

void display()
{
        int i;
        if(top == -1)
        {
```

```
                printf("\n~~~~Stack is empty~~~~");
                return;
        }
        printf("\nStack elements are:\n ");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);
}

void palindrome()
{
        int flag=1,i;
        printf("\nStack content are:\n");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);

        printf("\nReverse of stack content are:\n");
        for(i=0; i<=top; i++)
                printf("| %d |\n", s[i]);

        for(i=0; i<=top/2; i++)
        {
                if( s[i] != s[top-i] )
                {
                        flag = 0;
                        break;
                }
        }
        if(flag == 1)
        {
                printf("\nIt is palindrome number");
        }
        else
        {
                printf("\nIt is not a palindrome number");
        }
}
```

OUTPUT:

```
~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
```

=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 12

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 13

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 14

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 15

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
~~~~~~

=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 16
~~~~Stack overflow~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **4**

Stack elements are:
| 15 |
| 14 |
| 13 |
| 12 |
| 11 |

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 15

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **4**
Stack elements are:
| 14 |
| 13 |
| 12 |
| 11 |
~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo

=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 14

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 13

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 12

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
~~~~Stack underflow~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
~~~~~~

=>4.Display
=>5.Exit
Enter your choice: **4**
~~~~Stack is empty~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **3**
Stack content are:
| 11 |
| 22 |
| 11 |

Reverse of stack content are:

| 11 |
| 22 |
| 11 |

It is palindrome number

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
~~~~~~

=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 33

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **3**
Stack content are:
| 33 |
| 22 |
| 11 |

Reverse of stack content are:
| 11 |
| 22 |
| 33 |

It is not a palindrome number

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **5**
~~~~~~

4. Develop 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.

PROGRAM

```
#include <stdio.h>
#include <string.h>
int f(char symbol)
{
 switch(symbol)
 {
  case '+':
  case '-':return 2;
  case '*':
  case '/':return 4;
  case '^':
  case '$':return 5;
  case '(':return 0;
  case '#':return -1;
  default:return 8;
 }
}
int g(char symbol)
{
 switch(symbol)
 {
  case '+':
  case '-':return 1;
  case '*':
  case '/':return 3;
  case '^':
  case '$':return 6;
  case '(':return 9;
  case ')':return 0;
  default:return 7;
 }
}
void infix_postfix(char infix[],char postfix[])
{
 int top,i,j;
 char s[30],symbol;
 top= -1;
 s[++top]='#';
```

```
 j=0;
 for(i=0;i<strlen(infix);i++)
 {
  symbol=infix[i];
  while(f(s[top])>g(symbol))
   {
    postfix[j]=s[top--];
    j++;
   }
    if(f(s[top])!=g(symbol))
      s[++top]=symbol;
  else
      top--;
     }
  while(s[top]!='#')
   {
    postfix[j++]=s[top--];
   }
  postfix[j]='\0';
}

void main()
{
 char infix[20];
 char postfix[20];
 printf("Enter a valid infix expression\n");
 scanf("%s",infix);
 infix_postfix(infix,postfix);
 printf("the post fix expression is\n");
 printf("%s",postfix);
}
```

OUTPUT:

```
 Enter a valid infix expression
 A+(B*C-(D/E^F)*G)*H
 the post fix expression is
  ABC*DEF^/G*-H*+
```

5. Develop 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.**

PROGRAM 5A

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
double compute(char symbol,double op1,double op2)
{
 switch(symbol)
 {
  case'+':return op1+op2;
  case'-':return op1-op2;
  case'*':return op1*op2;
  case'/':return op1/op2;
  case'^':return pow(op1,op2);
  case'%':return op1%op2;
  return 0;
 }
}
void main()
{
 double s[20],res,op1,op2;
 int top,i;
 char postfix[20],symbol;
 printf("Enter the postfix expression\n");
 scanf("%s",postfix);
 top=-1;
 for(i=0;i<strlen(postfix);i++)
 {
  symbol=postfix[i];
  if(isdigit(symbol))
  {
   s[++top]=symbol-'0';
  }
  else
  {
   op2=s[top--];
   op1=s[top--];
   res=compute(symbol,op1,op2);
   s[++top]=res;
```

```
  }
 }
 res=s[top--];
 printf("the result is %f\n",res);
}
```

OUTPUT:

```
Enter the postfix expression
562+*124/-
the result is 0.500000
```

PROGRAM 5B

```
#include<stdio.h>
void tower(int n,int source,int temp,int destination);
void main()
{
 int n;
 printf("enter the number of disc\n");
 scanf("%d",&n);
 tower(n,'A','B','C');
}
void tower(int n,int source,int temp,int destination)
{
 if(n==0)
  return;
 tower(n-1,source,destination,temp);
 printf("move disc %d from %c to %c\n",n,source,destination);
 tower(n-1,temp,source,destination);
}
```

OUTPUT:

```
enter the number of disc
3
move disc 1 from A to C
move disc 2 from A to B
move disc 1 from C to B
move disc 3 from A to C
move disc 1 from B to A
move disc 2 from B to C
move disc 1 from A to C
```

6. Develop 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.

PROGRAM:

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

#define MAXSIZE 5

char cq[MAXSIZE];
int front, rear;

void insert(char item);
void del( );
void display( );

int main( )
{
char item;
int choice,i;
front=-1;
rear=-1;
        do
        {
                printf("\n\n-------CIRCULAR QUEUE MENU----------\n");
                printf("\n1. INSERT INTO QUEUE \n2. DELETE FROM QUEUE \n3.
DISPLAY                                 QUEUE \n4. EXIT\n");
                printf("\n\nENTER YOUR CHOICE: ");
                scanf("%d",&choice);
                switch(choice)
                {
                                case 1: printf("\nENTER THE QUEUE ELEMENT : ");
                                        scanf("%s",&item);
                                        insert(item);
                                        break;
```

```
                        case 2: del();
                        break;
                        case 3: display();
                                break;
                        case 4: exit(0);
                        default: printf("\nInvalid Choice.\n");
                }
        }while(choice!=4);
                return 0;
        }
                void insert(char item)
                {
                        if(front==(rear+1)%MAXSIZE)
                        printf("\n\nCIRCULAR QUEUE IS OVERFLOW\n");
                        else
                        {
                                if(front==-1)
                                front=rear=0;
                                else
                                        rear=(rear+1)%MAXSIZE;
                                        cq[rear]=item;
                                        printf("\nRear = %d Front = %d \n",rear,front);



                        }
                }
                void del()
                {
                        char item;
                        if(front==-1)
                        printf("\nCIRCULAR QUEUE IS UNDERFLOW\n");
                        else
                        {
                                item=cq[front];
                                cq[front]='0';
                                if(front==rear)
                                front=rear=-1;
                                else
                                front=(front+1)%MAXSIZE;
                                printf("\nDELETED ELEMENT FROM QUEUE                IS :
%c \n",item);
                                printf("\nRear = %d Front = %d \n",rear,front);
                        }
                }
```

```
void display()
{
        int i;
        if(front==-1)
        printf("CIRCULAR QUEUE IS EMPTY\n");
        else
        {
                printf("The Queue Elements are\n");
                for(i=0;i<MAXSIZE;i++)
                printf("%c\t",cq[i]);
        }
}
```

OUTPUT:

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : a

Rear = 0 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : b

Rear = 1 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : c

Rear = 2 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : d

Rear = 3 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : d

Rear = 4 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : e

CIRCULAR QUEUE IS OVERFLOW

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : f

CIRCULAR QUEUE IS OVERFLOW

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : g

CIRCULAR QUEUE IS OVERFLOW

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : j

CIRCULAR QUEUE IS OVERFLOW

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 3
The Queue Elements are
a b c d d

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : y

CIRCULAR QUEUE IS OVERFLOW

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : a

Rear = 4 Front = 1

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : b

Rear = 4 Front = 2

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : c

Rear = 4 Front = 3

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : d

Rear = 4 Front = 4

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : d

Rear = -1 Front = -1

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : b

Rear = 0 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 3
The Queue Elements are
b 0 0 0 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 1

ENTER THE QUEUE ELEMENT : m

Rear = 1 Front = 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 3
The Queue Elements are
b m 0 0 0

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE

2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 2

DELETED ELEMENT FROM QUEUE IS : b

Rear = 1 Front = 1

-------CIRCULAR QUEUE MENU----------

1. INSERT INTO QUEUE
2. DELETE FROM QUEUE
3. DISPLAY QUEUE
4. EXIT

ENTER YOUR CHOICE: 4

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

a) **Create a SLL of N Students Data by using front insertion.**
b) **Display the status of SLL and count the number of nodes in it**
c) **Perform Insertion and Deletion at End of SLL**
d) **Perform Insertion and Deletion at Front of SLL (Demonstration of stack)**
e) **exit**

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int MAX=4, count;
struct student
{
        char usn[20];
        char name[30];
        char branch[5];
        int sem;
        char phno[10];
        struct student *next;
};//Self referential pointer.
typedef struct student NODE;
int countnodes(NODE *head)
{
        NODE *p;
        count=0;
        p=head;
        while(p!=NULL)
        {
                p=p->next;
                count++;
        }
        return count;
}

NODE* getnode(NODE *head)
{
        NODE *newnode;
        newnode=(NODE*)malloc(sizeof(NODE));
        //Create first NODE
        printf("\nEnter USN, Name, Branch, Sem, Ph.No\n");
```

```
	scanf("\n%s%s%s%d%s",newnode->usn,newnode->name,newnode->branch,&(newnode->sem),newnode->phno);

	newnode->next=NULL;
	//Set next to NULL...
	head=newnode;
	return head;
}
NODE* display(NODE *head)
{
	NODE *p;
	if(head == NULL)
	printf("\nNo student data\n");
	else
	{
		p=head;
		printf("\n----STUDENT DATA----\n");
		printf("\nUSN\tNAME\t\tBRANCH\tSEM\tPh.NO.");
		while(p!=NULL)
		{
			printf("\n%s\t%s\t\t%s\t%d\t%s", p->usn, p->name, p->branch, p->sem, p->phno);
			p = p->next;
//Go to next node...
		}
	printf("\nThe no. of nodes in list is: %d",countnodes(head));
	}
return head;
}
NODE* create(NODE *head)
	{
		NODE *newnode;
		if(head==NULL)
		{
			newnode=getnode(head);
			head=newnode;
		}

		else
		{
			newnode=getnode(head);
			newnode->next=head;
			head=newnode;
		}
		return head;
```

```
	}
NODE* insert_front(NODE *head)
	{
			if(countnodes(head)==MAX)
			printf("\nList is Full / Overflow!!");
			else
				head=create(head); //create()insert nodes at front.
			return head;
	}
NODE* insert_rear(NODE *head)
{
		NODE *p, *newnode;
		p=head;
		if(countnodes(head) == MAX)
		printf("\nList is Full(QUEUE)!!");
		else
		{
			if(head == NULL)
			{
			newnode=getnode(head);
			head=newnode;
			//set first node to be head
			}
				else
				{
					newnode=getnode(head);
					while(p->next!=NULL)
					{
						p=p->next;
					}
					p->next=newnode;
				}
		}
		return head;
}

NODE* insert(NODE *head)
{
		int ch;
		do
		{
			printf("\n1.Insert at Front(First)\t2.Insert at End(Rear/Last)\t3.Exit");
			printf("\nEnter your choice: ");
			scanf("%d", &ch);
		switch(ch)
```

```
                {
                        case 1: head=insert_front(head); break;
                        case 2: head=insert_rear(head); break;
                        case 3: break;
                }
                        head=display(head);
                }while(ch!=3);
        return head;
}
NODE* delete_front(NODE *head)
{
        NODE *p;
        if(head==NULL)
        printf("\nList is Empty/Underflow (STACK/QUEUE)");
        else
        {
                p=head;
                head=head->next; /Set 2nd NODE as head
                free(p);
                printf("\nFront(first)node is deleted");
        }
return head;
}
NODE* delete_rear(NODE *head)
{
                NODE *p, *q;
                p=head;
                while(p->next!=NULL)
                {
                        q=p;
                        p=p->next; //Go upto -1 NODE which you want to delete
                }

                free(p);//Delete last NODE...
                q->next=NULL;
                printf("\nLast(end) entry is deleted");
                return head;
}

NODE* del(NODE *head)
{
        int ch;
        do
        {
                printf("\n1.Delete from Front(First)\t2. Delete from End(Rear/Last))\t3.Exit");
```

```
                printf("\nEnter your choice: ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: head=delete_front(head);
                                break;
                        case 2: head=delete_rear(head);
                                break;
                        case 3: break;
                }
                head=display(head);
        }while(ch!=3);
return head;
}
NODE* stack(NODE *head)
{
int ch;
        do
        {
        printf("\nSSL used as Stack...");
        printf("\n 1.Insert at Front(PUSH) \t 2.Delete from Front(POP))\t3.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
                case 1: head=insert_front(head); break;
                case 2: head=delete_front(head); break;
                case 3: break;
        }
                head=display(head);
        }while(ch!=3);
return head;
}
NODE* queue(NODE *head)
{
int ch;
do
{
        printf("\nSSL used as Queue...");
        printf("\n 1.Insert at Rear(INSERT) \t 2.Delete from Front(DELETE))\t3.Exit");
         printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
                case 1: head=insert_rear(head); break;
```

```
                case 2: head=delete_front(head); break;
                case 3: break;
        }
                head=display(head);
        }while(ch!=3);
return head;
}
void main()
{
int ch, i, n;
NODE *head;
head=NULL;

printf("\n*----------Studentt Database-----------*");
        do
                {
                        printf("\n 1.Create\t 2.Display\t 3.Insert\t 4.Delete\t 5.Stack\t 6.Queue\t
7.Exit");
                        printf("\nEnter your choice: ");
                        scanf("%d", &ch);
                switch(ch)
                {
                        case 1: printf("\nHow many student data you want to create: ");
                                scanf("%d", &n);
                                for(i=0;i<n;i++)
                                head=create(head);//Call to Create NODE...
                                break;
                        case 2: head=display(head); //Call to Display...
                                break;
                        case 3: head=insert(head); //Call to Insert...
                                break;
                        case 4: head=del(head);
                                //Call to delete
                                break;
                        case 5: head=stack(head);
                                break;
                        case 6: head=queue(head);
                                break;
                        case 7: exit(0);

                        }
                        }while(ch!=7);
}
```

OUTPUT:

```
*----------Student Database-----------*
 1.Create      2.Display     3.Insert      4.Delete      5.Stack       6.Queue       7.Exit
Enter your choice: 1

How many student data you want to create: 2

Enter USN, Name, Branch, Sem, Ph.No
1VI16CS004 ANAND CSE 3 9876356780

Enter USN, Name, Branch, Sem, Ph.No
1VI16CS006 BHARATHI CSE 4 236438945

 1.Create      2.Display     3.Insert      4.Delete      5.Stack       6.Queue       7.Exit
Enter your choice: 2

----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS006   BHARATHI          CSE   4     236438945
1VI16CS004   ANAND             CSE   3     9876356780
The no. of nodes in list is: 2
 1.Create      2.Display     3.Insert      4.Delete      5.Stack       6.Queue       7.Exit
Enter your choice: 3

1.Insert at Front(First)2.Insert at End(Rear/Last)      3.Exit
Enter your choice: 1

Enter USN, Name, Branch, Sem, Ph.No
1VI16CS94 CHITRA CSE 6 2329389345

----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS94    CHITRA            CSE   6     2329389345
1VI16CS006   BHARATHI          CSE   4     236438945
1VI16CS004   ANAND             CSE   3     9876356780
The no. of nodes in list is: 3
1.Insert at Front(First)2.Insert at End(Rear/Last)      3.Exit
Enter your choice: 2

Enter USN, Name, Branch, Sem, Ph.No
```

1VI16CS008 DIVYA ISE 8 298732367

----STUDENT DATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1VI16CS94	CHITRA	CSE	6	2329389345
1VI16CS006	BHARATHI	CSE	4	236438945
1VI16CS004	ANAND	CSE	3	9876356780
1VI16CS008	DIVYA	ISE	8	298732367

The no. of nodes in list is: 4
1.Insert at Front(First) 2.Insert at End(Rear/Last) 3.Exit
Enter your choice: 4

----STUDENT DATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1VI16CS94	CHITRA	CSE	6	2329389345
1VI16CS006	BHARATHI	CSE	4	236438945
1VI16CS004	ANAND	CSE	3	9876356780
1VI16CS008	DIVYA	ISE	8	298732367

The no. of nodes in list is: 4
1.Insert at Front(First) 2.Insert at End(Rear/Last) 3.Exit
Enter your choice: 3

----STUDENT DATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1VI16CS94	CHITRA	CSE	6	2329389345
1VI16CS006	BHARATHI	CSE	4	236438945
1VI16CS004	ANAND	CSE	3	9876356780
1VI16CS008	DIVYA	ISE	8	298732367

The no. of nodes in list is: 4
1.Create 2.Display 3.Insert 4.Delete 5.Stack 6.Queue 7.Exit
Enter your choice: 4

1.Delete from Front(First) 2. Delete from End(Rear/Last)) 3.Exit
Enter your choice: 1

Front(first)node is deleted
----STUDENT DATA----

USN	NAME	BRANCH	SEM	Ph.NO.
1VI16CS006	BHARATHI	CSE	4	236438945
1VI16CS004	ANAND	CSE	3	9876356780
1VI16CS008	DIVYA	ISE	8	298732367

```
The no. of nodes in list is: 3
1.Delete from Front(First)      2. Delete from End(Rear/Last))      3.Exit
Enter your choice: 2

Last(end) entry is deleted
----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS006   BHARATHI          CSE    4      236438945
1VI16CS004   ANAND             CSE    3      9876356780
The no. of nodes in list is: 2
1.Delete from Front(First)      2. Delete from End(Rear/Last))      3.Exit
Enter your choice: 3

----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS006   BHARATHI          CSE    4      236438945
1VI16CS004   ANAND             CSE    3      9876356780
The no. of nodes in list is: 2
 1.Create       2.Display      3.Insert       4.Delete       5.Stack       6.Queue       7.Exit
Enter your choice: 2

----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS006   BHARATHI          CSE    4      236438945
1VI16CS004   ANAND             CSE    3      9876356780
The no. of nodes in list is: 2
 1.Create       2.Display      3.Insert       4.Delete       5.Stack       6.Queue       7.Exit
Enter your choice: 5

SSL used as Stack...
 1.Insert at Front(PUSH)      2.Delete from Front(POP))    3.Exit
Enter your choice: 1

Enter USN, Name, Branch, Sem, Ph.No
1VI16CS002 PRIYA ECE 7 9876542689

----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS002   PRIYA             ECE    7      9876542689
1VI16CS006   BHARATHI          CSE    4      236438945
1VI16CS004   ANAND             CSE    3      9876356780
```

```
The no. of nodes in list is: 3
SSL used as Stack...
 1.Insert at Front(PUSH)        2.Delete from Front(POP))    3.Exit
Enter your choice: 2

Front(first)node is deleted
----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS006   BHARATHI          CSE    4        236438945
1VI16CS004   ANAND             CSE    3        9876356780
The no. of nodes in list is: 2
SSL used as Stack...
 1.Insert at Front(PUSH)        2.Delete from Front(POP))    3.Exit
Enter your choice: 3

----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS006   BHARATHI          CSE    4        236438945
1VI16CS004   ANAND             CSE    3        9876356780
The no. of nodes in list is: 2
 1.Create        2.Display       3.Insert        4.Delete        5.Stack         6.Queue         7.Exit
Enter your choice: 2

----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS006   BHARATHI          CSE    4        236438945
1VI16CS004   ANAND             CSE    3        9876356780
The no. of nodes in list is: 2
 1.Create        2.Display       3.Insert        4.Delete        5.Stack         6.Queue         7.Exit
Enter your choice: 6

SSL used as Queue...
 1.Insert at Rear(INSERT)       2.Delete from Front(DELETE))       3.Exit
Enter your choice: 1

Enter USN, Name, Branch, Sem, Ph.No
1VI16CS093 RAJ MECH 4 283733782

----STUDENT DATA----

USN    NAME          BRANCH      SEM   Ph.NO.
1VI16CS006   BHARATHI          CSE    4        236438945
```

```
1VI16CS004   ANAND              CSE   3       9876356780
1VI16CS093   RAJ          MECH  4       283733782
The no. of nodes in list is: 3
SSL used as Queue...
 1.Insert at Rear(INSERT)      2.Delete from Front(DELETE))      3.Exit
Enter your choice: 2

Front(first)node is deleted
----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS004   ANAND              CSE   3       9876356780
1VI16CS093   RAJ          MECH  4       283733782
The no. of nodes in list is: 2
SSL used as Queue...
 1.Insert at Rear(INSERT)      2.Delete from Front(DELETE))      3.Exit
Enter your choice: 3

----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS004   ANAND              CSE   3       9876356780
1VI16CS093   RAJ          MECH  4       283733782
The no. of nodes in list is: 2
 1.Create      2.Display      3.Insert       4.Delete       5.Stack       6.Queue       7.Exit
Enter your choice: 2

----STUDENT DATA----

USN    NAME          BRANCH      SEM    Ph.NO.
1VI16CS004   ANAND              CSE   3       9876356780
1VI16CS093   RAJ          MECH  4       283733782
The no. of nodes in list is: 2
 1.Create      2.Display      3.Insert       4.Delete       5.Stack       6.Queue       7.Exit
Enter your choice: 7
```

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

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

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int MAX=4, count;
struct emp
{
 int ssn;
 char name[20];
 char dept[10];
 char desig[15];
 int sal;
 char phno[10];
 struct emp *left;
 struct emp *right;
};
 typedef struct emp NODE;
 int countnodes(NODE *head)
{
   NODE *p;
   count=0;
   p=head;
   while(p!=NULL)
  {
     p=p->right;
     count++;
   }
    return count;
  }
NODE* getnode(NODE *head)
{
   NODE *newnode;
   newnode=(NODE*)malloc(sizeof(NODE));
   newnode->right=newnode->left=NULL;
```

```
   printf("\nEnter SSN, Name, Dept, Designation, Sal, Ph.No\n");
   scanf("%d%s%s%s%d%s",&newnode->ssn,newnode->name,newnode->dept,newnode-
>desig,&newnode->sal,newnode->phno);
   head=newnode;
return head;
}
NODE* display(NODE *head)
{
  NODE *p;
   if(head==NULL)
    printf("\nNo Employee data\n");
   else
 {
   p=head;
   printf("\n----EMPLOYEE DATA----\n");
   printf("\nSSN\tNAME\tDEPT\tDESINGATION\tSAL\t\tPh.NO.");
   while(p!=NULL)
  {
       printf("\n%d\t%s\t%s\t%s\t\t%d\t\t%s", p->ssn, p->name, p->dept, p->desig,
       p->sal, p->phno);
       p = p->right;
     //Go to next node...
   }
    printf("\nThe no. of nodes in list is: %d",countnodes(head));
  }
  return head;
}
NODE* create(NODE *head)// creating & inserting at end.
{
    NODE *p, *newnode;
    p=head;
    if(head==NULL)
  {
     newnode=getnode(head);
     head=newnode;
   }
 else
  {
    newnode=getnode(head);
    while(p->right!=NULL)
  {
     p=p->right;
   }
   p->right=newnode;
    newnode->left=p;
```

```
  }
  return head;
}

NODE* insert_end(NODE *head)
{
   if(countnodes(head)==MAX)
   printf("\nList is Full!!");
   else
   head=create(head);
   return head;
}
NODE* insert_front(NODE *head)
{
   NODE *p, *newnode;
   if(countnodes(head)==MAX)
   printf("\nList is Full!!");
  else
  {
    if(head==NULL)
   {
     newnode=getnode(head);
    head=newnode;
    //set first node to be head
   }
  else
{
   newnode=getnode(head);
   newnode->right=head;
   head->left=newnode;
   head=newnode;
}
}
return head;
}
NODE* insert(NODE *head)
{
  int ch;
  do
{
  printf("\n 1.Insert at Front(First) \t 2.Insert at End(Rear/Last)\t3.Exit");
  printf("\nEnter your choice: ");
  scanf("%d", &ch);
  switch(ch)
 {
```

```
    case 1: head=insert_front(head); break;
    case 2: head=insert_end(head); break;
    case 3: break;
 }

   head=display(head);
}while(ch!=3);
return head;
}
NODE* delete_front(NODE *head)
{
  NODE *p;
  if(head==NULL)
  printf("\nList is Empty (QUEUE)");
  else
{
  p=head;
  head->left=NULL;
  p->right=NULL;
 free(p);
 printf("\nFront(first)node is deleted");
}
 return head;
}
NODE* delete_end(NODE *head)
{
   NODE *p, *q;
   p=head;
   while(p->right!=NULL)
 {
    p=p->right;
  //Go upto -1 node which you want to delete
 }
  q=p->left;
  q->right=NULL;
  p->left=NULL;
   free(p);//Delete last node...
   printf("\nLast(end) entry is deleted");
return head;
}
NODE *del(NODE *head)
{
  int ch;
  do {
   printf("\n1.Delete from Front(First)\t2. Delete from End(Rear/Last))\t3.Exit");
```

```
  printf("\nEnter your choice: ");
  scanf("%d", &ch);
  switch(ch)
 {
  case 1: head=delete_front(head);
          break;
  case 2: head=delete_end(head);
          break;
  case 3: break;
        return 0;
}
  head=display(head);
}while(ch!=3);
return head;
}
NODE* queue(NODE *head)
{
  int ch, ch1, ch2;
 do
 {
printf("\nDLL used as Double Ended Queue");
printf("\n1.QUEUE- Insert at Rear & Delete from Front");
printf("\n2.QUEUE- Insert at Front & Delete from Rear");
printf("\n3.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
  case 1: do{
           printf("\n1.Insert at Rear\t2.Delete from From Front\t3.Exit");
           printf("\nEnter your choice: ");
          scanf("%d", &ch1);
           switch(ch1)
           {
              case 1: head=insert_end(head);
                      head=display(head);
                      break;
               case 2: head=delete_front(head);
                       head=display(head);
                        break;
               case 3: break;
            }
          }while(ch1!=3);
          break;
    case 2: do{
```

```
                    printf("\n1.Insert at Front\t2.Delete from Rear\t3.Exit");
                    printf("\nEnter your choice: ");
                    scanf("%d", &ch2);
                    switch(ch2)
                   {
                      case 1: head=insert_front(head);
                              head=display(head);
                              break;
                      case 2: head=delete_end(head);
                              head=display(head);
                              break;
                      case 3: break;
                   }
                 }while(ch2!=3);
                  break;
      case 3: break;
  }
}while(ch!=3);
head=display(head);
return head;
}
void main()
{
   int ch, i, n;
  NODE *head;
  head=NULL;
printf("\n----------Employee Database-----------");
do
{
    printf("\n1.Create\t2.Display\t3.Insert\t4.Delete\t5.Queue\t6.Exit");
    printf("\nEnter your choice: ");
     scanf("%d", &ch);
  switch(ch)
  {
   case 1: printf("\nHow many employees data you want to create: ");
           scanf("%d", &n);
           for(i=0;i<n;i++)
           head=create(head);//Call to Create node...
           break;
   case 2: head=display(head); //Call to Display...
           break;
   case 3: head=insert(head); //Call to Insert...
           break;
   case 4: head=del(head);//Call to delete
           break;
```

```
   case 5: head=queue(head);
            break;
    case 6: exit(0);
           //Exit...
           break;
  }
 }while(ch!=6);
}
```

OUTPUT:

```
 1.create 2.display 3.insert 4.delete 5.queue 6.exit
 Enter your choice: 1
 How many employee details you want to create: 2
 Enter sss,name,dept,desig,sal and phno:
 2243 abcd cse hod 56800 9874563214
 Enter ssn,name,dept,desig,sal and phno:
 2232 efgh cse ahod 56800 7789654123
 1.create 2.display 3.insert 4.delete 5.queue 6.exit
 Enter your choice: 2
 …employee data…
 Ssn     name   dept     desig   sal       phno
 2243   abcd    cse      hod     56800   9874563214
 2232 efgh      cse      ahod    56800 7789654123

 1.create 2.display 3.insert 4.delete 5.queue 6.exit
 Enter your choice: 3
 1.insert from front 2.insert from rear 3.exit
  Enter your choice: 1
 Enter ssn,name,dept,desig,sal and phno:
 2342 hijk cse prof 45000 7896541230
 …employee data…
 Ssn     name   dept     desig   sal       phno
 2342 hijk      cse      prof    45000   7896541230
 2243   abcd    cse      hod     56800   9874563214
 2232 efgh      cse      ahod    56800 7789654123
 The no of node:3
 1,insert at front 2,insert at rear 3.exit
 Enter your choice: 3
  …employee data…
 Ssn     name   dept     desig   sal       phno
 2342 hijk      cse      prof    45000   7896541230
 2243   abcd    cse      hod     56800   9874563214
 2232 efgh      cse      ahod    56800 7789654123
 The no of node:3
```

```
1.create 2.display 3.insert 4.delete 5.queue 6.exit
Enter your choice: 4
1.delete from front 2.delete from rear 3.exit
Enter your choice: 1
Front node deleted
 …employee data…
Ssn    name   dept    desig   sal      phno
2243   abcd   cse     hod     56800    9874563214
2232 efgh     cse     ahod    56800 7789654123
The no of nodes : 2
1.delete from front 2.delete from rear 3.exit
Enter your choice: 2
Last entry deleted
  …employee data…
Ssn    name   dept    desig   sal      phno
2243   abcd   cse     hod     56800    9874563214
The no of nodes: 1
1.delete from front 2.delete from rear 3.exit
Enter your choice: 3

1.create 2.display 3.insert 4.delete 5.queue 6.exit
 Enter your choice: 5
 Dll uses as double ended queue
 1.queue-insert at rear and delete from front
 2.queue-insert at front and delete from rear
 3.exit
Enter your choice: 1
1.insert at rear 2.delete from front 3.exit
Enter your choice: 1
Enter ssn,name,dept,desig,sal and phno:
2232 efgh cse ahod 56800 7789654123
  …employee data…
Ssn    name   dept    desig   sal      phno
2243   abcd   cse     hod     56800    9874563214
2232 efgh     cse     ahod    56800 7789654123
The no of nodes : 2
1,insert at rear 2.delete from front 3.exit
Enter your choice: 2
Front node deleted
   …employee data…
Ssn    name   dept    desig   sal      phno
2232  efgh     cse     ahod    56800 7789654123
The no of nodes: 1
1.insert at rear 2.delete from front 3.exit
Enter your choice: 3
```

9. Develop 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.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
// Define the structure for a node in the circular linked list
struct Node {
int coeff_x;
int coeff_y;
int coeff_z;
int power_x;
int power_y;
int power_z;
struct Node *next;
};
// Function prototypes
void insertTerm(struct Node **head, int coeff_x, int coeff_y, int coeff_z,
int power_x, int power_y, int power_z);
void displayPolynomial(struct Node *head);
void addPolynomials(struct Node *poly1, struct Node *poly2, struct Node **result);
int evaluatePolynomial(struct Node *head, int x, int y, int z);
int main()
{
// Define and initialize the header node for each polynomial
struct Node *poly1 = NULL;
struct Node *poly2 = NULL;
struct Node *polySum = NULL;
// Represent and evaluate POLY1
insertTerm(&poly1, 6, 1, 1, 2, 2, 1);
insertTerm(&poly1, 1, -4, 1, 0, 1, 5);
insertTerm(&poly1, 3, 1, 1, 3, 1, 1);
insertTerm(&poly1, 2, 1, 1, 1, 5, 1);
insertTerm(&poly1, -2, 1, 1, 1, 1, 3);
// Display POLY1
printf("POLY1(x, y, z) = ");
displayPolynomial(poly1);
printf("\n");
```

```
poly2=poly1;
// Creating Poly2
addPolynomials(poly1, poly2, &polySum);
// Display POLYSUM
printf("POLYSUM(x, y, z) = ");
displayPolynomial(polySum);
printf("\n");
// Evaluate POLY1, POLY2, and POLYSUM for specific values of x, y, and z
int x = 2, y = 3, z = 1;
printf("\n");
printf("Evaluate POLY1(x=2, y=3, z=1): %d\n", evaluatePolynomial(poly1, x, y, z));
return 0;
}
// Function to insert a term into the circular linked list
void insertTerm(struct Node **head, int coeff_x, int coeff_y, int coeff_z,
     int power_x, int power_y, int power_z)
{
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->coeff_x = coeff_x;
   newNode->coeff_y = coeff_y;
   newNode->coeff_z = coeff_z;
   newNode->power_x = power_x;
   newNode->power_y = power_y;
   newNode->power_z = power_z;

   if (*head == NULL) {
      newNode->next = newNode; // Points to itself for circular list
      *head = newNode;
   }
  else
  {
      struct Node *temp = *head;
      while (temp->next != *head)
          temp = temp->next;

temp->next = newNode;
       newNode->next = *head;
   }
}

// Function to display the polynomial
void displayPolynomial(struct Node *head)
{
   struct Node *temp = head;
   do {
```

```
        printf("(%dx^%d %dy^%d %dz^%d)", temp->coeff_x, temp->power_x,
temp->coeff_y, temp->power_y, temp->coeff_z, temp->power_z);
        temp = temp->next;
        if (temp != head)
            printf(" + ");
    } while (temp != head);
}

// Function to add two polynomials
void addPolynomials(struct Node *poly1, struct Node *poly2, struct Node **result)
{
    struct Node *temp1 = poly1;
    struct Node *temp2 = poly2;

    // Iterate through both polynomials
    do {
        // Add the coefficients of like terms
        insertTerm(result,
                   temp1->coeff_x + temp2->coeff_x,
                   temp1->coeff_y + temp2->coeff_y,
                   temp1->coeff_z + temp2->coeff_z,
                   temp1->power_x,
                   temp1->power_y,
                   temp1->power_z);

        temp1 = temp1->next;
        temp2 = temp2->next;
    } while (temp1 != poly1 && temp2 != poly2);
}

// Function to evaluate the polynomial for specific values of x, y, and z
int evaluatePolynomial(struct Node *head, int x, int y, int z)
{
    struct Node *temp = head;
    int result = 0;
    do {
        int termValue = temp->coeff_x * pow(x, temp->power_x) *
                        temp->coeff_y * pow(y, temp->power_y) *
                        temp->coeff_z * pow(z, temp->power_z);
        result += termValue;
        temp = temp->next;
    } while (temp != head);
    return result;
}
```

OUTPUT:

POLY1(x, y, z) = (6x^2 1y^2 1z^1) + (1x^0 -4y^1 1z^5) + (3x^3 1y^1 1z^1) + (2x^1 1y^5 1z^1) + (-2x^1 1y^1 1z^3)

POLYSUM(x, y, z) = (12x^2 2y^2 2z^1) + (2x^0 -8y^1 2z^5) + (6x^3 2y^1 2z^1) + (4x^1 2y^5 2z^1) + (-4x^1 2y^1 2z^3)

Evaluate POLY1(x=2, y=3, z=1): 1236

10. Develop 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**

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct BST
{
int data;
struct BST *left;
struct BST *right;
};
typedef struct BST NODE;
NODE *node;
NODE* createtree(NODE *node, int data)
        {
                if (node == NULL)
                {
                        NODE *temp;
                        temp= (NODE*)malloc(sizeof(NODE));
                        temp->data = data;
                        temp->left = temp->right = NULL;
                return temp;
                }
                if (data < (node->data))
                {
                        node->left = createtree(node->left, data);
                }
                else if (data > node->data)
                {
                        node -> right = createtree(node->right, data);
                }
        return node;
        }

NODE* search(NODE *node, int data)
{
        if(node == NULL)
        printf("\nElement not found");
```

```
        else if(data < node->data)
        {
                node->left=search(node->left, data);
        }
        else if(data > node->data)
        {
                node->right=search(node->right, data);
        }
        else
                printf("\nElement found is: %d", node->data);
        return node;
}
void inorder(NODE *node)
{
        if(node != NULL)
        {
                inorder(node->left);
                printf("%d\t", node->data);
                inorder(node->right);
        }
}
void preorder(NODE *node)
{
        if(node != NULL)
        {
                printf("%d\t", node->data);
                preorder(node->left);
                preorder(node->right);
        }
}
void postorder(NODE *node)
{
        if(node != NULL)
        {
                postorder(node->left);
                postorder(node->right);
                printf("%d\t", node->data);
        }
}


void main()
{
int data, ch, i, n;
NODE *root=NULL;
```

```
while (1)
{
printf("\n1.Insertion in Binary Search Tree");
printf("\n2.Search Element in Binary Search Tree");
printf("\n3.Inorder\n4.Preorder\n5.Postorder\n6.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch (ch)
{
        case 1: printf("\nEnter N value: " );
        scanf("%d", &n);
        printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
        for(i=0; i<n; i++)
        {
                scanf("%d", &data);
                root=createtree(root, data);
        }
        break;

case 2: printf("\nEnter the element to search: ");
        scanf("%d", &data);
        root=search(root, data);
        break;

case 3: printf("\nInorder Traversal: \n");
        inorder(root);
        break;

case 4: printf("\nPreorder Traversal: \n");
        preorder(root);
        break;

case 5: printf("\nPostorder Traversal: \n");
        postorder(root);
        break;
case 6: exit(0);
default:printf("\nWrong option");
break;
}
}
}
```

OUTPUT:

```
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 1

Enter N value: 12

Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)
6 9 5 2 8 15 24 14 7 8 5 2

1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 3

Inorder Traversal:
2       5       6       7       8       9       14      15      24
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 4

Preorder Traversal:
6       5       2       9       8       7       15      14      24
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 5

Postorder Traversal:
```

```
2       5       7       8       14      24      15      9       6
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 2

Enter the element to search: 50

Element not found
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 2

Enter the element to search: 5

Element found is: 5

1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Inorder
4.Preorder
5.Postorder
6.Exit
Enter your choice: 6
```

11. Develop 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**

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int a[10][10], n, i, j, source, queue[10];
int visit_bfs[10], visit_dfs[10];

void create_graph()
{
   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 source)
{
   int u, front=0, rear=-1, k=1;   //k is to hold sequence of vertices considered
   queue[++rear] = source;
   visit_bfs[source] = 1;
   printf("%d\t", source);
   while(front<=rear)
   {
      u = queue[front++];
      for(i=1; i<=n; i++)
      {
         if(a[u][i] == 1 && visit_bfs[i] == 0)
         {
            queue[++rear] = i;
            visit_bfs[i] = 1;
            printf("%d\t", i);
         }
      }
   }
}

void dfs(int source)
{
```

```
    int  v, top = -1, p=1;
    visit_dfs[source] = 1;
    printf("%d\t", source);
    for(v=1; v<=n; v++)
    {
        if(a[source][v] == 1 && visit_dfs[v] == 0)
        {

            dfs(v);
        }
    }
}

void main()
{
    int ch;
    while(1)
    {
        printf("\n---------- GRAPH TRAVERSALS ----------");
        printf("\n 1.Create Graph\n 2.BFS & DFS\n 3.Exit");
        printf("\nEnter your choice:    ");
        scanf("%d", &ch);
        switch(ch)
        {
        case 1:
                    create_graph();
                    break;
        case 2:
                    printf("\nEnter the source vertex to find other nodes reachable or not:  ");
                    scanf("%d", &source);
                    printf("Nodes Reachable from Source Vertex using BFS\n");
                    bfs(source);
                    for(i=1; i<=n; i++)
                          if(visit_bfs[i]==0)
                             printf("\nThe vertex that is not reachable  %d",i);

                    printf("\nNodes Reachable from Source Vertex using DFS\n");
                    dfs(source);

                    break;
        default:
                    exit(0);
        }
    }
}
```

OUTPUT:

```
---------- GRAPH TRAVERSALS ----------
 1.Create Graph
 2.BFS & DFS
 3.Exit
Enter your choice:    1
Enter the number of vertices of the digraph: 4
Enter the adjacency matrix of the graph:
0 0 1 1
0 0 0 0
0 1 0 0
0 1 0 0

---------- GRAPH TRAVERSALS ----------
 1.Create Graph
 2.BFS & DFS
 3.Exit

Enter your choice:    2
Enter the source vertex to find other nodes reachable or not:  1
Nodes Reachable from Source Vertex using BFS
1 3 4 2
Nodes Reachable from Source Vertex using DFS
1 3 2 4

---------- GRAPH TRAVERSALS ----------
 1.Create Graph
 2.BFS & DFS
 3.Exit
Enter your choice:    3
```

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 .

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
int create(int);
void linear_prob(int[], int, int);
void display (int[]);
 void main()
  {
        int a[MAX],num,key,i;
        int ans=1;
        printf(" collision handling by linear probing : \n");
         for (i=0;i<MAX;i++)
           {
                 a[i] = -1;
           }
                  do
                  {
                           printf("\n Enter the data:");
                           scanf("%4d", &num);
                            key=create(num);
                            linear_prob(a,key,num);
                             printf("\n Do you wish to continue ? (1/0) ");
                             scanf("%d",&ans);

                  }while(ans);
             display(a);
  }
int create(int num)
  {
          int key;
          key=num%100;
          return key;
```

```
   }
void linear_prob(int a[MAX], int key, int num)
{
   int flag, i, count=0;
   flag=0;
   if(a[key]== -1)
   {
         a[key] = num;
   }
    else
    {
            printf("\nCollision Detected...!!!\n");
            i=0;
              while(i<MAX)
              {
                    if (a[i]!=-1)
                      count++;
                           i++;
              }
                    printf("Collision avoided successfully using LINEAR PROBING\n");
                    if(count == MAX)
                    {
                       printf("\n Hash table is full");
                       display(a);
                             exit(1);
                    }
                       for(i=key+1; i<MAX; i++)
                       if(a[i] == -1)
                       {
                                  a[i] = num;
                                    flag =1;
                                    break;
                       }

                       i=0;
                              while((i<key) && (flag==0))
                              {
                                    if(a[i] == -1)
                                    {

                                       a[i] = num;
                                       flag=1;
                                       break;
                                    }
                                i++;
```

```
                        }
                }
}
void display(int a[MAX])
{
     int i,choice;
     printf("1.Display ALL\n 2.Filtered Display\n");
          scanf("%d",&choice);
          if(choice==1)
            {
                    printf("\n the hash table is\n");
                    for(i=0; i<MAX; i++)
                    printf("\n %d %d ", i, a[i]);
            }
   else
   {
     printf("\n the hash table is\n");
     for(i=0; i<MAX; i++)
     if(a[i]!=-1)
     {
              printf("\n %d %d ", i, a[i]);
              continue;
     }
  }
}
```

OUTPUT:

collision handling by linear probing :

Enter the data:31

Do you wish to continue ? (1/0) 1

Enter the data:44

Do you wish to continue ? (1/0) 1

Enter the data:43

Do you wish to continue ? (1/0) 1

Enter the data:78

Do you wish to continue ? (1/0) 1

```
Enter the data:19

Do you wish to continue ? (1/0) 1

Enter the data:36

Do you wish to continue ? (1/0) 1

Enter the data:57

Do you wish to continue ? (1/0) 1

Enter the data:77

Collision Detected...!!!
Collision avoided successfully using LINEAR PROBING

Do you wish to continue ? (1/0) 0
1.Display ALL
2.Filtered Display
2

the hash table is

1 31
3 43
4 44
6 36
7 57
8 78
9 19
10 77
```