||Jai Sri Gurudev||

Sri AdichunchanagiriShikshana Trust(R)

SJB Institute of Technology

No. 67, BGS Health & Education City, Dr.Vishnuvardhan Road

Kengeri, Bangalore – 560 060

Department of CSE (Data Science)

Engineering

Data Structures Laboratory Manual

[BCSL305]

III SEMESTER – B. E

Academic Year 2023-2024

Staff Name:	**Dr.S. Nagamani**		
Section:	**III -A**	**Batch:**	**A1, A2,A3**

PREFACE

A data structure is a special way of organizing and storing data in a computer so that it can be used efficiently. Array, Linked List, Stack, Queue, Tree, Graph etc are all data structures that stores the data in a special way so that we can access and use the data efficiently. Each of these mentioned data structures has a different special way of organizing data so we choose the data structure based on the requirement, we will cover each of these data structures in a separate tutorials.

The C programming language is a structure oriented programming language, developed at Bell Laboratories in 1972 by Dennis Ritchie. C programming language features were derived from an earlier language called “B” (Basic Combined Programming Language – BCPL). C language was invented for implementing UNIX operating system.

Below are the steps to be followed for any C program to create and get the output. This is common to all C program and there is no exception whether its a very small C program or very large C program. 1.Create, 2.Compile, 3.Execute or Run, 4.Get the Output.

SJB INTITUTE OF TECHNOLOGY

Institution's Vision

To become a recognized technical education centre with a global perspective.

Institution's Mission

To provide learning opportunities that foster students ethical values, intelligent development in science & technology and social responsibility so that they become sensible and contributing members of the society.

Department of CSE (Data Science) Engineering

Department Vision

To enrich the next generation of young data practitioners, accomplish academic excellence and bring forward the Data Scientists.

Department Mission

M1: Grooming the students equipping with advanced technical knowledge to be industry-ready and globally competent.

M2: Facilitate quality data science education, enable students to become skilled professionals to solve real-time problems through industry collaboration.

M3: Encourage ethical value based transformation to serve the society with responsibility emphasizing on innovation and research methods

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

Graduates will -

PEO1. Apply the structured statistical and mathematical methodology to process massive amounts of data to detect underlying patterns to make predictions under realistic constraints and to visualize the data.

PEO2. Promote design, research, product implementation and services in the field of Data Science by using modern IT tools

PROGRAM SPECIFIC OUTCOMES (PSO'S)

Graduates will be able to -

PSO1 : Apply the skills in the multi-disciplinary area of data science.

PSO2: Demonstrate engineering practice learnt to solve real-time problems in various domains.

PROGRAM OUTCOMES-PO's

Engineering graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution ofcomplex 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 considerations.

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 withan 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.

Course Title: DSA Laboratory As per Choice Based Credit System (CBCS) scheme SEMESTER: III			
Subject code	**BCSL305**	IA Marks	**50**
Number of lecture hours/week	0:0:2	Exam Marks	**50**
Total Number of Lecture Hours	28	Exam Hours	**03**
Credits -03		Total Marks-100	

Course 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

Sl.No.	Experiment	RBT
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.	L1,L2,L3
2	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.	L1,L2,L3
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	L1,L2,L3
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.	L1,L2,L3
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	L1,L2,L3

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	L1,L2,L3
7.	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	L1,L2,L3
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	L1,L2,L3
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) = 6x2 y 2 z-4yz5 +3x3 yz+2xy5 z-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	L1,L2,L3
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	L1,L2,L3
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	L1,L2,L3
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.	L1,L2,L3

- **Course Outcomes:** At the end of the course the student will be able to:

CO 1. Analyze various linear and non-linear data structures
CO 2. Demonstrate the working nature of different types of data structures and their applications
CO 3. Use appropriate searching and sorting algorithms for the give scenario
CO 4. Apply the appropriate data structure for solving real world problems

Conduction of Practical Experiment distribution

o For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.

o 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

University Syllabus

COURSE OUTCOMES

On successful completion of this course students will be able to,

CO's	COURSE OUTCOMES	PO's and PSO's MAPPING
CO1	Analyze various linear and non-linear data structures	PO1, PO2,PO3 / PSO1
CO2	Demonstrate the working nature of different types of data structures and their applications	PO1, PO2,PO3 / PSO1
CO3	Use appropriate searching and sorting algorithms for the give scenario	PO1, PO2,PO3,PO5 / PSO1
CO4	Apply the appropriate data structure for solving real world problems	PO1,PO2,PO3,PO4/ PSO1

General Instructions for the Laboratory

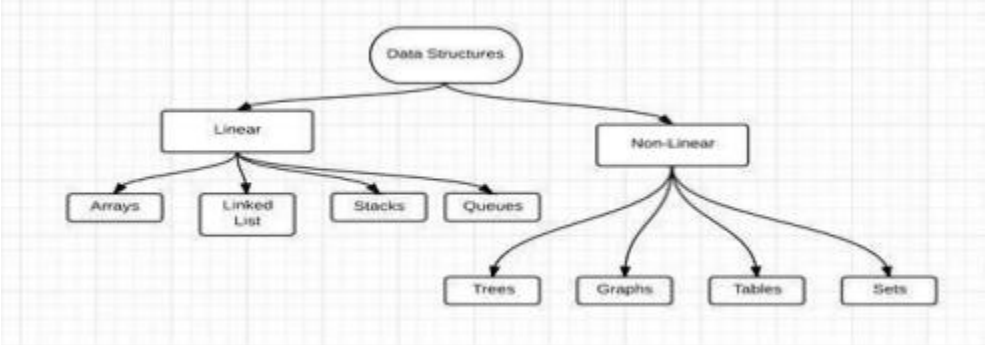
Do's

- It is mandatory for all the students to attend all practical classes & complete the experiments as per syllabus.
- Students should strictly follow the lab timings, dress code with Apron & ID cards.
- Should maintain a neat observation book.
- Study the theory and logic before executing the program.
- Submit the completed lab records of executed programs and update the index book in every lab session.
- Should prepare for viva questions regularly.
- Handle the computer systems carefully.
- Maintain discipline and silence in the lab.

Don'ts

- Should not take Bags and Mobile phones into the Laboratory.
- Do not wear footwear inside the Laboratory
- Systems & Components should be handled carefully failing to which penalty will be imposed.
- Do not switch off the system abruptly.
- Should not chew gum or eat in the lab

INTRODUCTION TO DATA STRUCTURES



Data Structure is defined as the way in which data is organized in the

memory location. There are 2 types of data structures:

Linear Data Structure:

In linear data structure all the data are stored linearly or contiguously in the memory. All the data aresaved in continuously memory locations and hence all data elements are saved in one boundary. A linear data structure is one in which we can reach directly only one element from another while travelling sequentially. The main advantage, we find the first element, and then it's easy to find the next data elements. The disadvantage, the size of the array must be known before allocating the memory.

The different types of linear data structures are:

- Array
- Stack
- Queue
- Linked List

Non-Linear Data Structure:

Every data item is attached to several other data items in a way that is specific for reflectingrelationships. The data items are not arranged in a sequential structure.

The various types of non linear data structures are:

- Trees
- Graphs

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.**

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

    #define NUM_DAYS_IN_WEEK 7

    //Global variable Declaration
    int i;


    // Structure to represent a day
    typedef struct
    {
       char *acDayName;          // Dynamically allocated string for the day name
       int iDate;  // Date of the day
       char *acActivity; // Dynamically allocated string for the activity
       description
    }DAYTYPE;


    //Prototypes
    void  FreeCal(DAYTYPE  *);
    void  DispCal(DAYTYPE  *);
    void ReadCal(DAYTYPE *);
    DAYTYPE *CreateCal();


    //Main function
    int main()
    {
       // Create the calendar
       DAYTYPE *weeklyCalendar = CreateCal();


       // Read data from the keyboard
       ReadCal(weeklyCalendar);
```

```
    // Display the week's activity details
    DispCal(weeklyCalendar);
        // Free allocated memory

    FreeCal(weeklyCalendar);

    return 0;
}

// Createcalender function definition
DAYTYPE *CreateCal()
{
    DAYTYPE *calendar = (DAYTYPE *)malloc(NUM_DAYS_IN_WEEK *
sizeof(DAYTYPE));


    for( i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
        calendar[i].acDa
        yName = NULL;
        calendar[i].iDate
        = 0;
        calendar[i].acAct
        ivity = NULL;
    }

    return calendar;
}
// Read Calender function definition
void ReadCal(DAYTYPE *calendar)
{
    char Choice;
              for( i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
        printf("Do you want to enter details for day %d [Y/N]:
        ", i + 1);
        scanf("%c", &Choice);
          getchar();

        if(tolower(Choice) == 'n')
        continue;

        printf("Day Name: ");
        char nameBuffer[50]; scanf("%s",
        nameBuffer);
        calendar[i].acDayName = strdup(nameBuffer); // Dynamically
allocate and copy the string
```

```
        printf("Date: ");
        scanf("%d", &calendar[i].iDate);

        printf(
       "Activit
       y: ");
       char
       activity
       Buffer[
       100];

      scanf(" %[^\n]", activityBuffer); // Read the entire line, including spaces
      calendar[i].acActivity = strdup(activityBuffer);
         printf("\n");
        getchar(); //remove trailing enter character in input buffer


    }
}

// DisplayCalender function definition
void DispCal(DAYTYPE *calendar)
{
    printf("\nWeek's Activity Details:\n");
    for(i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
        printf("Day %d:\n", i + 1); if(calendar[i].iDate
              == 0)
               {
                       printf("No Activity\n\n");
                       continue;
               }

        printf(" Day Name: %s\n", calendar[i].acDayName);
        printf(" Date: %d\n", calendar[i].iDate);
        printf(" Activity: %s\n\n", calendar[i].acActivity);
    }
}

// FreeCalender function definition
void FreeCal(DAYTYPE *calendar)
{
    for( i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
        free(calendar[i].a
        cDayName);
```

```
            free(calendar[i].a
            cActivity);
        }
        free(calendar);
    }
```

Output :

```
Name of the day: Monday
Date of the day: 24
Activity for the day: running
Enter details for day 2:
Name of the day: tuesday
Date of the day: 23
Activity for the day: walking
Enter details for day 3:
Name of the day: wednesday
Date of the day: 25
Activity for the day: thursday
Enter details for day 4:
Name of the day: friday
Date of the day: 26
Activity for the day: reading
Enter details for day 5:
Name of the day: saturday
Date of the day: 27
Activity for the day: writing
Enter details for day 6:
Name of the day: sunday
Date of the day: 28
Activity for the day: jogging
Enter details for day 7:
Name of the day: thursday
Date of the day: 22
Activity for the day: gym workout

Weekly Activity Details:
Day 1: Monday, Date: 24, Activity: running
Day 2: tuesday, Date: 23, Activity: walking
Day 3: wednesday, Date: 25, Activity: thursday
Day 4: friday, Date: 26, Activity: reading
Day 5: saturday, Date: 27, Activity: writing
Day 6: sunday, Date: 28, Activity: jogging
Day 7: thursday, Date: 22, Activity: gym workout
```

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.

```
#include<std
io.h>
#include<stri
ng.h>
#include<std
lib.h>

char str[50],
pat[50],
rep[50]; int
start = 0,
patfound = 0;
int lasts,
lastp, lastr;

void replacepattern()
{
        int i, j;
        lastr = strlen(rep)-2;

        if(lastp != lastr)
        {
        printf("\nInvalid length of replace
    string");
        exit(0);

        }
        else
        {
            i = start;
         for(j=0; j<=lastr; j++)
            {
            str[i] =
         rep[j];
             i++;
                }
        }
        return;
}
```

```
void findpattern()
    {
            int i, j, inmatch;
            lasts = (strlen(str))-2; lastp = (strlen(pat))-2; int endmatch;

            for(endmatch = lastp; endmatch<=lasts; endmatch++, start++)
            {
                    if(str[endmatch] == pat[lastp])
                    {inmatch = start;
                    j=0;
                            while(j<lastp)
                            {
                                    if(str[inmatch] == pat[j])
                                    {
                                            inmatch++;

                                       j++;
                                    }
                                    else
                                    {

                                        break;

                                    }
                            }

                            if(j == lastp)
                            {patfound = 1;
                             replacepattern();
                               }
                           }
                     }
                       return;
            }


    void main()
    {
            printf("\nEnter the main string(STR): ");
             fgets(str, 50, stdin);

            printf("\nEnter the pattern to be
            matched(PAT): ");
            fgets(pat, 50, stdin);

            printf("\nEnter the string to be
```

```
        replaced(REP): ");
        fgets(rep, 50, stdin);

        printf("\nThe string before pattern match is:\n %s", str); findpattern();

        if(patfound == 0)
                printf("\nThe pattern is not found in the main string");
        else
                printf("\nThe string after pattern match and replace is: \n %s ",str);

        return;
}
```

Output:

Case 1:

Enter the main string(STR): **Hello hii how are you hii**

Enter the pattern to be matched(PAT): **hii** Enter the string to be replaced(REP):

xyz The string before pattern match is:
Hello hii how are you hii

The string after pattern match and replace is:
Hello xyz how

are you xyz Case

2:

Enter the main string(STR): **Hello hii how are you**

Enter the pattern to be

matched(PAT): **abc** Enter

the string to be

replaced(REP): **xyz** The

string before pattern

match is:
Hello hii how are you

The pattern is not found in the main string

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

```
#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.Di
                splay ");
                printf("\n=>5.E
                xit");
                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
Eleme
nt
poppe
d 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
choic
e: **4**
Stac
k
elem
ents
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
Eleme
nt
poppe
d 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
Eleme
nt
poppe
d 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
Eleme
nt
poppe
d 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.

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

void evaluate();
void push(char);
 char pop();
int prec(char);

char infix[30],
postfix[30],
stack[30]; int top = -
1;

void main()
{
        printf("\nEnter the valid infix
        expression:\t"); scanf("%s",
        infix);
        evaluate();
        printf("\nThe entered infix expression is :\n %s \n",
        infix); printf("\nThe corresponding postfix expression
        is :\n %s \n", postfix);
```
~~~~~

```
}

void evaluate()
{int i = 0, j = 0; char symb, temp;

        push('#');

        for(i=0; infix[i] != '\0'; i++)
        {
                symb =
                infix[i];
                switch(sym
                b)
                {
                        case '(' :


                                        p
                                        u
                                        s
                                        h
                                        (
                                        s
                                        y
                                        m
                                        b
                                        )
                                        ;

                                        b
                                        r
                                        e
                                        a
                                        k
                                        ;

                        case ')' :    temp = pop();
                                        while(temp != '(' )
                                        {
                                       postfix[j] = temp;

                                                j++;
                                        temp = pop();}
                                        break;
                         case '+' :
                         case '-' :
                         case '*' :
```

```
                        case '/' :
                        case '%' :
                        case '^' :

 }
                case '$'  :         while( prec(stack[top]) >= prec(symb) )
        {       temp = pop(); postfix[j] = temp; j++;  }
                        push(symb); break;
                           default;
                           postfix[j = symb; j++;
                 }
        }
        while(top > 0)
        {
                temp = pop();
                postfix[j] = temp; j++;
        }
        postfix[j] = '\0';
}

 void push(char item)
{
        top = top+1; stack[top] = item;
}

 char pop()
{
        char item;
        item = stack[top]; top = top-1; return
        item;
}

int prec(char symb)
{int p; switch(symb)
        {
        case '#' :    p = -1; break;
                case '(' :
           case ')' :  p = 0; break;
                case '+' :
           case '-' :  p = 1; break;

                case '*' :
                case '/' :
                case '%' :   p = 2; break;

                case '^' :
```

```
                case '$' :    p = 3; break;
        }
        return p;
}
```

Output:

Enter the valid infix expression: **(a+b)+c/d*e**
The entered infix expression is : (a+b)+c/d*e
The corresponding postfix expression is : ab+cd/e*+

5a. Develop a Program in C for the following Stack Applications. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %,^

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

int i, top = -1;
int op1, op2, res, s[20]; char postfix[90], symb;

void push(int item)
{top = top+1; s[top] = item;
}

int pop()
{       int item;item = s[top];
top = top-1;
return item;
}

void main()
{
        printf("\nEnter a valid postfix
        expression:\n"); scanf("%s",
        postfix);
        for(i=0; postfix[i]!='\0'; i++)
        {  symb = postfix[i];
        if(isdigit(symb))
                {
                    push(symb - '0');
                }
                else
                {       op2 = pop();
                        op1 = pop();
                switch(symb)
```

```
                {
                        case '+':       push(op1+op2); break;
                        case '-':       push(op1-op2); break;
                        case '*':       push(op1*op2); break;
                        case '/':

                                        p
                                        u
                                        s
                                        h
                                        (
                                        o
                                        p
                                        1
                                        /
                                        o
                                        p
                                        2
                                        )
                                        ;

                                        b
                                        r
                                        e
                                        a
                                        k
                                        ;
                        case '%':       push(op1%op2); break;
                        case '$':
                        case '^':       push(pow(op1, op2));
                                        break;
                        default :       push(0);
                    }
                }
            }
            res = pop();
            printf("\n Result = %d", res);
    }
```

Output:

To compile in Linux: gcc –lm 5.c

Enter a valid postfix expression:

623+-382/+*2$3+

Result = 52

Enter a valid postfix expression:
42$3*3-84/11+/+
Result = 46

5b. Solving Tower of Hanoi problem with n disks.

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

void tower(int n, char from_peg, char aux_peg, char to_peg);

void main()
{
     int n;
    printf("\nEnter the number of disks: ");
scanf("%d", &n);
     tower(n, 'A', 'B', 'C');  // A-> from_peg B->aux_peg C->to_peg
    printf("\nTotal number of moves = %0.0f", pow(2,n)-1 );
}

void tower(int n, char from_peg, char aux_peg, char to_peg)
  // A-> from_peg B->aux_peg C->to_peg
{
          if(n == 1)
          { printf("\nMove disk %d from %c peg to %c peg", n, from_peg,
   to_peg);
           return;
           }

          // move n-1 disks from A(from_peg) to B(to_peg) using C(aux_peg) as auxiliary
          tower(n-1, from_peg, to_peg, aux_peg);

          printf("\nMove disk %d from peg %c to %c peg", n, from_peg, to_peg);

          // move n-1 disks from B(aux_peg) to C(to_peg) using A(from_peg) as auxiliary
          tower(n-1, aux_peg, from_peg, to_peg);
}
```

Output:

Enter the number of disks: **3**

Move disk 1 from A peg to C peg Move disk 2 from peg A to B peg

Move disk 1 from C peg to B peg Move disk 3 from peg A to C peg Move disk 1 from B peg to A peg Move disk 2 from peg B to C peg Move disk 1 from A peg to C peg Total number of moves = **7**

6. Develop a menu driven Program in C for the following operations on CircularQUEUE of Characters (Array Implementation of Queue with maximum size MAX)
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
Support the program with appropriate functions for each of the above operations.

```
#include <stdio.h>
#include<stdlib.h>
 #include<stdio_ext.h>
#define MAX 3

char cq[MAX];
int front = -1, rear = -1;

void insert(char);
void delete();
void display();
void main()
{int ch; char item;
while(1)
        {
                printf("\n\n~~Main Menu~~");
                printf("\n==> 1. Insertion and Overflow
                Demo"); printf("\n==> 2. Deletion and
```

```
                    Underflow Demo"); printf("\n==> 3.
                    Display");
                    printf("\n==> 4.
                    Exit");
                    printf("\nEnter
                    Your Choice: ");
                    scanf("%d", &ch);

                    __fpurge(stdin);
                   switch(ch)
                   {
                            case 1:     printf("\n\nEnter the element
                                        to be inserted: "); scanf("%c",
                                        &item);
                                        insert(item);
                                        break;
                            case 2:       delete();
                                        break;
                            case 3:       display();
                                        break;
                             case 4:      exit(0);
                             default:     printf("\n\nPlease enter a valid choice");

                    }
          }
}


void insert(char item)
{
          if(front == (rear+1)%MAX)
          {
                    printf("\n\n~~Circular Queue Overflow~~");
          }
          else
          {  if(front == -1)
               front = rear = 0;
                    else
                    rear = (rear+1)%MAX;
                    cq[rear] = item;
           }
}

void delete()
{char item;
      if(front == -1)
            {
```

```
                printf("\n\n~~Circular Queue Underflow~~");
            }
        else
          {

    item = cq[front];
    printf("\n\nDeleted element from the queue is: %c ",item );

             if(front == rear) //only one element
              front = rear = -1;
         else
            front = (front+1)%MAX;

        }
    }


void display ()
{
            int i ;
            if(front == -1)
            {
                    printf("\n\nCircular Queue Empty");

            }
            else
            {    printf("\nCircular Queue contents are:\n");

                  printf("Front[%d]-> ", front);
                   for(i = front; i != rear ; i = (i+1)%MAX)
                   {
                                printf(" %c", cq[i]);
                   }
                   printf(" %c", cq[i]);
                   printf(" <-[%d]Rear", rear);
            }
}
```

Output:

```
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1
```

Enter the element to be inserted: A

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **1**

Enter the element to be inserted: B

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **1**

Enter the element to be inserted: C

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **1**

Enter the element to be inserted: D
~~Circular Queue Overflow~~
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **3**

Circular Queue contents are: Front[0]-> A B C <-[2]Rear

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **2**

Deleted element from the queue is: A

~~Main Menu~~
==> 1. Insertion and Overflow Demo

==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your
Choice: **3**

Circular Queue contents are: Front[1]-> B C <-[2]Rear

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **1**

Enter the element to be inserted: E
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: **3**
Circular Queue contents are: Front[1]-> B C E <-[0]Rear

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 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, 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

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

struct node
{
```

```
    char
    usn[25],name[25],branch[25];
    int sem;
    long int phone; struct node *link;
};
typedef struct node * NODE;

NODE start = NULL; int
count=0;

NODE create()
{
    NODE snode;
    snode = (NODE)malloc(sizeof(struct node));

    if(snode == NULL)
    {
        printf("\nMemory is
        not available");
        exit(1);
    }
    printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
    scanf("%s %s %s %d %ld",snode->usn, snode->name, snode-
>branch, &snode->sem, &snode->phone);
 snode -
    >link=NULL;
    count++;
    return snode;
}

NODE insertfront()
{
    NODE temp; temp =
    create(); if(start == NULL)
    {
          return temp;
    }

    temp->link = start; return temp;
}
```

```
NODE deletefront()
{
   NODE temp; if(start == NULL)
   {
      printf("\nLinked
      list is empty");
      return NULL;
   }

   if(start->link == NULL)
   {
         printf("\nThe Student node with usn:%s is
         deleted ",start->usn);
         count--;
         fee(start); return NULL;
   }
   temp = start;
   start = start->link;
   printf("\nThe Student node with usn:%s is
   deleted",temp->usn); count--;
   free(temp); return start;
}

NODE insertend()
{ NODE cur,temp; temp = create();

   if(start == NULL)
   {
    return temp;
   }
   cur = start;
   while(cur->link !=NULL)
   {
       cur = cur->link;
   }
   cur->link = temp; return start;
}

NODE deleteend()
{
```

```
        NODE cur,prev;
        if(start == NULL)
        {
          printf("\nLinked
          List is empty");
          return NULL;
        }

        if(start->link == NULL)
        {
          printf("\nThe student node with the usn:%s is
          deleted",start->usn); free(start);
          count--; return NULL;
        }
        prev = NULL; cur = start;
        while(cur->link!=NULL)
        {
           prev = cur;
           cur = cur->link;
        }
         printf("\nThe student node with the usn:%s is
         deleted",cur->usn); free(cur);
         prev>link = NULL; count--;
         return start;
    }

    void display()
    {
    NODE cur; int num=1;

if(start == NULL)
        {
         printf("\nNo Contents to display in SLL \n"); return;
        }
        printf("\nThe
        contents of SLL:
        \n"); cur = start;
        while(cur!=NULL)
        {
          printf("\n||%d|| USN:%s| Name:%s| Branch:%s| Sem:%d| Ph:%ld|",num,cur->usn, cur-
    >name,cur->branch, cur-
```

```
      >sem,cur->phone); cur
      = cur->link;
      num++;
    }
    printf("\n No of student nodes is %d \n",count);
}

void stackdemo()
{ int ch;
  while(1)
  { printf("\n~~~Stack Demo using SLL~~~\n");
    printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit
    \n"); printf("\nEnter your choice for stack demo");

    scanf("%d",&ch);
    switch(ch)
    {
            case 1: start = insertfront();
                    break;
           case 2: start = deletefront();
                   break;
      case 3: display(); break;
      default : return;
    }
  }
  return;
}

int main()
{
   int ch,i,n;
   while(1)
   {
      printf("\n~~~Menu~~~");
      printf("\nEnter your choice for SLL
      operation \n"); printf("\n1:Create
      SLL of Student Nodes");
      printf("\n2:DisplayStatus");
      printf("\n3:InsertAtEnd");
      printf("\n4:DeleteAtEnd");
      printf("\n5:Stack Demo using SLL(Insertion and Deletion at
```

```
        Front)"); printf("\n6:Exit \n");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1 : printf("\nEnter the no
                of students:      ");
                scanf("%d",&n);
                for(i=1;i<=n;i++)
                  start = insertfront();
                  break;
                case 2: display();
                        break;
                 case 3: start = insertend();
                          break;
                 case 4: start = deleteend();
                          break;
                 case 5: stackdemo();
                           break;

                  case 6: exit(0);

        default: printf("\nPlease enter the valid choice");
       }
      }
    }
```

Output:

```
~~~Menu~~~
Enter your choice
for SLL operation
1:Create SLL of
Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and
Deletion at Front) 6:Exit
Enter your choice:1
```

Enter the no of students: 3

Enter the usn,Name,Branch, sem,PhoneNo

of the student: 111
aaa
cs
1
111111

Enter the usn,Name,Branch, sem,PhoneNo
of the student: 222
bbb
ec
2
222222

Enter the usn,Name,Branch, sem,PhoneNo
of the student: 333
ccc
ec
3
333333

~~~Menu~~~
Enter your choice for SLL
operation 1:Create SLL of
Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**2**

The contents of SLL:

||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|

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

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

struct node
{
        char
        ssn[25],name[25],dept[10],design
        ation[25]; int sal;
        long int phone; struct
        node *llink; struct node
        *rlink;
};
typedef struct node* NODE;

NODE first = NULL;
int count=0;

NODE create()
{
         NODE enode;
         enode =
         (NODE)malloc(sizeof(struct
         node)); if( enode== NULL)
         {
                 printf("\nRunning out
                 of memory"); exit(0);
         }
         printf("\nEnter the ssn,Name,Department,Designation,Salary,PhoneNo of the
         employee: \n");
         scanf("%s %s %s %s %d %ld", enode->ssn, enode->name, enode->dept, enode->
         designation, &enode->sal, &enode->phone);
         enode->llink=NULL;
         enode->rlink=NULL;

count+
+;
```

```
return
enode;
    }

    NODE insertfront()
    {
        NODE temp; temp = create();
        if(first == NULL)
        {
            return temp;
        }
        temp->rlink = first;
        first->llink = temp;
        return temp;
}

    void display()
    {
        NODE cur; int nodeno=1; cur = first;
        if(cur == NULL)
            printf("\nNo Contents to display in DLL");
        while(cur!=NULL)
       { printf("\nENode:%d||SSN:%s|Name:%s|Department:%s|Designation:%s|
    Salary:%d|Phone no:%ld", nodeno, cur->ssn, cur->name,cur->dept, cur-
    >designation, cur->sal, cur->phone);
            cur = cur->rlink;
            nodeno++;
        }


        printf("\nNo of employee nodes is %d",count);
    }

    NODE deletefront()
    {
        NODE temp; if(first == NULL)
        {
            printf("\nDoubly Linked List is empty");
```

```
            return NULL;
        }

        if(first->rlink== NULL)
        {
            printf("\nThe employee node with the ssn:%s is deleted",
            first->ssn); free(first);
            count--;
            return NULL;
        }
        temp = first;
        first = first->rlink; temp->rlink = NULL;
        first->llink = NULL;
        printf("\nThe employee node with the ssn:%s is
        deleted",temp->ssn); free(temp);
        count--; return first;
}

NODE insertend()
{NODE cur, temp; temp = create();

        if(first == NULL)
        {
            return temp;
        }
        cur= first;
        while(cur->rlink!=NULL)
        {
            cur = cur->rlink;
        }

        cur->rlink = temp; temp->llink = cur; return
        first;
}

NODE deleteend()

{
        NODE prev,cur; if(first == NULL)
        {
            printf("\nDoubly
```

```
            Linked List is empty");
            return NULL;
        }

        if(first->rlink == NULL)
        {
            printf("\nThe employee node with the ssn:%s is
            deleted",first->ssn); free(first);
            count--; return NULL;
        }
      prev=NULL;
       cur=first;

        while(cur->rlink!=NULL)
        {
            prev=cur;
            cur = cur->rlink;
         }

        cur->llink = NULL;
        printf("\nThe employee node with the ssn:%s is
        deleted",cur->ssn); free(cur);
        prev->rlink = NULL; count--;
        return first;
}

void deqdemo()
{ int ch;
while(1)
     {
          printf("\nDemo Double Ended Queue Operation");
     printf("\n1:InsertQueueFront\n 2: DeleteQueueFront\n
     3:InsertQueueRear\n


4:DeleteQueueRear\n
          5:DisplayStatus\n 6: Exit
          \n"); scanf("%d", &ch);

          switch(ch)
          {    case 1: first=insertfront(); break;
               case 2:
```

```
                    first=de
                    letefron
                    t();
                    break;
                case 3: first=insertend();
                    break;
                case 4:
                    first=
                    delete
                    end();
                    break;
                case 5: display();
                break; default : return;
            }
        }
}
   void main()
{   int ch,i,n;
 while(1)
    {
       printf("\n\n~~~Menu~~~");
       printf("\n1:Create DLL of
       Employee Nodes");
       printf("\n2:DisplayStatus");
       printf("\n3:InsertAtEnd");
       printf("\n4:DeleteAtEnd");
       printf("\n5:InsertAtFront");
       printf("\n6:DeleteAtFront");
        printf("\n7:Double Ended Queue Demo
        using DLL"); printf("\n8:Exit \n");
        printf("\nPlease enter your
       choice: "); scanf("%d",&ch);

       switch(ch)
       {
        case 1 : printf("\nEnter the no of Employees: ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        first = insertend();
        break;
        case 2: display();
```

```
        break;
        case 3: first = insertend(); break;
        case 4: first = deleteend(); break;
        case 5: first = insertfront(); break;
        case 6: first = deletefront(); break;
         case 7: deqdemo(); break;
         case 8 : exit(0);
       default: printf("\nPlease Enter the valid choice");
       }
   } }
```

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.

```
#include<stdio.h>
 #include<stdlib.h>
#include<math.h>
#define COMPARE(x, y) ( (x == y) ? 0 : (x > y) ? 1 : -1)

struct node
{
          int coef;
          int xexp, yexp, zexp; struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
          NODE x;
          x = (NODE)
          malloc(sizeof(struct
          node)); if(x == NULL)
          {
                    printf("Running out of memory \n"); return NULL;
          }
          return x;
}

NODE attach(int coef, int xexp, int yexp, int zexp, NODE head)
{
          NODE temp, cur;
           temp = getnode();
```

```
        temp->coef = coef;
        temp->xexp = xexp;
        temp->yexp = yexp;
        temp->zexp = zexp;
        cur = head->link;
        while(cur->link != head)
        {
                cur = cur->link;
        }
        cur->link = temp;
        temp->link = head;
 return head;
}

NODE read_poly(NODE head)
{
        int i, j, coef, xexp, yexp, zexp,  n;
        printf("\nEnter the no of terms in the
        polynomial: "); scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
                printf("\n\tEnter
                the %d term: ",i);
                printf("\n\t\tCoef
                = "); scanf("%d",
                &coef);
                printf("\n\t\tEnter Pow(x) Pow(y)
                and Pow(z): "); scanf("%d",
                &xexp);
                scanf("%d", &yexp);
                scanf("%d", &zexp);
                head = attach(coef, xexp, yexp, zexp, head);
        }
         return head;
}

void display(NODE head)
{
        NODE temp;
        if(head->link == head)
        {
                printf("\nPolynomial
                does not exist.");
                return;
        }
        temp = head->link;
```

```
            while(temp != head)
            {
     printf("%dx^%dy^%dz^%d", temp->coef, temp->xexp, temp->yexp, temp-
    >zexp);
     temp = temp->link; if(temp != head)
            printf(" + ");

        }
}


    int poly_evaluate(NODE head)
    {
            int x, y, z, sum = 0;
            NODE poly;

            printf("\nEnter the value of x,y and z: ");
            scanf("%d %d %d", &x, &y, &z);

            poly = head->link; while(poly != head)
            {
    sum += poly->coef * pow(x,poly->xexp)* pow(y,poly->yexp) * pow(z,poly-

    >zexp);

    poly = poly->link;

  }

            return sum;
    }

    NODE poly_sum(NODE head1, NODE head2, NODE head3)
    {
       NODE a, b;
       int coef;
       a = head1->link; b = head2->link;

      while(a!=head1 && b!=head2)
       {
            while(1)
            {
                  if(a->xexp == b->xexp && a->yexp == b->yexp && a->zexp == b->zexp)
                  {
                        coef = a->coef + b->coef;
                        head3 = attach(coef, a->xexp, a->yexp,
```

```
            a->zexp, head3);
             a = a->link;
            b = b->link; break;
} //if ends here
if(a->xexp!=0 || b->xexp!=0)
{
            switch(COMPARE(a->xexp, b->xexp))
            {
             case -1 : head3 = attach(b->coef, b->xexp, b->yexp,
                        b->zexp, head3); b = b->link;
                        break;

            case 0 :     if(a->yexp > b->yexp)
                        {
                              head3 =attach(a->coef, a->xexp, a->yexp,
                              a->zexp, head3); a = a->link;
                              break;
                   }
                         else if(a->yexp < b->yexp)
                         {
                              head3 =attach(b->coef, b->xexp, b->yexp,
                               b->zexp, head3); b = b->link;
                               break;
                         }
                        else if(a->zexp > b->zexp)
                        {
                              head3 = attach(a->coef, a->xexp, a->yexp,
                               a->zexp, head3); a = a->link;
                               break;
                        }
                        else if(a->zexp < b->zexp)
                        {
                             head3 = attach(b->coef, b->xexp, b->yexp,
                               b->zexp, head3);



                               b = b->link;
                               break;
                        }
            case 1 :    head3 = attach(a->coef,a->xexp,a-
                        >yexp,a->zexp,head3); a = a->link;
                        break;
            } //switch ends here
            break;
```

```
                } //if ends here
                if(a->yexp!=0 || b->yexp!=0)
                {
                        switch(COMPARE(a->yexp, b->yexp))
                        {
                            case -1 : head3 = attach(b->coef, b->xexp, b->yexp,
                                        b->zexp, head3); b = b->link;
                                        break;
                             case 0 : if(a->zexp > b->zexp)
                                        {
                                         head3 = attach(a->coef, a->xexp, a->yexp,
                                             a->zexp, head3); a = a->link;
                                             break;
                                         }
                                         else if(a->zexp < b->zexp)
                                         {
                                        head3 = attach(b->coef, b->xexp, b->yexp,
                                             b->zexp, head3); b = b->link;
                                             break;
                                    }
                             case 1 : head3 = attach(a->coef, a->xexp, a->yexp, a-
                                                >zexp, head3);
                                            a = a->link; break;
                        }
                        break;
                }
                 if(a->zexp!=0 || b->zexp!=0)
                 {
                  switch(COMPARE(a->zexp,b->zexp))
                        {

                    case -1 : head3 = attach(b->coef,b->xexp,b->yexp,b-
                >zexp,head3);
                              b = b->link;
                               break;
                    case 1 : head3=attach(a->coef, a->xexp, a->yexp, a->zexp,
                head3);
                             a = a->link;
                             break;

                        }
                        break;
                 }
        }
}

  while(a!= head1)
```

```
        {
                        head3 = attach(a->coef,a->xexp,a->yexp,a-
                >zexp,head3);
                        a = a->link;
        }

    while(b!= head2)
    {
        head3 = attach(b->coef,b->xexp,b-
        >yexp,b->zexp,head3); b = b->link;
    }
    return head3;
}

void main()
{
 NODE head,
 head1, head2,
 head3; int res, ch;
 head =  getnode();        /* For polynomial evalaution */
 head1 = getnode();        /* To
 hold POLY1 */ head2 =
 getnode();          /* To hold
 POLY2 */ head3 = getnode();
   /* To hold POLYSUM */

 head->link=head;
 head1->link=head1;
 head2->link=head2;
 head3->link= head3;
 while(1)
 {
     printf("\n~~~Menu~~~");
     printf("\n1.Represent and Evaluate a Polynomial
     P(x,y,z)"); printf("\n2.Find the sum of two
     polynomials POLY1(x,y,z)"); printf("\nEnter your
     choice:");
     scanf("
     %d",&c
     h);
     switch(
     ch)
     {
        case 1:     printf("\n~~~~Polynomial evaluation P(x,y,z)~~~\n");

                    head = read_poly(head);
                    printf("\nRepresentation of Polynomial for evaluation: \n");
```

```
                    display(head);
                    res = poly_evaluate(head);
                    printf("\nResult of polynomial evaluation
                    is : %d \n", res); break;
        case 2:   printf("\nEnter the POLY1(x,y,z): \n");
                   head1 = read_poly(head1);
                   printf("\nPolynomial 1 is: \n");
                   display(head1);
                   printf("\nEnter the POLY2(x,y,z): \n");
                  head2 = read_poly(head2);
                   printf("\nPolynomial 2 is: \n");
                     display(head2);
                     printf("\nPolynomial addition result: \n");
                     head3 = poly_sum(head1,head2,head3);
                     display(head3);
                      break;
          case 3:  exit(0);
      }
    }
  }
```

Output:

~~~Menu~~~

1. Represent and Evaluate a Polynomial P(x,y,z)
2. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) Enter your choice: **1**

~~~~Polynomial evaluation P(x,y,z)~~~

Enter the no of terms in the polynomial: **5**

Enter the 1 term:

Coef = **6**
Enter Pow(x) Pow(y) and Pow(z): **2 2 1**
Enter the 2 term: Coef = **-4**
Enter Pow(x) Pow(y) and Pow(z): **0 1 5**
Enter the 3 term: Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **3 1 1**
Enter the 4 term: Coef = **2**
Enter Pow(x) Pow(y) and Pow(z): **1 5 1**
Enter the 5 term: Coef = **-2**
Enter Pow(x) Pow(y) and Pow(z): **1 1 3**

Representation of Polynomial for evaluation:

6x^2y^2z^1 + -4x^0y^1z^5 + 3x^3y^1z^1 + 2x^1y^5z^1 + -2x^1y^1z^3

Enter the value of x,y and z: **1 1 1**

Result of polynomial evaluation is : **5**
~~~

~~~Menu~~~
1. Represent and Evaluate a Polynomial P(x,y,z)
2. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)
3. Enter your choice: **2**

Enter the POLY1(x,y,z):
Enter the no of terms in the polynomial: **5**
Enter the 1 term: Coef = **6**
Enter Pow(x) Pow(y) and Pow(z): **4 4 4**
Enter the 2 term: Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **4 3 1**
Enter the 3 term: Coef = **5**
Enter Pow(x) Pow(y) and Pow(z): **0 1 1**
Enter the 4 term: Coef = **10**
Enter Pow(x) Pow(y) and Pow(z): **0 1 0**
Enter the 5 term: Coef = **5**
Enter Pow(x) Pow(y) and Pow(z): **0 0 0**
Polynomial 1 is:
6x^4y^4z^4 + 3x^4y^3z^1 + 5x^0y^1z^1 + 10x^0y^1z^0 + 5x^0y^0z^0

Enter the POLY2(x,y,z):
Enter the no of terms in the polynomial: **5**
Enter the 1 term: Coef = **8**
Enter Pow(x) Pow(y) and Pow(z): **4 4 4**
Enter the 2 term: Coef = **4**
Enter Pow(x) Pow(y) and Pow(z): **4 2 1**
Enter the 3 term: Coef = **30**
Enter Pow(x) Pow(y) and Pow(z): **0 1 0**
Enter the 4 term: Coef = **20**
Enter Pow(x) Pow(y) and Pow(z): **0 0 1**
Enter the 5 term: Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **0 0 0**
Polynomial 2 is:
8x^4y^4z^4 + 4x^4y^2z^1 + 30x^0y^1z^0 + 20x^0y^0z^1 + 3x^0y^0z^0

Polynomial addition result:
14x^4y^4z^4 + 3x^4y^3z^1 + 4x^4y^2z^1 + 5x^0y^1z^1 + 40x^0y^1z^0 + 20x^0y^0z^1 + 8x^0y^0z^0

~~~Menu~~~
1. Represent and Evaluate a Polynomial P(x,y,z)
2. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) Enter your choice:3

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

```
#include<stdio.h>
 #include<stdlib.h>
struct BST
{
        int data;
        struct BST *lchild; struct BST *rchild;
};
typedef struct BST * NODE;

NODE create()
{
        NODE temp;
        temp = (NODE)
        malloc(sizeof(struct BST));
        printf("\nEnter The value: ");
        scanf("%d", &temp->data);
       temp->lchild = NULL;
       temp->rchild = NULL;
       return temp;
}

void insert(NODE root,
NODE newnode); void
inorder(NODE root);
void
preorder(N
ODE root);
void
postorder(N
ODE root);
void
search(NO
DE root);

void insert(NODE root, NODE newnode)
    {

   /*Note: if newnode->data == root->data it will be skipped. No duplicate nodes
are allowed */
```

```
        if (newnode->data < root->data)
        {
                if (root->lchild == NULL)
                        root->lchild = newnode;
                        else
             insert(root->lchild, newnode);
        if (newnode->data > root->data)
        {
                if (root->rchild == NULL)
                        root->rchild = newnode;
                else
           insert(root->rchild, newnode);
      }
}

  void search(NODE root)
{     int key; NODE cur;
        if(root == NULL)
        {
                printf("\nBST is empty."); return;
          }
        printf("\nEnter Element to
        be searched: "); scanf("%d",
        &key);
        cur = root;
        while (cur != NULL)
        {
                if (cur->data == key)
                {
                        printf("\nKey element is
                        present in BST"); return;
                }
                if (key < cur->data)
                        cur = cur->lchild;
              else
                        cur = cur->rchild;

         }

        printf("\nKey element is not found in the BST");
}

void inorder(NODE root)
{
        if(root != NULL)
        {
```

```
                inorder(root->lchild);
                printf("%d ", root->data);

                inorder(root->rchild);
        }
}

void preorder(NODE root)
{
        if (root != NULL)
        {
                printf("%d ", root->data);
                preorder(root->lchild);
                 preorder(root->rchild);
        }
}

void postorder(NODE root)
{
        if (root != NULL)
        {
                postorder(root->lchild);
                 postorder(root->rchild);
                printf("%d ", root->data);
        }
}

void main()
{
        int ch, key, val, i, n;
        NODE root =
        NULL, newnode;
        while(1)
        {
                printf("\n~~~~BST
                MENU~~~~");
                printf("\n1.Create a
                BST");
                printf("\n2.Search")
                ; printf("\n3.BST
                Traversals: ");
                printf("\n4.Exit");
                printf("\nEnter your
                choice: ");
                scanf("%d", &ch);
                switch(ch)
```

```
                {
                        case 1:         printf("\nEnter the
                                        number of elements: ");
                                        scanf("%d", &n);
                                        for(i=1;i<=n;i++)
                                        {newnode = create();
                                           if (root == NULL)
                                                root = newnode;
                                            else
                                                insert(root, newnode);
                                        }
                                        break;
                                        case 2:     if (root == NULL)
                                                    printf("\nTree Is Not Created");


                                        else
                                        {  printf("\nThe Preorder display : ");
                                           preorder(root);
                                           printf("\nThe Inorder display : ");
                                           inorder(root);
                                           printf("\nThe Postorder display : ");
                                           postorder(root);

                                        }
                                        break;
                        case 3:     search(root);
                                    break;

                        case 4:     exit(0);
                }
        }
}
```

Output:

~~~~BST MENU~~~~ 1.Create a BST 2.Search
3. BST Traversals:
4. Exit
Enter your choice: **1**

Enter the number of elements: **12**
Enter The value: **6** Enter The value: **9** Enter The value: **5**
Enter The value: **2** Enter The value: **8** Enter The value: **15**
Enter The value: **24** Enter The value: **14** Enter The value: **7**
Enter The value: **8**
~~~~

Enter The value: **5**
Enter The value: **2**

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: **3**

The Preorder display:	**6**	**5**	**2**	**9**	**8**	**7**	**15**	**14**	**24**
The Inorder display:	**2**	**5**	**6**	**7**	**8**	**9**	**14**	**15**	**24**
The Postorder display:	**2**	**5**	**7**	**8**	**14**	**24**	**15**	**9**	**6**

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2

Enter Element to be searched: 66 Key element is not found in the BST

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2

Enter Element to be searched: 14 Key element is present in BST

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 4

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.

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

int a[50][50], n, visited[50];
int q[20], front = -1,rear = -1;
int s[20], top = -1, count=0;

void bfs(int v)
{
        int i, cur; visited[v] = 1; q[++rear] = v;
        while(front!=rear)
        {
                cur = q[++front];
                for(i=1;i<=n;i++)
                {
                        if((a[cur][i]==1)&&(visited[i]==0))
                        {
                                q[++rear] = i; visited[i] = 1;
                                 printf("%d ", i);
                        }
                }
        }
}

void dfs(int v)
{
        int i; visited[v]=1; s[++top] = v;
        for(i=1;i<=n;i++)
        {
                if(a[v][i] == 1&& visited[i] == 0 )
                {printf("%d ", i); dfs(i);
                 }
        }
}
```

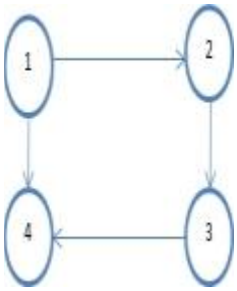
```
int main()
{
        int ch, start, i,j;
        printf("\nEnter the number of
        vertices in graph: ");
        scanf("%d",&n);
        printf("\nEnter the adjacency
        matrix:\n"); for(i=1; i<=n;
        i++)
        {
             for(j=1;j<=n;j++)
                  scanf("%d",&a[i]
                  [j]);
         }

    for(i=1;i<=n;i++) visited[i]=0;
    printf("\nEnter the starting vertex: ");
    scanf("%d",&start);

       printf("\n==>1. BFS: Print all nodes reachable from a
       given startingnode");
       printf("\n==>2. DFS: Print all nodes reachable from a
       given startingnode");
       printf("\n==>3:Exit");
       printf("\nEnter your choice: "); scanf("%d",
       &ch);
       switch(ch)
       {
          case 1: printf("\nNodes reachable from starting vertex %d are: ", start);
                      bfs(start);
                for(i=1;i<=n;i++)
                {
                   if(visited[i]==0)
                      printf("\nThe vertex that is not reachable is %d" ,i);
                }
                break;
          case 2: printf("\nNodes reachable from starting
                vertex %d are:\n",start); dfs(start);
          break;
          case 3: exit(0);
```

```
            default: printf("\nPlease enter valid choice:");
        }
    }
```

Output:



Case 1:

Enter the number of vertices in graph: **4**

Enter the adjacency matrix:

0 1 0 1
0 0 1 0
0 0 0 1
0 0 0 0

~~~Menu~~~~

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: **1**

Enter the starting vertex: 1

Nodes reachable from starting vertex 1 are: 24 3

Case 2:

Enter the number of vertices in graph: **4**

Enter the adjacency matrix:

0 1 0 1
0 0 1 0
0 0 0 1

0 0 0 0

~~~Menu~~~~

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: **1**

Enter the starting vertex: 2

Nodes reachable from starting vertex 2 are: 3 4 The vertex that is not reachable is 1

Case 3:

Enter the number of vertices in graph: **4**

Enter the adjacency matrix:

0 1 0 1
0 0 1 0
0 0 0 1
0 0 0 0

~~~Menu~~~~

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: **2**

Enter the starting vertex: 1

Nodes reachable from starting vertex 1 are: 2 3 4

Case 4:

Enter the number of vertices in graph: **4**

Enter the adjacency matrix:

0 1 0 1
0 0 1 0
0 0 0 1
0 0 0 0

~~~Menu~~~~

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: **2**

Enter the starting vertex: 2

Nodes reachable from starting vertex 2 are: 3 4

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

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

```
  int key[20],n,m; int *ht,index; int count = 0;

void insert(int key)
{
        index = key % m;
        while(ht[index] != -1)
        {
                index = (index+1)%m;
        }
        ht[index] = key;
        count++;
}

void display()
{
        int i;
        if(count == 0)
        {
                printf("\nHash Table is empty"); return;
        }

        printf("\nHash Table contents are:\n "); for(i=0;
        i<m; i++)
                printf("\n T[%d] --> %d ", i, ht[i]);
}

void main()

{
        int i;
        printf("\nEnter the number of employee
        records (N) : "); scanf("%d", &n);

        printf("\nEnter the two digit memory locations (m)
        for hash table: "); scanf("%d", &m);

        ht = (int *)malloc(m*sizeof(int)); for(i=0; i<m; i++)
                ht[i] = -1;

        printf("\nEnter the four digit key values (K) for N Employee
        Records:\n "); for(i=0; i<n; i++)
```

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

        for(i=0;i<n;i++)
        {
                if(count == m)
                {
                    printf("\n~~~Hash table is full. Cannot insert the record %d
                    key~~~",i+1); break;
                }
                insert(key[i]);
    }

          //Displaying Keys inserted into hash table
           display();
}
```

Output:

Enter the number of employee records (N) : **12**

Enter the two digit memory locations (m) for hash table: **15**

Enter the four digit key values (K) of 'N' Employee Records:
1234
5678
3456
2345
6799
1235
7890
3214
3456
1235
5679
2346

Hash Table contents are:

T[0] --> 7890
T[1] --> -1
T[2] --> -1
T[3] --> -1
T[4] --> 1234
T[5] --> 2345
T[6] --> 3456

T[7] --> 6799
T[8] --> 5678
T[9] --> 1235
T[10] --> 3214
T[11] --> 3456
T[12] --> 1235
T[13] --> 5679
T[14] --> 2346

Viva Questions

1) What is a Data Structure?
2) What are the types of Data Structures. Give examples.
3) What is a Singly Linked List?
4) What is Doubly Linked List?
5) Differentiate Array and Linked List.
6) What is a Stack?
7) What is stack overflow and underflow?
8) What are the Applications of Stack?
9) What is Queue Data structure?
10) Differentiate between Linear Queue and Circular Queue.
11) What is Binary search Tree?
12) What is Hashing Technique? Why it is Used.
13) How can we resolve the collision with linear probing?
14) Whether linked list is linear or non linear data structure.
15) Differentiate between file and structure storage structure.
16) What are multidimensional arrays?
17) What is BFS (Breadth First Search) Method?
18) What is DFS (Depth First Search) Method?
19) What is the difference between a PUSH and a POP?
20) What is a postfix expression?