YENEPOYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belagavi)

N.H. 13, Thodar, Moodbidri, Mangaluru 574225 (D.K), Karnataka

DEPARTMENT OF CSE (IoT, Cyber Security IncludingBlock Chain Technology)

LAB MANUAL

DATA STRUCTURES LABORATORY

BCSL305

[As per Choice Based Credit System (CBCS) scheme]

VISION

To empower students such that they will be technologically adept, innovation driven, self-motivated and responsible global citizens possessing human values by imparting quality technical education in the field of Computer Science.

MISSION

1. To Facilitating and exposing the students to various learning opportunities through dedicated academic guidance and monitoring.
2. To Provide a learning ambience to encourage innovations, problem solving skills, leadership qualities, team-spirit, entrepreneurship skills and ethical responsibilities.
3. To Encourage faculty and students to actively participate in innovation, industry solutions, research and lifelong learning so that their contribution makes a substantial difference to the society.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

- Graduates will be equipped to be employed in IT industries and be engaged in learing understanding and applying new ideas.
- Graduates through academic training will be able to take up higher studies and industry specific research programs.
- Graduates will be responsible computing professional with social obligations in their own area of interest.

PROGRAM SPECIFIC OUTCOMES (PSO's)

- Graduates will be able to use the knowledge and ability to write programs and integrate them with the hardware/software products in the domains of embedded systems, databases/data analytics, network/web systems and mobile products.
- Graduate will be able to use knowledge of information science in various domains to identify research gaps and hence to provide solution to new ideas and innovations.

PROGRAM OUTCOMES

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

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

6. **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 considerations.

7. **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.

8. **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.

9. **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.

10. **Environment and sustainability**: Understand the impact of the professional engineering solutionsin societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

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

12. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

13. **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.0

Course Details:

Course Name : Data Structures Laboratory

Course Code : BCSL305

Course prerequisite : Basics of Programming

Course Outcomes	BCSL305 –Data Structure Laboratory
BCSL305 .1	Analyze various linear and non-linear data structures.
BCSL305 .2	Demonstrate the working nature of different types of data structuresand their applications.
BCSL305 .3	Use appropriate searching and sorting algorithms for the givescenario.
BCSL305 .4	Apply the appropriate data structure for solving real world problems.

Course Outcomes & CO-PO-PSO Mapping and Justification

Title of the Course: DATA STRUCTURES LABORATORY

Semester: III–CSE **Year2023-24**

Course Code : BCSL305	**I.A Marks :** 50
Hours/week (L-T-P):0-0-2	**Exam Hrs. :** 03
Total Hours : 28 Hours of Practicals	**Marks (Min/Max):**38/100 **VTU Exam:**35/100 **Internal Assessment:**20/50
Course Plan Author: Mr. Basavaraj Neelagund	**Date: 02/11/2023**
Checked By: Sayeesh	**Date**: 10/11/2023

Prerequisites: Basics of Programming
Co-requisites: Knowledge of Memory Management.
Relevance of the Course: Programming
Application Area: 1.Software Development

Objectives:

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

CLO1: Dynamic memory management

CLO2: Linear data structures and their applications such as stacks, queues and lists

CLO3: Non-Linear data structures and their applications such as trees and graphs

Course outcome (Course Skill Set)

At the end of the course the student will be able to:

BCSL305.1: Analyze various linear and non-linear data structures

BCSL305.2: Demonstrate the working nature of different types of data structures and their applications

BCSL305.3: Use appropriate searching and sorting algorithms for the give scenario.

BCSL305.4: Apply the appropriate data structure for solving real world problems

CORRELATION LEVELS: Correlation Levels:

Slight (low) =1, Moderate (Medium) =2, Substantial (High)=3.

CO-PO MATRIX:

PO	CO1	CO2	CO3	CO4	COURSE
PO1	3	3	3	3	**3**
PO2	2	2	2	2	**2**
PO3	-	2	2	1	**2**
PO4	-	-	-	1	**1**
PO5	-	-	-	-	**-**
PO6	-	-	-	-	**-**
PO7	-	-	-	-	**-**
PO8	-	-	-	-	**-**
PO9	-	-	-	-	**-**
PO10	-	-	-	-	**-**
PO11	-	-	-	-	**-**
PO12	1	1	1	1	**1**
Justification	**PO1:** Strongly mapped as knowledge in fundamental programming methodologies help students to solve the problems. **PO2:** Moderately mapped as students will apply linear and non linear	**PO1:** Strongly mapped as knowledge in different Data structures help students to solve the problems. **PO2:** Moderately mapped as students will apply Data	**PO1:** Strongly mapped as knowledge in Searching and Sorting help students to solve the problems. **PO2:** Moderately mapped as students will apply Different	**PO1:** Strongly mapped as knowledge in Data Structures helps students to solve the problems. **PO2:** Moderately mapped as students will apply	**PO1:** Strongly mapped as knowledge in fundamental programming methodologies help students to solve the problems **PO2:** Moderately mapped as students will apply Data structures to create,

	concepts to analyze complex Engineering Problems **PO12:**Slightly mapped as Information acquired from the fundamentals of Data Structures provides lifelong learning in the context of technological change	structures to create, analyze complex Engineering Problems **PO3:** Moderately mapped as students will apply Data structures to Design complex Engineering Problems **PO12:** Slightly mapped as Information acquired from Different Data Structures provides lifelong learning in the context of technological change	Searching and Sorting Algorithms to analyze complex Engineering Problems **PO3:** Moderately mapped as students will apply Different Searching and Sorting Algorithms to Design complex Engineering Problems **PO12:** Slightly mapped as Information acquired from Searching and Sorting Techniques provides lifelong learning in the context of technological change	appropriate Data structure to analyze complex Engineering Problems **PO3:** Slightly mapped as students will apply appropriate Data structure to Design complex Engineering Problems **PO4:** Slightly mapped as students will apply appropriate Data structure to research and Design complex Engineering Problems **PO12**: Slightly mapped as Information acquired from Data structures	analyze complex Engineering Problems **PO3:** Slightly mapped as students will apply Different Searching and Sorting Algorithms to Design complex Engineering Problems **PO4:** Slightly mapped as students will apply appropriate Data structure to research and Design complex Engineering Problems **PO12**: Slightly mapped as Information acquired from Data structures Techniques provides lifelong learning in the context of technological change

				Techniques provides lifelong learning in the context of technological change	

CO-PSO MATRIX:

PSO	**CO1**	**CO2**	**CO3**	**CO4**	**COURSE**
PSO1	1	1	1	1	**1**
PSO2	-	-	-	-	**-**
Justification	**PSO1:**Slightly mapped as students will have the knowledge in programming methodologies help in designing solutions and analyzing its complexity	**PSO1:**Slightly mapped as students will have the knowledge of Different Data Structures help in designing solutions and analyzing its complexity	**PSO1:**Slightly mapped as students will have the knowledge of Searching and Sorting Techniques help in designing solutions and analyzing its complexity	**PSO1:**Slightly mapped as students will have the knowledge of Hashing Techniques help in designing solutions and analyzing its complexity	**PSO1:**Slightly mapped as students will have the knowledge of data structures help in designing solutions and analyzing its complexity

COURSE-COORDINATOR **HOD**

SYLLABUS

DATA STRUCTURES AND ITS APPLICATION LABORATORY			Semester:3
Course Code	BCSL305	CIE Marks	**50**
Teaching Hours/Week (L:T:P)	**0:0:2**	SEE Marks	**50**
Total Hours of Pedagogy	28	Exam Hours	**3**

CREDIT-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:

Program 01: 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 02: 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 03: 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 04: 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 05: 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 06: 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 07: 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
Program 08: 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 09: 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) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-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
Program 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 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 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

TABLE OF CONTENT

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.	22-26
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	27-34
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	35-42
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) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3	43-49

	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	50-54
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.	55-57
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.	58-63

Program 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 reporton screen.

```
/*****************************************************************
*File          : 01_Calendar.c
*Description   : Calender Operations
*Author        : Dept of CSE,YIT
*Compiler      : gcc compiler
*Date          : 04 December 2023
*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
// Structure to represent a day in the calendar
struct Day
{
    char *dayName; // Dynamically allocated string for the day name
    int date;
    char *activity;//Dynamically allocated string for the activitydescription
};
// Function to create a day
void create(struct Day *day)
 { // Allocate memory for the day name and activity
 day -> dayName = (char * )malloc(sizeof(char) * 20); // Assuming day namesare
 less than 20 characters
 day -> activity = (char * )malloc(sizeof(char) * 100); // Assuming activity
 descriptions are less than 100 characters
 // Input the day details
 printf("Enter the day name:");
 scanf("%s",day>dayName);
 printf("Enter the date:");
 scanf("%d", & day-> date);
 printf("Enter the activity for the day:");
 scanf(" %s", day -> activity); // Read the entire line, including spaces
```

```
  }
  // Function to read data from the keyboard and create the calendar
void read(struct Day *calendar, int size)
{
for (int i = 0; i < size; i++)
    {
     printf("Enter details for Day %d:\n", i + 1);
     create( & calendar[i]);
    }
}
  // Function to display the calendar
void display(struct Day * calendar, int size)
   {
     printf("\nWeek's Activity Details:\n")
     ;for (int i = 0; i < size; i++)
        {
           printf("Day %d:\n", i + 1);
           printf("Day Name:%s\n",calendar[i].dayName);
           printf("Date: %d\n", calendar[i].date);
           printf("Activity: %s\n", calendar[i].activity);
           printf("\n");
        }
}
  // Function to free the dynamically allocated memory
void freeMemory(struct Day * calendar, int size)
{
                for (int i = 0; i < size; i++)
                free(calendar[i].dayName);
                free(calendar[i].activity);
          }
}
int main()
{
          int size;
          printf("Enter the number of days in the week:");
          scanf("%d", & size);
          // Dynamically allocate memory for the calendar
```

```
        struct Day *calendar = (struct Day * ) malloc(sizeof(struct Day) * size);
        // Check if memory allocation is successful
if (calendar == NULL)
   {
     printf("Memory allocation failed. Exiting program.\n");
    return 1;
   }
  // Read and display the calendar
  read(calendar, size);

  display(calendar, size);

  // Free the dynamically allocated memory
freeMemory(calendar, size);
  // Free the memory allocated for the calendar arrayfree(calendar);

return 0;
}
```

----------OUTPUT----------

```
Enter the number of days in the week: 7
Enter details for Day 1:
Enter the day name: Sunday
Enter the date: 1
Enter the activity for the day: Learning

Enter details for Day 2: Enter
the day name: Monday
Enter the date: 2
Enter the activity for the day: Coding

Enter details for Day 3:
Enter the day name: Tuesday
Enter the date: 3
Enter the activity for the day: Testing

Enter details for Day 4:
Enter the day name: Wednesday
Enter the date: 4
Enter the activity for the day: Debugging
```

Enter details for Day 5:
Enter the day name: Thrusday
Enter the date: 5
Enter the activity for the day: Publishing

Enter details for Day 6:
Enter the day name: Friday
Enter the date: 6
Enter the activity for the day: Marketing

Enter details for Day 7:
Enter the day name: Saturday
Enter the date: 7
Enter the activity for the day: Earning

Week's Activity
Details:Day 1:
Day Name: Sunday
Date: 1
Activity: Learning

Day 2:
Day Name: Monday
Date: 2
Activity: Coding
Day 3:
Day Name: Tuesday
Date: 3
Activity: Testing

Day 4:
Day Name:Wednesday
Date: 4
Activity:Debugging

Day 5:
Day Name: Thrusday
Date: 5
Activity: Publishing

Day 6:
Day Name: Friday
Date: 6
Activity: Marketing

Day 7:
Day Name: Saturday
Date: 7
Activity: Earning

Program 02: Design, Develop and Implement a Program in C for thefollowing operations on Strings

1. **Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)**
2. **Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages incase PAT does not exist in STR**

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

```
/*****************************************************************************
*File          : 02_String.c
*Description: String Operations
*Author        :Dept of CSE,YIT
*Compiler      : gcc compiler
*Date          : 1 January 2024
*****************************************************************************/
#include<stdio.h>
char STR[100],PAT[100],REP[100],ANS[100];
int i,j,c,m,k,flag=0;
void read()
{
     printf("\n Enter MAIN string:\n");gets(STR);
     printf("\n     Enter     PATTERN     string:\n");
     gets(PAT);
     printf("\n     Enter     REPLACE     string\n");
     gets(REP);
}
void replace()
{
    i=m=c=j=0;
    while(STR[c]!='\0')
    {
      if(STR[m]==PAT[i])
       i++;
       m++;
         if(PAT[i]=='\0')
           {
            for(k=0;REP[k]!='\0';k++,j++)
            ANS[j]=REP[k];
            i=0;
            c=m;
```

```
            flag=1;
           }
       }
       else
       {
        ANS[j]=STR[c];
        j++;c++;m=c;i=0;
       }
    } // while
if(flag==0)
   printf("pattern doesen't found!!!\n");
   else
    {
      ANS[j]='\0';
      printf("\n The RESULTANT string is %s \n",ANS);
    }
}// replace
void main()
{
   read();
   replace();
}
```

----------OUTPUT----------

```
Enter MAIN string:
   Engg is so fun
Enter PATTERN string:
    fun
Enter   REPLACE   string
 tough
The RESULTANT string is Engg is so tough
```

----------OUTPUT----------

```
Enter MAIN string:
Ram is a god
Enter PATTERN string:
Rama
Enter   REPLACE   string
Ravana
pattern doesen't found
```

Program 03: Stack of Integers

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

a. Push an Element on to Stack

b. Pop an Element from Stack

c. Demonstrate Overflow and Underflow situations on Stack

d. Display the status of Stack

e. Exit

Support the program with appropriate functions for each of the aboveoperations

```
/*****************************************************************************
        *File   : 03_Stack.c
        *Description: Stack Operations
        *Author      : Dept of CSE,YIT
        *Compiler   : gcc compiler
        *Date   : 15 December 2023
 *****************************************************************************
  #include <stdio.h>
  #include <conio.h>
  #include <math.h>
  #define max 5
  int s[max],stop;
  int ele,stk[max],sp,ch;

  void push(int ele,int s[],int *stop)
  {
     if(*stop>=max-1)
     printf("stock overflow\n");else
     s[++*stop]=ele;
  }

  int pop(int s[],int *top)
  {
     if(*top==-1)
      {
         printf("stack empty | underflow\n");return
          0;
        }
      else
```

```
        return(s[(*top)--]);
 }

void palindrome(int ele,int st[])
{
   int rem,rev=0,temp=ele,i=0;while(temp!=0)
      {
        rem=temp%10;
        push(rem,st,&sp);
        temp=temp/10;
      }

      while(sp!=-1)
          rev=rev+(pop(st,&sp)*pow(10,i++));

       if(ele==rev)
            printf("palendrome\n");
       else
            printf("not a palindrome\n");
}


void display(int s[],int *stop)
{
    int i;
    if(*stop==-1)
         printf("stalk is empty\n");
    else
          for(i=*stop;i>-1;i--)
              printf("%d\n",s[i]);
}


void main()
 {
      stop= -1,sp= -1;while(1)
         {
            printf("Enter the choice:\n");
            printf("..........\n");
            printf("Enter 1 to insert an element into the STACK\n");
            printf("Enter 2 to delete an element from the STACK\n");
            printf("Enter 3 to check an element is palindrome or not\n");
            printf("Enter 4 to check the status of the STACK\n"); printf("Enter
            5 to exit\n");
```

```
        printf("----------\n");
        scanf("%d",&ch);switch(ch)
            {
              case 1:printf("Enter the element to de inserted to STACK\n");scanf("%d",&ele);
                     push(ele,s,&stop); break;
              case 2:ele=pop(s,&stop);
                        if(ele!=0)
                            printf("element poped is %d\n",ele);break;
              case 3:printf("Enter the elements to chech weather it is apalindrome\n");
                     scanf("%d",&ele); palindrome(ele,stk);
                     break;
              case 4:printf("the status of the STACK \n");display(s,&stop);
                     break; case 5:exit(0);
            }
        }
}
```

----------OUTPUT----------

```
Enter the choice:
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or  not
Enter 4 to check the status of the STACK
Enter 5 to exit
--------------------------------------------------------------------------------
Enter the choice:
1
Enter the element to de inserted to  STACK10
Enter 1 to insert an element into  the  STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
```

Enter 4 to check the status of the STACK
Enter 5 to exit

Enter the choice:
1
Enter the element to de inserted to STACK20

Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit

Enter the choice:
1
Enter the element to de inserted to STACK30

Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit

Enter the choice:
1
Enter the element to de inserted to STACK40
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or notEnter
4 to check the status of the STACK
Enter 5 to exit

Enter the choice:
1
Enter the element to de inserted to STACK 50
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK

```
Enter 3 to check an element is palindrome or  not
Enter 4 to check the status of the STACK
Enter 5 to exit
---------------------------------------------------------------------------

Enter the choice:

1
Enter the element to de inserted to STACK60
stock overflow
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or  not
Enter 4 to check the status of the STACK
Enter 5 to exit
---------------------------------------------------------------------------

Enter the choice:

3
Enter the elements to check weather it is a palindrome121
palindrome
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or  not
Enter 4 to check the status of the STACK
Enter 5 to exit
---------------------------------------------------------------------------

Enter the choice:

3
Enter the elements to chech weather it is a palindrome12345
not a palindrome
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or  not
Enter 4 to check the status of the STACK
Enter 5 to exit
---------------------------------------------------------------------------
```

```
Enter the choice:
2
element poped is 50
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit
--------------------------------------------------------------------------------
Enter the choice:
2
element poped is 40
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit
--------------------------------------------------------------------------------
Enter the choice:
4
the status of the STACK
30
20
10
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit
--------------------------------------------------------------------------------
Enter the choice:
2
element poped is 30
Enter 1 to insert an element into the STACK
Enter 2 to delete an element from the STACK
Enter 3 to check an element is palindrome or not
Enter 4 to check the status of the STACK
Enter 5 to exit
```

Enter the choice:

2

element poped is 20

Enter 1 to insert an element into the STACK

Enter 2 to delete an element from the STACK

Enter 3 to check an element is palindrome or not

Enter 4 to check the status of the STACK

Enter 5 to exit

Enter the choice:

2

element poped is 10

Enter 1 to insert an element into the STACK

Enter 2 to delete an element from the STACK

Enter 3 to check an element is palindrome or not

Enter 4 to check the status of the STACK

Enter 5 to exit

Enter the choice:

2

stack empty | underflow

Enter 1 to insert an element into the STACK

Enter 2 to delete an element from the STACK

Enter 3 to check an element is palindrome or not

Enter 4 to check the status of the STACK

Enter 5 to exit

Program 04: 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.

```
/*****************************************************************************
*File            : 04_Infix.c
*Description: infix to postfix
*Author          : Dept of CSE,YIT
*Compiler        : gcc compiler
*Date            : 30 December 2023
*****************************************************************************/
#include<stdio.h>
#include<string.h>

int F(char symbol)
{
   switch(symbol)
    {
     case '+':
     case  '-': return 2;
     case '*':
     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 '%':
        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];
        char 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--];
             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 postfix expression is\n");
printf("%s\n",postfix);
}
```

----------**OUTPUT**----------

Enter a valid infix expression
A+B
The postfix expression is
AB+

----------**OUTPUT**----------

Enter a valid infix expression
((A+B)/(C-D))
The postfix expression is
AB+CD-/

----------**OUTPUT**----------

Enter a valid infix expression
(A+B-C^D*E/F)
The postfix expression is
AB+CD^E*F/-

Program 05: 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

```
/*****************************************************************************
*File           : 05_Evaluation_Tower.c
*Description: Evaluation of Expression and Tower of Hanoi
*Author         : Dept of CSE,YIT
*Compiler       : gcc compiler
*Date           : 08 January 2024
*****************************************************************************/
#include<stdio.h>
#include<string.h>
#include<math.h>
int count=0, top=-1;
   int operate(char symb, int op1, int op2)
    {
        switch(symb)
          {
           case '+': return op1+op2;
           case '-': return op1-op2;
           case '/': return op1/op2;
           case '*': return op1*op2;
           case '%': return op1%op2;
           case '^':return pow(op1,op2);
          }
    }
   void push(int stack[],int d)
    {
       stack[++top]=d;
    }
   int pop(int stack[])
    {
      return(stack[top--]);
    }
void tower( int n,char src, char int r, char des)
    {
        if(n)
```

```
        {
           tower(n-1,src,des,intr);
           printf("disk %d moved from %c to %c\n",n,src,des);
            count++;
           tower(n-1,intr,src,des);
         }
   }
void main()
{
     int   n,   choice,i,op1,op2,ans,stack[50];
     char expr[20],symb;
while(1)
{
 printf("\n Program to perform evaluation of suffix expression and tower of hanoi problem\n");
printf("\n1.Evaluate suffix expression\n2.Tower of hanoi\n3.Exit\n");
printf("\n Enter the choice\n");
  scanf("%d",&choice);
    switch(choice)
     {
      case 1: printf("Enter the suffix expression :");
                scanf("%s",expr);
                   for(i=0;expr[i]!= '\0';i++)
                       {
                          symb=expr[i];
                            if(symb>='0' && symb<='9')
                                push(stack, symb-'0');
                              else
                                    {
                                          op2=pop(stack); op1=pop(stack);
                                           printf("given expr is %d %d %c\n",op2,op1,symb);
                                           ans=operate(symb,op1,op2);
                                             push(stack,ans);
 }
                                        }
                    ans=pop(stack);
                    printf("The result of the suffix expression is %d",ans);
```

```
                break;
        case    2:  printf("Enter     the     number     of     disks\n");
                scanf("%d",&n);
                tower(n,'a','b','c');
                printf("Number of moves taken to move disks from source todestination
                 %d",count);
                 break;
          case 3: return;
         }
       }
  }
```

----------OUTPUT----------

```
Program to perform evaluation of suffix expression and tower of hanoi problem
1.Evaluate     suffix     expression
2.Tower of hanoi
3.Exit

enter the choice1
Enter the suffix expression : 345+*
given expr is 5 4 +
given expr is 9 3 *
The result of the suffix expression is 27
Program to perform evaluation of suffix expression and tower of hanoi problem
1.Evaluate suffix expression
2.Tower    of    hanoi
3.Exit

Enter the choice2
Enter the number of disks3
disk 1 moved from a  to c
disk 2 moved from a to b disk
1 moved from c to b disk 3
moved from a to c disk 1
moved from b to a disk 2
moved from b to c disk  1
moved from a to c
```

Number of moves taken to move disks from source to destination 7
Program to perform evaluation of suffix expression and tower of hanoi problem
1.evaluate suffix expression
2.Tower of hanoi
3.Exit

Enter the choice 3

Program 06: 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 aboveoperations

```
/*****************************************************************************
*File           : 06_Circular_Queue.c
*Description: Operations_of_Circular_Queue
*Author         : Dept of CSE,YIT
*Compiler       : gcc compiler
*Date           : 08 January 2024
*****************************************************************************/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAX 5
int front=0,rear=-1,count=0,item_deleted,element;
charcqueue[MAX];
void insert();
void delete();
void display();
void main()
  {
     int choice;
     while(1)
     {
      printf("\n\nProgram to illustrate operations on CIRCULAR QUEUE of
characters\n");
printf("\n\t1.Insert an element on to CIRCULAR QUEUE\n\t2.Deletean element from
CIRCULAR QUEUE\n\t3.Display the status of CIRCULAR QUEUE\n\t4.Exit\n");
 printf("\nEnter your choice: ");
            scanf("%d",&choice);
               switch(choice)
               {
               case   1:insert();
                     break;
```

```
                case   2:delete();
                       break;
                case   3:display();
                       break;
                case 4:return;
                }
        }
}


void insert()
{
     if(count==MAX)
     {
        printf("CIRCULAR QUEUE is full,elements can not be inserted\n");
        return;
     }
     rear=(rear+1)%MAX;
     printf("\n Enter the element to be inserted into the CIRCULAR QUEUE\n");
     scanf("%d",&element);
     cqueue[rear]=element;
     count++;
}

void delete()
{
     if(count==0)
       {
       printf("CIRCULAR QHEUE is empty,no element  to  delete\n");
         return;
       }
       item_deleted=cqueue[front];
       printf("the element deleted is %d\n",item_deleted);
       front=(front+1)%MAX;
       count-=1;
}
void display()
{
Int i,f;
```

```
if(count==0)
      {
         printf("CIRCULAR QUEUE is empty , no element to display\n");
         return;
      }
      printf("CIRCULAR QUEUE contants are\n");
      for(i=0,f=front;i<count;i++)
      {
       printf("%d\t",cqueue[f]);
       f=(f+1)%MAX;
      }
}
```

----------OUTPUT----------

Program to illustrate operations on CIRCULAR QUEUE of characters

1. Insert an element on to CIRCULAR QUEUE
2. Delete an element from CIRCULAR QUEUE
3. Display the status of CIRCULAR QUEUE
4. Exit

Enter your choice: 1

Enter the element to be inserted into the CIRCULAR QUEUE10

Program to illustrate operations on CIRCULAR QUEUE of characters

1. Insert an element on to CIRCULAR QUEUE
2. Delete an element from CIRCULAR QUEUE
3. Display the status of CIRCULAR QUEUE
4. Exit

Enter your choice: 1

Enter the element to be inserted into the CIRCULAR QUEUE20

Program to illustrate operations on CIRCULAR QUEUE of characters

1. Insert an element on to CIRCULAR QUEUE
2. Delete an element from CIRCULAR QUEUE
3. Display the status of CIRCULAR QUEUE
4. Exit

Enter your choice: 1

Enter the element to be inserted into the CIRCULAR QUEUE30

Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 1
Enter the element to be inserted into the CIRCULAR QUEUE40

Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 1

Enter the element to be inserted into the CIRCULAR QUEUE50

Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 3
CIRCULAR QUEUE contants are
10 20 30 40 50

Program to illustrate operations on CIRCULAR QUEUE of characters

1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR
QUEUE4.Exit
Enter your choice: 2
The element deleted is 10

Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit

Enter your choice: 2
The element deleted is 20
Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 2
The element deleted is 30
Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 2
The element deleted is 40
Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 2
The element deleted is 50
Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit
Enter your choice: 2
CIRCULAR QUEUE is empty,no element to delete
Program to illustrate operations on CIRCULAR QUEUE of characters
1.Insert an element on to CIRCULAR QUEUE
2.Delete an element from CIRCULAR QUEUE
3.Display the status of CIRCULAR QUEUE
4.Exit

Program 07: Develop a menu driven Program in C for the followingoperations 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

```
/******************************************************************************
*File           : 07_Singly_LinkedList.c
*Description    : Operations_of_Circular_Queue
*Author         : Dept of CSE,YIT
*Compiler       : gcc compiler
*Date           : 15 January 2024
******************************************************************************/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
    {
    char usn[11];
    char name[20];
    char branch[20];
    int semester;
    char phone[20];
 }STUDENT;
struct node
    {
    char    usn[11];
    char  name[20];
    char branch[20];
    int semester;
    char phone[20];
    struct node *link;
    };
    typedef struct node*NODE;
 NODE getnode()
{
```

```
        NODE x;
        x=(NODE)malloc(sizeof(structnode));
        if(x==NULL)
        {
      printf("out of memory\n");exit(0);
      }
      return x;
      }

      NODE insert_front(STUDENT item,NODE first)
      {
NODE temp;
temp=getnode();
        strcpy(temp->usn,item.usn);
        strcpy(temp->name,item.name);
        strcpy(temp->branch,item.branch);
        temp->semester=item.semester;
        strcpy(temp->phone,item.phone);
        temp->link=NULL;
      if(first==NULL)
      return temp;
        temp-
        >link=first;
        return temp;
      }
      NODE insert_rear(STUDENT item,NODE first)
      {
NODE temp,cur;
temp=getnode();
      strcpy(temp->usn,item.usn);
      strcpy(temp->name,item.name);
      strcpy(temp->branch,item.branch);
      temp->semester=item.semester;
      strcpy(temp->phone,item.phone);
      temp->link=NULL;
      if(first==NULL)
      return temp;
     cur=first;
```

```
 while(cur->link!=NULL)
    {
    cur=cur->link;
    }
  cur->link=temp;
  return first;
  }

NODE delete_front(NODE first)
  {
NODE temp;
if(first==NULL)
    {
    printf("student list is empty\n");return
    NULL;
    }
  temp=first;
  temp=temp->link;
  printf("delete student record:USN=%s\n",first->usn);
  free(first);
  return temp;
  }
NODE delete_rear(NODE first)
  {
NODE cur,prev;

if(first==NULL)

  {
  printf("student list is empty cannot delete\n");
  return first;
  }
  if(first->link==NULL)
  {
  printf("delete student record:USN=%s\n",first->usn);
  free(first);
  return NULL;
  }
  prev=NUL;
  cur=first;
  while(cur->link!=NULL)
  {
```

```
        prev=cur;
        cur=cur->link;
        }
        printf("delete student record:USN=%s\n",cur->usn);
        free(cur);
        prev->link=NULL;
        return first;
        }
void display(NODE first)
        {
        NODE    cur;
        int count=0;
        if(first==NULL)
        {
        printf("student list is empty\n");
        return;
        }
        cur=first;
        while(cur!=NULL)
        {
        printf("%s\t%s\t%s\t%d\t%s\t\n",cur->usn,cur->name,cur->branch,cur-
  >semester,cur->phone);
        cur=cur->link;
        count++;
        }
        printf("number of students=%d\n",count);
        }
       void main()
        {
        NODE first;
        int choice;
        STUDENT item;
        first=NULL;
        for(;;)
        {
printf("1.insert_front\n2.insert_rear\n3.delete_front\n4.delete_rear\n5.display
\n6.exit\n");
        printf("Enter the choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
```

```
            printf("USN :");
            scanf("%s",item.usn);
            printf("name :");
            scanf("%s",item.name);
            printf("branch:");
            scanf("%s",item.branch);
            printf("semester:");
            scanf("%d",&item.semester);
            printf("phone :");
            scanf("%s",item.phone);
            first=insert_front(item,first);
            break;
      case 2:
            printf("USN :");
            scanf("%s",item.usn);
            printf("name :");
            scanf("%s",item.name);
            printf("branch                    :");
            scanf("%s",item.branch);
            printf("semester:");
            scanf("%d",&item.semester);
            printf("phone :");
            scanf("%s",item.phone);
            first=insert_rear(item,first);
            break;
      case 3:
            first=delete_front(first);
            break;
      case 4:
            first=delete_rear(first);
            break;
      case 5:
            display(first);
            break;
      default:
            exit(0);
      }
    }
}
```

----------OUTPUT----------

```
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display 6.exit
Enter the choice:1
USN :4DM22CS001
name :Abhi
branch:CSE
semester:3
phone:9900123456
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
6.exit
Enter the choice:2
USN :4DM22ME002
name:Sriram
branch:ME
semester:5
phone:9912909012
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
6.exit
Enter the choice:5
  4DM22CS001      Abhi    CSE  3       9900123456
  4DM22ME002      Sriram  ME       5       9912909012
number of students=2
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
```

```
6.exit
Enter the choice:1
USN :4DM22EE001
name :Praveen
branch:EEE
semester:2
phone:9880789789
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
6.exit

Enter the choice:5
  4DM22EE001    Praveen EEE    2    9880789789
  4DM22CS001    Abhi CSE       3    9900123456
  4DM22ME002    Sriram ME      5    9912909012
number of students=3
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
6.exit

Enter the choice:3
delete student record:USN=4DM22EE001
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display 6.exit

Enter the choice:4
delete student record:USN=4DM22ME002
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
```

5.display
6.exit
Enter the choice:5
4DM22CS001 Abhi CSE 3 9900123456

Number of students=1
1.insert_front
2.insert_rear
3.delete_front
4.delete_rear
5.display
6. exit
Enter the choice:

Program 08: 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

```
/*****************************************************************************
*File            : 08_Doubly_LinkedList.c
*Description     : Operations_of_Doubly_Linked_List
*Author          : Dept of CSE,YIT
*Compiler        : gcc compiler
*Date            : 22 January 2024
*****************************************************************************
******/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
{
        int ssn;
        char name[20];
        char    department[20];
        char    designation[20];
        float salary;
        char phone[20];
}  EMPLOYEE;
struct node
{
        int ssn;
        char name[20];
        char    department[20];
        char    designation[20];
        float salary;
        char phone[20];
        struct node *llink;
        struct node *rlink;
```

```
};
typedef struct node* NODE;
NODE getnode()
{
     NODE x;
     x = ( NODE ) malloc(sizeof(struct node)); /* allocate the memory space */
     if ( x == NULL ) /* Free nodes don't exist */
          {
            printf("Out of memory\n"); /* Allocation failed */
            exit(0); /* Terminate the program */
          }
      return x; /* allocation successful */
}


// Insert node at the front end
NODE insert_front(EMPLOYEE emp, NODE first)
{
    NODE temp;
    temp = getnode(); /*obtain a node from OS */
    temp->ssn = emp.ssn; /* Insert various items into new node */
    strcpy(temp->name, emp.name);
    strcpy(temp->department,emp.department);
    strcpy(temp->designation,emp.designation);
    temp->salary = emp.salary;
    strcpy(temp->phone,emp.phone);
    temp->llink = temp->rlink = NULL;
          if (first == NULL)
             return temp; /* Insert a node for the first time */
             temp->rlink = first; /* Insert at the beginning of existing list */
             first->llink = temp;
             return temp; /* return address of new first node */
}

// Inset node at the rear end
NODE insert_rear(EMPLOYEE emp, NODE first)
{
    NODE temp, cur;
    temp = getnode(); /*obtain a node from OS */
```

```
    temp->ssn = emp.ssn; /* Insert various items into new node */
    strcpy(temp->name, emp.name);
    strcpy(temp->department,emp.department);
    strcpy(temp->designation,emp.designation);
    temp->salary = emp.salary;
    strcpy(temp->phone,      emp.phone);
    temp->llink = temp->rlink = NULL;
          if (first == NULL)
             return temp; /* Insert a node for the first time */
             cur = first; /* Get the address of the first node */
          while (cur->rlink != NULL) /* Find the address of the last node */
          {
            cur = cur->rlink;
          }
  /* Insert the node at the end */
    cur->rlink = temp;
    temp->llink = cur;
  /* return address of the first node */
    return first;
}
// Delete node at the front end
NODE delete_front(NODE first)
{
    NODE second;
      if ( first == NULL ) /* Check for empty list */
      {
         printf("employee list is empty \n");
         return NULL; // We can replace NULL with first also
      }
if (first ->rlink == NULL) /* Delete if there is only one node */
     {

         printf("Employee details deleted: ssn=%d\n", first->ssn);
         free(first);
         return NULL;
      }
second = first->rlink; /* Get the address of second node */
 second->llink  = NULL; /*  Make second  node  as the first node  */
printf("Employee details deleted: ssn=%d\n", first->ssn);
```

```
 free(first); /* Delete the first node */
return second;
}
// Delete node at rear end NODE
delete_rear(NODE first)
{
    NODE cur, prev;
      if (first == NULL) /* Check for empty list */
       {
         printf("List is empty cannot delete\n");
         return first;
        }
if (first->rlink == NULL ) /*Only one node is present and delete it */
{
     printf("Employee details deleted: ssn= %d\n",first->ssn);
     free(first); /* return to availability list */
     return NULL; /* List is empty so return NULL */
}
/* Obtain address of the last node and just previous to that */
prev = NULL;
cur = first;
while(cur->rlink != NULL )
   {
     prev = cur;
     cur = cur->rlink;
   }
printf("Employee details deleted: ssn=%d\n",cur->ssn);
prev->rlink = NULL; /* Make last but one node as the last node */free(cur);
/* delete the last node */

return first; /* return address of the first node */
}
// Display employee informationvoid
display(NODE first)
{
    NODE   temp,cur;
    int count = 0;
       if (first == NULL) /* List is empty */
       {
```

```
            printf("employee list is empty\n");
            return;
          }
/* Display employee details */
cur = first;
while (cur != NULL)
    {
          printf("%d %.2f %s %s %s %s\n", cur->ssn, cur->salary, cur-
                >name,cur->department, cur->designation, cur->phone);
                cur = cur->rlink;
                count++;
    }
     printf("Number of employees = %d\n", count);
}
// Main function
void main()

{
for (;;)
{
NODE first;
int choice;
EMPLOYEE item;
first = NULL;
printf("1:Insert_Front 2: Insert_Rear\n");
printf("3:Delete_Front 4: Delete_Rear\n");
printf("5:Display 6: Exit\n");
printf("Enter the choice\n");
scanf("%d", &choice);
switch(choice)
{
```

```
case 1: printf("ssn :");
            scanf("%d",&item.ssn);
            printf("name :");
            scanf("%s",item.name);        printf("department:");
            scanf("%s",item.department);
            printf("designation:");
            scanf("%s",item.designation);       printf("salary:");
            scanf("%f",&item.salary);
            printf("phone :");
            scanf("%s",item.phone);
            first = insert_front(item,first);
            break;
case 2: printf("ssn :");
            scanf("%d",&item.ssn);
            printf("name :");
            scanf("%s",item.name);        printf("department:");
            scanf("%s",item.department);
            printf("designation:");
            scanf("%s",item.designation);       printf("salary:");
            scanf("%f",&item.salary);
            printf("phone :");
            scanf("%s",item.phone);
            first = insert_rear (item,first);
            break;
  case 3: first=delete_front(first);break;
  case 4: first=delete_rear(first);break;
case 5: display(first);
             break;

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

----------OUTPUT----------

```
1:Insert_Front
2: Insert_Rear
3:Delete_Front
4: Delete_Rear
5:Display
```

```
6:Exit
Enter the choice1
ssn:3434 name :Ravi
department:cse designation :asstprof
salary :34000
phone :9900123456

1:Insert_Front
2:  Insert_Rear
3:Delete_Front
4: Delete_Rear
5:Display
6: Exit
Enter the choice2
ssn  :1212
 name:Ramu department  :ise
designation:profsalary :12000
phone :8972345678

1:Insert_Front
2:  Insert_Rear
3:Delete_Front
 4: Delete_Rear
5:Display
6: Exit
Enter the choice5
  3434      34000.00     Ravi     cse          asstprof     9900123456
  1212      12000.00     Ramu    ise           prof         8972345678
Number of employees = 2

1:Insert_Front
2: Insert_Rear
3:Delete_Front
4: Delete_Rear
5:Display
6:Exit
Enter the choice4
Employee details deleted: ssn=1212

1:Insert_Front
2: Insert_Rear
```

```
3:Delete_Front
4: Delete_Rear
5:Display
6:Exit
Enter the choice2
ssn :1234
name :Reetha
department:AIML
designation:Asstprof
salary :45000
phone :9272347899

1:Insert_Front
2:  Insert_Rear
3:Delete_Front
 4: Delete_Rear
5:Display
6:Exit
Enter the choice5
3434  34000.00       Ravi      cse       asstprof       9900123456
1234      45000.00      Reetha     AIML     Asstprof      9272347899 Number of
employees  =  2
1:Insert_Front
2: Insert_Rear
3:Delete_Front
4: Delete_Rear
5:Display
6:Exit
Enter the choice3
Employee details deleted: ssn=3434

1:Insert_Front
2:  Insert_Rear
3:Delete_Front
4: Delete_Rear
5:Display
6:Exit
Enter the choice: 6
```

Program 09: 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) =
$6x^2y^2z4yz^5+3x^3yz+2xy^5z-2xyz^3$
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 aboveoperations

```
/*****************************************************************************
*File            : 09_Polynomial_Operations.c
*Description: Operations_of_Polynomials
*Author          : Dept of CSE,YIT
*Compiler        : gcc compiler
*Date            : 22 January 2024
*****************************************************************************
******/
#include <stdio.h>
#include<stdlib.h>
#include <math.h>

struct node
{
int cf;
int px, py, pz;
struct node *link;
};
typedef struct node* NODE;
```

//create new node

```
NODE getnode()
{
   NODE x;
   x = ( NODE ) malloc(sizeof(struct node)); /* allocate the memory space */
   if ( x == NULL ) /* Free nodes don't exist */
{
    printf("Out of memory\n"); /* Allocation failed */
    exit(0); /* Terminate the program */
          }
      return x; /* allocation successful */
}
```

```
  // Insert new term at rear end
NODE insert_rear(int cf, int px, int py, int pz, NODE head)
{
      NODE temp, cur;
      temp = getnode(); /* create a node */
      temp->cf = cf; /* insert coefficient */
      temp->px = px; /* insert power of x */
      temp->py = py; /* insert power of y */
      temp->pz = pz; /* insert power of z */
      cur = head->link; /* obtain the address of the first  node */
      while (cur->link != head) /* obtain the address of last node */
      {
      cur = cur->link;
      }
cur->link = temp; /* insert the node at the end */
temp->link = head;
return head;
}

  // Read a polynomial
NODE read_poly(NODE head)
{
      int i, n;
      int cf, px, py, pz; /* To hold term of a polynomial */
      printf("Enter the number of terms in the polynomial:");
      scanf("%d", &n);
  for(i = 1; i <= n; i++)
      {
      printf("Enter term: %d\n", i);
      printf("Cf px py pz = ");
      scanf("%d %d %d %d", &cf, &px, &py, &pz); /* Enter each term */
      head = insert_rear (cf, px, py, pz, head); /* insert at the end */
      }
   return head;
}
  // Function to Display the Polynomial
void display(NODE head)
{
```

```
NODE temp;
if ( head->link == head )
{
printf("Polynomial does not exist\n");
return;
}
temp=head->link;
while (temp != head)
{
if (temp->cf < 0) /* Print –ve coefficient */
printf("%d", temp->cf);
else /* Print +ve coefficient */
printf("+%d", temp->cf);
if (temp->px != 0)
printf("x^%d", temp->px);
if (temp->py != 0)
printf("y^%d", temp->py);
if (temp->pz != 0)
printf("z^%d", temp->pz);
temp = temp->link;
}
printf("\n");
}
```

// Function to evaluate the polynomials

```
float evaluate(NODE head)
{
int x, y, z;
float sum = 0;
NODE p;
printf("Enter the value of x, y and z\n");
scanf("%d %d %d", &x, &y, &z);
p = head->link; /* Access each term, substitute x, y and z */
while (p != head )
{
sum += p->cf * pow(x, p->px) * pow(y, p->py) * pow(z, p->pz);
p = p->link;
}
return sum;
```

```
}
  // function to search for term of poly1 in Poly2
NODE search(NODE p1, NODE h2)
{
int cf1, px1, py1, pz1, cf2, px2, py2, pz2;
NODE p2;
/* coefficient power of x power of y power of z */
cf1 = p1->cf;
px1 = p1->px;
py1 = p1->py;
pz1 = p1->pz;
p2 = h2->link;
while ( p2 != h2 )
{
/* coefficient power of x power of y power of z */
cf2 = p2->cf;
px2 = p2->px;
py2 = p2->py;
pz2 = p2->pz;
if ( px1 == px2 && py1 == py2 && pz1 == pz2)
 break;
 p2 = p2->link; // obtain the next term of polynomial 2
}
return p2;
}
  //copy polynomial
NODE copy_poly ( NODE h2, NODE h3)
{
NODE p2;
int cf2, px2, py2, pz2;
p2 = h2->link;
while ( p2 != h2 )
{
/* Add remaining terms of poly 2 into poly 3);
if (p2->cf != -999)
{
cf2 = p2->cf, px2 = p2->px, py2 = p2->py, pz2 = p2->pz;
```

```
h3 = insert_rear (cf2, px2, py2, pz2, h3);
}
p2 = p2->link; /* Get the next term of polynomial 2 */
}
return h3;
}
```

// function to add two polynomials

```
NODE add_poly (NODE h1, NODE h2, NODE h3)
{
NODE p1, p2;
int cf1, px1, py1, pz1, sum;
p1 = h1->link;
while (p1 != h1) /* As long as term of polynomial 1 exists */
{
/* coeeficent power of x power of y power of z of poly1 */
cf1 = p1->cf, px1 = p1->px, py1 = p1->py, pz1 = p1->pz;
p2 = search(p1, h2); /* search power of p1 in p2 */
if (p2 != h2) /* powers of poly1 found in poly 2 */
{
sum = cf1 + p2->cf; /*Add coefficients, insert to poly3*/
h3 = insert_rear (sum, px1, py1, pz1, h3);
p2->cf = -999; /* Delete the term of poly2 */
}
else /* If not found, insert term of poly 1 to poly 3*/
h3 = insert_rear (cf1, px1, py1, pz1, h3);
p1 = p1->link; /* Get the next term of polynomial 1 */
}
h3 = copy_poly(h2, h3); /* Copy remaining terms of poly 2 into poly 3*/
return h3; /* return total terms in poly 3 */
}
void main()
{
    NODE head,h1,h2,h3;
    float res;
   int choice;
  for (;;)
{
    printf("1:To Evvaluate the Polynomial 2: To add two Polynomials\n");
```

```
        printf("3:Exit\n");
        printf("Enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1: head = getnode();
                head->link = head;
                printf("Enter the polynomial\n");
                head = read_poly(head);
                res = evaluate(head);
                printf("The given polynomial is\n");
                display(head);
               printf("The result = %f\n", res);
               break;
        case 2: h1 = getnode();
                h2 = getnode();
                h3 = getnode();
                h1->link = h1;
                h2->link = h2;
                h3->link = h3;
                printf("Enter the first polynomial\n");
                h1 = read_poly(h1);
                printf("Enter the second polynomial\n");
                h2 = read_poly(h2);
                printf("Poly 1:");
                display(h1);
                printf("Poly 2:");
                display(h2);
                printf("..........\n");
                h3 = add_poly(h1, h2, h3);
                printf("Poly 3:");
                display(h3);
                printf("..........\n");
                break;
        default: exit(0);
         }
      }
}
```

OUTPUT

```
1:To Evaluate the Polynomial 2: To add two Polynomials3:Exit
Enter the choice1
Enter the polynomial
Enter the number of terms in the polynomial:3Enter
term: 1
Cf px py pz = 2 3 4 5

Enter term: 2
Cf px py pz = 3 4 5 6
Enter term: 3
Cf px py pz = 2 3 4 5
Enter the value of x, y and z1 2
3
The given polynomial is
+2x^3y^4z^5+3x^4y^5z^6+2x^3y^4z^5
Theresult = 85536.000000
1:To Evaluate the Polynomial
2: To add two Polynomials
3:ExitEnter the choice 2
Enter the first polynomial
Enter the number of terms in the polynomial:2Enter
term: 1
Cf px py pz = 2 3 0 4
Enter term: 2
Cf px py pz = 3 0 1 2
Enter the second polynomial
Enter the number of terms in the polynomial:3Enter
term: 1
Cf px py pz = 1 2 3 4
Enter term: 2
Cf px py pz = 2 0 0 0
Enter term: 3
Cf px py pz = 3 3 0 2
Poly 1:+2x^3z^4+3y^1z^2
Poly 2:+1x^2y^3z^4+2+3x^3z^2
```

Program 10: Design, Develop and Implement a menu driven Program in Cfor the following operations on Binary Search Tree (BST) of Integers

c. Create a BST of N Integers

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

```
/*************************************************************************
*File            : 10_BinarySearchTree.c
*Description: Applications of Binary Search Tree
*Author          : Dept of CSE,YIT
*Compiler        : gcc compiler
*Date            : 29 January 2024
*************************************************************************/
#include<stdio.h>
#include    <stdlib.h>
#define MAX 20

struct node
{
        int data;
        struct node *lchild, *rchild;
};
typedef  struct  node*  NODE;
NODE tree = NULL;

void CreateBST (int a[MAX], int n)
{
        NODE temp, p,  q;
        int i; for(i=0;i<n;i++)
        {
                temp =(struct node *)malloc(sizeof(struct node*));
                temp->data = a[i];
                temp->lchild = temp->rchild = NULL;
                if(tree == NULL)
                        tree = temp;
                else
                {

                  p = q = tree;
                  while(q !=NULL)
                  {
```

```
                    p=q;
                    if(a[i] < p->data)
                        q = p->lchild;
                    else if(a[i] > p->data)
                        q = p->rchild;
                    else
                    {
                        free(temp);
                        break;
                    }
                }


                if( q == NULL)
                {
  if(a[i] < p->data)
  p->lchild = temp;
else                        p->rchild = temp;



            }
}
}

        printf("Binary Seacrh Tree created\n\n");
}

void Inorder(NODE tree)
{
        if(tree != NULL)
        {
            Inorder(tree->lchild);
            printf("%d",tree->data);
            Inorder(tree->rchild);
        }
}
void Preorder(NODE tree)
{
        if(tree != NULL)
        {
```

```
                printf("%d",tree->data);
                Preorder(tree->lchild);
                Preorder(tree->rchild);
        }
}

void Postorder(NODE tree)
{
        if(tree != NULL)
        {
                Postorder(tree->lchild);
                Postorder(tree->rchild);
                printf("%d",tree->data);
        }
}
int main()
{
        int a[MAX], n, i, choice;
        while(1)
        {
              printf("\n\n**********************MENU*****************");
printf("\n1. Create a BST of n integers\n2. Traverse the BST in Inorder\n3. Traverse the BST in Preorder\n4. Traverse the BST in Postorder\n5. Exit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1 : printf("Enter the number of integers : ");
                                 scanf("%d",&n);
                                 printf("Enter the elements\n");
                                 for(i=0; i<n; i++)
                                 scanf("%d",&a[i]);
                                 CreateBST(a, n);
                                 break;
                        case 2 : printf("Inorder Traversal :\n");
                                  Inorder(tree);
                                  break;
                        case 3 : printf("Preorder Traversal :\n");
```

```
                    Preorder(tree);
                    break;
                case 4 : printf("Postoder Traversal :\n");
                    Postorder(tree);
                    break;
                case 5 : exit(0);
                    break;
            }
        }
}
```

----------OUTPUT----------

************************MENU***********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit

Enter your choice 1

Enter the number of integers : 3
Enter the elements
10 20 5

Binary Seacrh Tree created

************************MENU***********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit

Enter your choice : 2
Inorder Traversal :

5 10 20

************************MENU**********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit
Enter your choice : 3
Preorder Traversal :

10 5 20

************************MENU**********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit
Enter your choice : 4
Postoder Traversal :

5 20 10

************************MENU**********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit
Enter your choice : 4
Postoder Traversal :

5 20 10

************************MENU**********************

1. Create a BST of n integers
2. Traverse the BST in Inorder
3. Traverse the BST in Preorder
4. Traverse the BST in Postorder
5. Exit
Enter your choice : 5

Program 11: Design, Develop and Implement a Program in C for thefollowing operations on Graph(G) of Cities

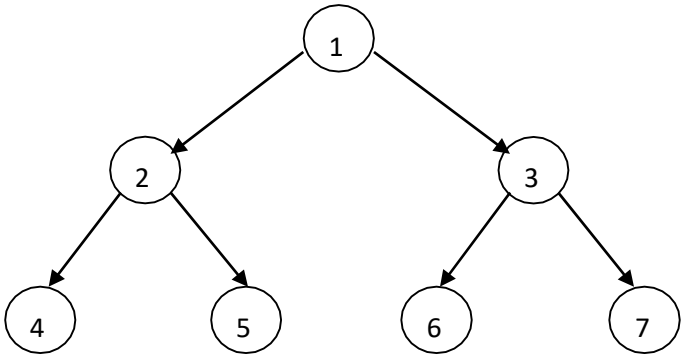
e. Create a Graph of N cities using Adjacency Matrix.

f. Print all the nodes reachable from a given starting node in a digraphusing DFS/BFS method

```
/*************************************************************************
*File           : 11_Graph.c
*Description: Find reachable node using BFS and DFS
*Author         : Dept of CSE,YIT
*Compiler       : gcc compiler
*Date           : 04 February 2024
*************************************************************************/
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

int source;

void DFS(int a[MAX][MAX], int visited[MAX], int s, int n)
{
        int u, v;
        u = s;
        visited[u] = 1;
        if(u != source)
                printf(" %d ", u);
        for(v=1; v<=n; v++)
        {
                if(a[u][v] == 1 && visited[v] == 0)
                        DFS(a,visited, v, n);
        }
}

void BFS(int a[MAX][MAX], int visited[MAX], int source, int n)
{
        int queue[MAX], f=0, r=0, u, v;
        queue[r]=source;
        visited[source] =1;
        while(f<=r)
        {
                u=queue[f++];
                for(v=1; v<=n; v++)
                {
                        if(a[u][v] == 1 && visited [v] == 0)
                        {
```

```
                        printf("%d",v);
                        visited[v]=1;
                        queue[++r] = v;
                    }
                }
            }
        }
int main()
{
        int a[MAX][MAX], visited[MAX], n, choice, i, j, x, y;
        printf("Enter the number of vertices in the graph : ");
        scanf("%d", &n);
        printf("Enter the adjacency matrix for the graph\n");
        for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
        scanf("%d",&a[i][j]);
        printf("Enter the starting node of the graph : ");scanf("%d",&source);
        while(1)
        {
            printf("\n\n**********MENU**********");
            printf("\n1. DFS\n2. BFS\n3. Exit\n");
            printf("Enter your choice : ");
            scanf("%d",&choice);
            switch(choice)
            {
        case 1 : printf("Nodes reachable from %d using DFS method\n", source);
                        for(x=1; x<=n; x++)
                        visited[x]=0;
                        DFS(a, visited, source, n);
                        break;
        case 2 : printf("Nodes reachable from %d using BFS method\n", source);
                        for(y=1; y<=n; y++)
                        visited[y]=0;
                        BFS(a, visited, source, n);
                        break;

        case 3: exit(0);

            }
          }
 }
```

**********OUTPUT**********
******** Consider the following graph**



Enter the number of vertices in the graph : 7Enter
the adjacency matrix for the graph
0 1 1 0 0 0 0
0 0 0 1 1 0 0
0 0 0 0 0 1 1
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
Enter the starting node of the graph: 1
**********MENU**********

1. DFS
2. BFS
3. Exit

Enter your choice : 1
Nodes reachable from 1 using DFS method2
4 5 3 6 7
**********MENU**********

1. DFS
2. BFS
3. Exit

Enter your choice: 2
Nodes reachable from 1 using BFS method2 3
4 5 6 7
**********MENU**********

1. DFS
2. BFS
3. Exit

Enter your choice : 3

Program 12: Design and develop a program in C that uses Hash FunctionH:K->L as H(K)=K mod m(reminder method) and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

```
/*************************************************************************
*File           : 12_HashingTechnique.c
*Description: Implement Hashing Technique
*Author         : Dept of CSE,YIT
*Compiler       : gcc compiler
*Date           : 10 February 2024
*************************************************************************/
#include<stdio.h>
#include<stdlib.h>
#define MAX 100

int main()
{
        int n,m,ht[MAX],i, j, k, rec, address, homebucket, currentbucket, count =0, choice;
        printf("Enter the number of employee records : ");
        scanf("%d", &n);
        for(i = 0; i < MAX; i++)
                ht[i] = -1;
        for(k = 0; k <n; k++)
        {
                printf("\nEnter the record %d\n", k+1);
                scanf("%d", &rec);
                address = rec % MAX;
                homebucket=address;
                currentbucket=homebucket;
                while(ht[currentbucket] != -1)
                {
                        currentbucket = (currentbucket + 1)  % MAX;
                        if(currentbucket == homebucket)
                        {
                                printf("Hash Table Overflow");
                                exit(0);
                        }
                        count++;
                }
                if(count != 0)
```

```
printf("Collision occured %d times and solved using LinearProbing\n", count);
 count=0;
 ht[currentbucket] = rec;
printf("Record:%d\nHome Address : %d\nCurrent
Adrress:%d\n",rec,homebucket,currentbucket);

}

printf("\nHASHTABLE DISPLAY\n");

        while(1)

         {
                printf("\n\n**********************MENU*****************");
                printf("\n1. Complete Hash table contents\n2. Hash Table showing
                only record entries\n3. Exit\n\n");
                printf("Enter your choice  :
                "); scanf("%d", &choice);
                switch(choice)
                {
                        case 1 : printf("Complete Hash Table Contents :\n");
                                 for(j = 0; j < MAX; j++)
                                 printf("%d%d\n",j,ht[j]);
                                 break;
                        case 2 : printf("Hash Table showing Records : \n");
                                 for(j = 0; j < MAX; j++)
                                  if(ht[j] != -1)
                                  printf("%d%d\n",j,ht[j]);
                                  break;
                        case 3: exit(0);
                                  break;
                }
         }
}
```

************************************OUTPUT************************************

Enter the number of employee records : 5

Enter the record 1
1231
Record : 1231
Home Address : 31
Current Adrress : 31

Enter the record 2
1299
Record : 1299
Home Address : 99
Current Adrress : 99

Enter the record 3
1265
Record : 1265
Home Address : 65
Current Adrress : 65

Enter the record 4
2231
Collision occured 1 times and solved using Linear Probing
Record : 2231
Home Address : 31
Current Adrress : 32

Enter the record 5
2299
Collision occured 1 times and solved using Linear Probing
Record : 2299
Home Address : 99
Current Adrress : 0

HASH TABLE DISPLAY

**********************MENU*********************

1. Complete Hash table contents
2. Hash Table showing only record entries
3. Exit

Enter your choice :
1

Complete Hash Table Contents :

0	2299
1	-1
2	-1
3	-1
4	-1
5	-1
6	-1
7	-1
8	-1
9	-1
10	-1
11	-1
12	-1
13	-1
14	-1
15	-1
16	-1
17	-1
18	-1
19	-1
20	-1
21	-1
22	-1
23	-1
24	-1
25	-1
26	-1
27	-1
28	-1
29	-1
30	-1
31	1231
32	2231
33	-1
34	-1
35	-1
36	-1
37	-1
38	-1
39	-1
40	-1
41	-1
42	-1

43 -1
44 -1
45 -1
46 -1
47 -1
48 -1
49 -1
50 -1
51 -1
52 -1
53 -1
54 -1
55 -1
56 -1
57 -1
58 -1
59 -1
60 -1
61 -1
62 -1
63 -1
64 -1
65 1265
66 -1
67 -1
68 -1
69 -1
70 -1
71 -1
72 -1
73 -1
74 -1
75 -1
76 -1
77 -1
78 -1
79 -1
80 -1
81 -1
82 -1
83 -1
84 -1
85 -1
86 -1

```
87      -1
88      -1
89      -1
90      -1
91      -1
92      -1
93      -1
94      -1
95      -1
96      -1
97      -1
98      -1
99      1299
**********MENU**********
  1. Complete Hash table contents
  2. Hash Table showing only record entries
  3. Exit

Enter your choice : 2
Hash Table showing Records :
  0     2299
  31    1231
  32    2231
  65    1265
  99    1299

**********MENU**********
  1. Complete Hash table contents
  2. Hash Table showing only record entries
  3. Exit
Enter your choice : 3
```