
**An Autonomous Institute**
**NEAR ITPB, CHANNASANDRA, BENGALURU – 560 067**
**Affiliated to VTU, Belagavi**
**Approved by AICTE, New Delhi**
**Recognized by UGC under 2(f) & 12(B)**
**Accredited by NBA & NAAC**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**III SEMESTER**
**DATA STRUCTURES LABORATORY**
**MVJ22CSL34**
**ACADEMIC YEAR 2024 – 2025(ODD)**

# LABORATORY MANUAL

**NAM E OF THE STUDENT** : ----------

**BRANCH** : ----------

**UNIVERSITY SEAT NO.** : ----------

**SEMESTER & SECTION** : ----------

**BATCH** :

----------

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## VISION:

To create an ambience in excellence and provide innovative emerging programs in Computer Science and Engineering and to bring out future ready engineers equipped with technical expertise and strong ethical values.

## MISSION:

1. **Concepts of computing discipline**: To educate students at under graduate, postgraduate and doctoral levels in the fundamental and advanced concepts of computing discipline.

2. **Quality Research**: To provide strong theoretical and practical background across the Computer Science and Engineering discipline with the emphasis on computing technologies, quality research, consultancy and trainings.

3. **Continuous Teaching Learning:** To promote a teaching learning process that brings advancements in Computer Science and Engineering discipline leading to new technologies and products.

4. **Social Responsibility and Ethical Values:** To inculcate professional behavior, innovative research Capabilities, leadership abilities and strong ethical values in the young minds so as to work with the commitment for the betterment of the society.

## Programme Educational Objectives (PEOs):

**PEO01**: **Current Industry Practices**: Graduates will analyze real world problems and give solution using current industry practices in computing technology.

**PEO02**: **Research & Higher Studies:** Graduates with strong foundation in mathematics and engineering fundamentals will pursue higher learning, R&D activities and consultancy.

**PEO03**: **Social Responsibility:** Graduates will be professionals with ethics, who will provide industry growth and social transformation as responsible citizens.

**PEO4: Entrepreneur:** Graduates will be able to become entrepreneur to address social, technical, and business challenges.

**Programme Outcomes (POs):**

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

2. **Problem analysis**: Identify, formulate, 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 with an understanding of the limitations.

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

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

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

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

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Program Specific Outcome:**

***PSO1 : Programming:*** Ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, DBMS, and networking for efficient design of computer-based systems of varying complexity.

***PSO2: Practical Solution:*** Ability to practically provide solutions for real world problems with a broad range of programming language and open source platforms in various computing domains.

**Course objectives:**

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

- Asymptotic performance of algorithms.
- Linear data structures and their applications such as stacks, queues and lists
- Non-Linear data structures and their applications such as trees and graphs
- Sorting and searching algorithms

**Course outcomes:**

| CO No | CO's |
|---|---|
| CO1 | Explain different data structures and their applications. |
| CO2 | Apply Arrays, Stacks and Queue data structures to solve the given problems |
| CO3 | Use the concept of linked list in problem solving. |
| CO4 | Develop solutions using trees and graphs to model the real-world problem |
| CO5 | Explain the advanced Data Structures concepts such as Hashing Techniques and OptimalBinary Search Trees |

**Prerequisites:**

- Basics of Computers
- Knowledge of C language

# CONTENTS

| | | |
|---|---|---|
| 5. | Develop a Program in C for the following Stack Applications<br>a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^<br>Solving Tower of Hanoi problem with n disks | **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)<br>a. Insert an Element on to Circular QUEUE<br>b. Delete an Element from Circular QUEUE<br>c. Demonstrate Overflow and Underflow situations on Circular QUEUE<br>d. Display the status of Circular QUEUE<br>e. Exit<br>Support the program with appropriate functions for each of the above operations | **L3** |
| 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,<br>PhNo<br>a. Create a SLL of N Students Data by using front insertion.<br>b. Display the status of SLL and count the number of nodes in it<br>c. Perform Insertion / Deletion at End of SLL<br>d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)<br>Exit | **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,<br>Sal, PhNo<br>a. Create a DLL of N Employees Data by using end insertion.<br>b. Display the status of DLL and count the number of nodes in it<br>c. Perform Insertion and Deletion at End of DLL<br>d. Perform Insertion and Deletion at Front of DLL<br>e. Demonstrate how this DLL can be used as Double Ended Queue.<br>Exit | **L3** |
| 9. | Develop a Program in C for the following operationson Singly Circular Linked List (SCLL)with header nodes.<br>a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$<br>b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store theresult in POLYSUM(x,y,z)<br>Support the program with appropriate functions for each of the above operations. | **L3** |
| 10. | Develop a menu driven Program in C for the following operations on Binary Search Tree(BST) of Integers.<br>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<br>c. Search the BST for a given element (KEY) and report the appropriate message.<br>Exit | |
| 11. | Develop a Program in C for the following operations on Graph(G) of Cities<br>a. Create a Graph of N cities using Adjacency Matrix.<br>Print all the nodes reachable from a given starting node in a digraph using DFS/BFSmethod | |
| 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:<br>K →L as H(K)=K mod m (remainder method), and implement hashing<br>technique to map a given key K to the address space L. Resolve the collision (if any) usinglinear probing. | **L3** |

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

**PROGRAM:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define NUM_DAYS_IN_WEEK 7// 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;
void fnFreeCal(DAYTYPE *);
void fnDispCal(DAYTYPE *);
void fnReadCal(DAYTYPE *);
DAYTYPE *fnCreateCal();
int main()
{
   // Create the calendar
   DAYTYPE *weeklyCalendar = fnCreateCal();
   // Read data from the keyboard
   fnReadCal(weeklyCalendar);
   // Display the week's activity details
   fnDispCal(weeklyCalendar);
   // Free allocated memory
   fnFreeCal(weeklyCalendar);
return 0;
}
DAYTYPE *fnCreateCal()
{
     DAYTYPE *calendar=(DAYTYPE*)malloc(NUM_DAYS_IN_WEEK *
  sizeof(DAYTYPE));
```

```
for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
  {
      calendar[i].acDayName = NULL;
      calendar[i].iDate = 0;
      calendar[i].acActivity = NULL;
   }
   return calendar;
}
void fnReadCal(DAYTYPE *calendar)
{
  char cChoice;
   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
  {
      printf("Do you want to enter details for day %d [Y/N]: ", i + 1);
      scanf("%c", &cChoice); getchar();

      if(tolower(cChoice) == '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("Activity: ");
      char activityBuffer[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
   }
}
void fnDispCal(DAYTYPE *calendar)
{
   printf("nWeek's Activity Details:n");
   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
  {
      printf("Day %d:n", i + 1);
          if(calendar[i].iDate == 0)
          {
printf("No Activitynn");
                  continue;
          }
      printf("  Day Name: %sn", calendar[i].acDayName);
      printf("  Date: %dn", calendar[i].iDate);
      printf("  Activity: %snn", calendar[i].acActivity);
}
```

```
}
void fnFreeCal(DAYTYPE *calendar)
{
   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
{
      free(calendar[i].acDayName);
      free(calendar[i].acActivity);
   }
   free(calendar);
}
```

## Sample Output:

```
Do you want to enter details for day 1 [Y/N]: N
Do you want to enter details for day 2 [Y/N]: Y
Day Name: Monday
Date: 10
Activity: Meeting with Chairman.
Do you want to enter details for day 3 [Y/N]: N
Do you want to enter details for day 4 [Y/N]: N
Do you want to enter details for day 5 [Y/N]: Y
Day Name: Thursday
Date: 13
Activity: Product Survey
Do you want to enter details for day 6 [Y/N]: Y
Day Name: Friday
Date: 14
Activity: Budget Breakdown and Planning

Do you want to enter details for day 7 [Y/N]: Y
Day Name: Saturday
Date: 15
Activity: Outing with family

Week's Activity Details:
Day 1:
No Activity

Day 2:
  Day Name: Monday

 Date: 10
Activity: Meeting with Chairman.
```

Day 3:
No Activity

Day 4:
No Activity

Day 5:
Day Name: Thursday
Date: 13
Activity: Product Survey

Day 6:
Day Name: Friday
Date: 14
Activity: Budget Breakdown and Planning

Day 7:
Day Name: Saturday
Date: 15
Activity: Outing with family

**VIVA Questions:**

1. What is an array & how many types of arrays represented in memory?
2. How can be declared an array?
3. How can be insert an element in an array?
4. How can be delete an element in an array?
5. How many types of implementations of a two-dimensional array?
6. What is array of pointers?
7. What are limitations of array?
8. Where the elements of the array are stored respectively in successive?
9. How can merge two arrays?
10. What are the operations of array?

**2. Develop a Program in C for the following operations on Strings.**

**a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)**

**b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR Support the program with functions for each of the above operations. Don't use Built-in functions**

**PROGRAM:**

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

int main()
{
   char acMainStr[200], acSrchStr[30], acRepStr[30], acResStr[200], acCopyStr[200];
   int i=0, j=0 ,k=0, l, iMtchCnt, iStop, len, iNumOfMatch=0;

   printf("nEnter the main stringn");
  scanf(" %[^n]", acMainStr);

   printf("nEnter the Pattern stringn");
  scanf(" %[^n]", acSrchStr);

   printf("nEnter the Replace stringn");
  scanf(" %[^n]", acRepStr);

   strcpy(acCopyStr, acMainStr);
   for(i=0;i<(strlen(acMainStr)-strlen(acSrchStr)+1);i++)
   {
      iMtchCnt = 0;
      for(j=0;j<strlen(acSrchStr);j++)
      {
         if(acMainStr[i+j] == acSrchStr[j])
         {
            iMtchCnt++;
         }
         else
         {
            break;
         }
         if(iMtchCnt == strlen(acSrchStr))   //Check if number of character matches equals length
of pattern string
         {
            iNumOfMatch++;      //update number of total matches by 1
//             printf("nMatch occured at %d position in textn", i+1);
            for(k=0;k<i;k++)
```

```
        {
                    acResStr[k] = acMainStr[k];     //copy till the ith character where the match occured
                }
                iStop = k + strlen(acSrchStr); //point from where rest of the original string has to be
        copied
                acResStr[k] = '';
                strcat(acResStr, acRepStr); // append the replacement string
                len = strlen(acResStr);
                for(k=iStop, l=0; acMainStr[k] != '';k++, l++) //copy rest of original string
                {
                   acResStr[len+l] = acMainStr[k];
                }
                acResStr[len+l] = '';
        //              printf("n%s",acResStr);
                strcpy(acMainStr,acResStr);
              }
            }

          }
          printf("nInput Textn");
          printf("%sn",acCopyStr);

          if(iNumOfMatch > 0)
          {
             printf("n%d matches occurred nn Text after replacing matched patterns is shown below n",
        iNumOfMatch);
             printf("n%sn",acResStr);
          }
          else
          {
             printf("nPattern String not found in Textn");
          }
          return 0;
        }
```

## Sample Output:

```
Enter the main string
Raju and Ramu went to shop to buy milk and cakes.
Enter the Pattern string
and
Enter the Replace string
&
Input Text
Raju and Ramu went to shop to buy milk and cakes.
2 matches occured
Text after replacing matched patterns is shown below
Raju & Ramu went to shop to buy milk & cakes.
```

## VIVA Questions:

1 Write a program to check whether the given string is a Palindrome.
2 Write a program to count vowels in the given string
3 Write a program to copy a string into another string.
4 Write a program to delete the leading space in the string
5 Write a program to count the space in the given string
6 Demonstrate simple arithmetic operations on Strings
7 Write a program to find the reverse of a given 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**

**PROGRAM:**

```
#include<stdio.h>
#include<stdlib.h>
#define max 20
int top=-1;
int a[10];
void push();
void pop();
void display(); void
palindrome(); void
main()
{
int ch;
while(1)
{
printf("\n1.Push\n2.Pop\n3.Display\n4.Check for palindrom\nEnter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
        {
        push();
        break;
        }
case 2:
        {
        pop();
        break;
        }
case 3:
        {

        display();
        break;
        }
```

```
case 4:  {
         palindrome();
         break;
         }
default:
         exit(0);
   }
   }
}
void push()
{
int item;
if(top>=max)
         {
         printf("\nStack overflow….\n");
         }
else
         {
         top+=1;
         printf("Enter element:");
         scanf("%d",&item);
         a[top]=item;
         }
}
void pop()
{
int item;
if(top<0)
         {
         printf("\nStack underflow…\n");
         }
else
         {
         item=a[top];
         top-=1;
         printf("\nValue deleted is:%d",item);
         }
}
```

```
void display()
{
int i;
if(top<0
)
{
printf("\nStack underflow……");
}
   else
        {
        for(i=0;i<=top;i++)
        {
        printf("\n%d",a[i]);
        }
        }
}

void palindrome()
{
int i,
temp=top;
if(top!=-1)
         {
              for(i=0;i<=top/2;i++)
                   if(a[i]!=a[temp--])
                   {
                   printf(“the Stack is not
                   palindrome”);
                    return;
                   }
              printf(“the Stack is palindrome”);
        }
return;
}
```

## Sample Output:

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice: 1**
**Enter element: 23**

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice: 1**
**Enter element: 34**

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice: 1**
**Enter element: 56**

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice: 3**
**Enter element: 23**

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice: 2**

**Value deleted is:56**

**1. push**
**2. pop**
**3. display**
**4. Check for palindrome**
**Enter your choice:**

**VIVA Questions:**

1. What is stack?
2. What are the operations performed on stack?
3. What is push operation?
4 What is pop operation?
5. How stacks are implemented?
6. What are the applications of stack?
7. What is recursion?
8. What are the different types of stack?
9. Define "Top of stack"**.**

**4. Develop a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands**

**PROGRAM:**

```
#include<stdio.h>
#include<string.h>
char infix[50],suffix[50],stack[25]; int
top=-1;
void convert_fun(); int
precedence(char); void
push(char); char pop();
main()
{
        printf("\n\tENTER INFIX EXPRESSION : ");
        scanf("%s",infix);
        convert_fun();
        printf("\n\tTHE INFIX EXPRESSION IS: %s",infix);
        printf("\n\tTHE SUFFIX EXPRESSION IS : %s",suffix);
}
void convert_fun()
{
        int
        i=0,j=0;
        char
        c,tmp;
        push('#');
        while(i<strlen(infix))
        {
          c=infix[i++
          ]; switch(c)
          {
              case '(': push(c); break;

              case ')': tmp=pop();
                        while(tmp!='(')
                        {
                          suffix[j++]=tm
                          p; tmp=pop();
                        }
                        break;
              case '+':
              case '-':
              case '*':
```

```
        case '/':
        case '^': while(precedence(stack[top])>=precedence(c))
                  {
                    tmp=pop();
                    suffix[j++]=tm
                    p;
                  }
                  push(c
                  );
                  break;
        default :
                  suffix[j++]=c
                  ; break;
    }
}
while(top>0)
{
  tmp=pop();
  suffix[j++]=tm
  p;
}
}
void push(char c)
{
          stack[++top]=c;
}
char pop()
{
         return(stack[top--]);
}
int precedence(char c)
{
switch(c)
{
  case '^':
  return(3); case
  '*':
  case '/': return(2);
  case '+':
  case '-':
  return(1); case
  '(':
  case ')': return(0);
  case '#': return(-1);
}
return(0);
}
```

## Sample Output:

ENTER INFIX EXPRESSION : A+B
THE INFIX EXPRESSION IS: A+B
THE SUFFIX EXPRESSION IS :AB+

## VIVA Questions:

1. What is Stack?
2. Application of Stack?
3. Terms used in Stack?
4. Explain infix in Stack?
5. Explain Postfix in Stack?
6. Define operations on Stack?
7. Give postfix form for (A+B) *C/D
8. Give postfix form for A+B/C-D
9. Give prefix form for A/B^C+D.
10. Give prefix form for A*B+C.

**5. Develop a program in C for the following Stack Applications.**

**a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^**

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

**PROGRAM:**

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

#define STK_SIZE 10
void fnPush(int [], int*, int);
int fnPop(int [], int*);
int main()
{
    int iaStack[50], i, iOp1, iOp2, iRes;
    char acExpr[50], cSymb;
    int top = -1;

    printf("nEnter a valid postfix expressionn");
    scanf("%s", acExpr);

    for(i=0; i<strlen(acExpr); i++)
    {
        cSymb = acExpr[i];
        if(isdigit(cSymb))
        {
            fnPush(iaStack, &top, cSymb-'0');
        }
        else
        {
            iOp2 = fnPop(iaStack, &top);
            iOp1 = fnPop(iaStack, &top);
            switch(cSymb)
            {
                case '+' :  iRes = iOp1 + iOp2;
                            break;
                case '-' :  iRes = iOp1 - iOp2;
                            break;
                case '*' :  iRes = iOp1 * iOp2;
```

```
break;
                        case '/' :      iRes = iOp1 / iOp2;
                                        break;
                        case '%' :      iRes = iOp1 % iOp2;
                                        break;
                        case '^' :      iRes = (int)pow(iOp1 , iOp2);
                                        break;
                }
                fnPush(iaStack, &top, iRes);
        }

    }
    iRes = fnPop(iaStack, &top);
    printf("nValue of %s expression is %dn", acExpr, iRes);
    return 0;
}

void fnPush(int Stack[], int *t , int elem)
{
    *t = *t + 1;
    Stack[*t] = elem;
}

int fnPop(int Stack[], int *t)
{
    int elem;
    elem = Stack[*t];
    *t = *t -1;
    return elem;
}
```

**Sample Output**

```
Enter a valid postfix expression
45+95-*

Value of 45+95-* expression is 36

putta:~/.../Programs$ ./a.out

Enter a valid postfix expression
459*+62-+

Value of 459*+62-+ expression is 53
```

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

    **Support the program with appropriate functions for each of the above operations.**

**PROGRAM:**

```
#include <stdio.h>
#include<stdlib.h>
//#include<stdio_ext.h>
#define MAX 5
int rb[MAX];
int front = -1,rear = -1;
void inset(int);
void delete();
void display();
int main()
{
int ch; int
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(" %d", &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(int item)
{
if(front ==(rear+1)%MAX)
{
printf("\n\n~~Ring Buffer Overflow~~");
}
else
{
if(front == -1) front =
rear = 0; else
rear = (rear+1)%MAX; rb[rear]
= item;
}
}
void delete()
{
int item; if(front == -
1)
{
printf("\n\n~~Ring Buffer Underflow~~");
}
else
{
item = rb[front];
if(front == rear) //only one element front =
rear = -1;
else
front = (front+1)%MAX;
printf("\n\nDeleted element from the queue is: %d ", item );
}
}
void display()
{
int i ;
if(front ==-1)
{
printf("\n\nCircular Queue Empty"); return;
}
else
{
printf("\nCircular Queue contents are:\n");
printf("\nFront[%d]-> ", front); for(i=front; i!=rear
; i=(i+1)%MAX)
{
```

```
printf(" %d", rb[i]);
}
printf(" %d", rb[i]);
printf(" <-[%d]Rear", rear);
printf("\n");
}
}
}
```

## Sample 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: 1


~~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: 2


~~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: 3


~~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: 4

~~Ring Buffer 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]-> 1 2 3 <-

[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: 1

~~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: 2

~~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: 3

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

~~Ring Buffer Underflow~~

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

**VIVA Questions:**

1. What is Circular Queue?
2. Why use of Circular Queue?
3. Explain Dequeue?
4. What is Queue?
5. Variation in a Queue?
6. What is Priority Queue?
7. Application of Queue?
8. What is Queue implementation?

Define operations on queue?

**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**

**e. Exit**

**PROGRAM:**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h> struct
node
{
int usn,sem,phno; char
name[20]; char
branch[20]; struct node
*link;
}*start;
void create(int,int,int,char[],char[]); void
disp();
void insertfront(int,int,int,char[],char[]); void
deletefront();
void insertend(int,int,int,char[],char[]); void
deleteend();
        main()
{
int ch,n,i,m,a,pos; int
usn,sem,phno;
char name[20],branch[20];
start=NULL;
do
{
printf("\n\nMENU\n\n");
printf("\n1.CREATE\n");
printf("\n2.DISPLAY\n");
printf("\n3.INSERT FRONT\n");
printf("\n4.DELETE FRONT \n");
printf("\n5.INSERT END\n");
printf("\n6.DELETE END\n");
printf("\n7.EXIT\n");

printf("\nENTER UR CHOICE\n");
scanf("%d",&ch);
switch(ch)
{
```

```
case 1:
printf("\n\nHOW MANY NODES U WANT TO CREATE\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nENTER USN");
scanf("%d",&usn); printf("\nENTER
NAME"); scanf("%s",name);
printf("\nENTER branch");
scanf("%s",branch); printf("\nENTER
SEM"); scanf("%d",&sem);
printf("\nENTER Phone no");
scanf("%d",&phno);
create(usn,sem,phno,name,branch);
}
break;
case 2:
disp();
break;
case 3:
printf("\nENTER USN");
scanf("%d",&usn); printf("\nENTER
NAME"); scanf("%s",name);
printf("\nENTER branch");
scanf("%s",branch); printf("\nENTER
SEM"); scanf("%d",&sem);
printf("\nENTER Phone no");
scanf("%d",&phno);
insertfront(usn,sem,phno,name,branch); break;
case 4: deletefront();
break;
case 5:
printf("\nENTER USN");
scanf("%d",&usn);
printf("\nENTER NAME");
scanf("%s",name);
printf("\nENTER branch");
scanf("%s",branch);
printf("\nENTER SEM");
 scanf("%d",&sem);
printf("\nENTER Phone no");
scanf("%d",&phno);
insertend(usn,sem,phno,name,branch);
break;
case 6:
deleteend(); break;
case 7:
exit(0);
```

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

}

void create(int us,int se,int ph,char na[],char br[])
{
struct node *q,*tmp;
        tmp=(struct node *)malloc(sizeof(struct node));
        tmp->usn=us;
        tmp->sem=se;
        tmp->phno=ph;
        strcpy(tmp->name,na);
        strcpy(tmp->branch,br);
        tmp->link=NULL;

          if(start==NULL)
          {
            start=tmp;

          }
          else
          {
            q = start;
              while(q->link!=NULL)
              {
              q = q->link;
              }
            q->link = tmp;
          }
        }
        void disp()

        {
        struct node *q;
        int count=0;
        if(start==NULL)


        {
        printf("\n\nLIST IS EMPTY");
        }
        else
        {
        q=start;
        while(q!=NULL)
        {
        count++;
        printf("\nUSN %d",q->usn);
```

```
printf("\nSEM %d",q->sem);
printf("\nPH NO %d",q->phno);
printf("\nNAME %s",q->name);
printf("\nBRANCH %s",q->branch);
q=q->link;
}
printf("NULL");
printf("Total number of student=%d",count);
}
}
void insertfront(int us,int se,int ph,char na[],char br[])
{
        struct node *tmp;
        tmp=(struct node *)malloc(sizeof(struct node));
        tmp->usn=us;
        tmp->sem=se;
        tmp->phno=ph;
        strcpy(tmp->name,na);
        strcpy(tmp->branch,br);
        tmp->link=NULL;
        if(start==NULL)
        {
                start=tmp;
        }
        else
        {
                tmp->link=start;
                start=tmp;
        }
                                                }
        void insertend(int us,int se,int ph,char na[],char br[])
        {
        struct node *q,*tmp;
        tmp=(struct node *)malloc(sizeof(struct node));
        tmp->usn=us;
        tmp->sem=se;
        tmp->phno=ph;

        strcpy(tmp->name,na);

        strcpy(tmp->branch,br);
        tmp->link=NULL;

        if(start==NULL)
        {
         start=tmp;

        }
```

```
  else
  {
    q = start;
      while(q->link!=NULL)
      {
      q = q->link;
      }
    q->link = tmp;
  }
}
void deletefront()
{
    struct node *tmp;
    tmp = start;
  if(start->link==NULL)
   {
     free(start);
     start=NULL;
   }
   else {

    start = start->link;
      free(tmp);
     }
            printf("nThe Element deleted Successfully ");
}
void deleteend()
{
  struct node *temp =start;
   struct node *t;
   if(start->link==NULL)
   {
     free(start);
     start=NULL;
   }
   else
   {
      while(temp->link != NULL)

      {
        t=temp;
        temp=temp->link;
      }
      free(t->link);
      t->link=NULL;
   }
printf("nThe Element deleted Successfully ");
}
```

## Sample Output:

MENU 1.CREATE
2.DISPLAY
3.INSERT FRONT
4.DELETE FRONT
5.INSERT END
6.DELETE END
7.EXIT
ENTER UR CHOICE 1
HOW MANY NODES U WANT TO CREATE 1
ENTER USN 1 ENTER
NAME abc
ENTER branch cs
ENTER SEM 6
ENTER Phone no 84753 MENU
1.CREATE
2.DISPLAY
3.INSERT FRONT
4.DELETE FRONT
5.INSERT END
6.DELETE END
7.EXIT
ENTER UR CHOICE 2
USN 1
SEM 6
PH NO 84753
NAME abc
BRANCH cs
Total number of student=1

**VIVA Questions:**

1. Mention what is Linked lists?
2. What type of memory allocation is referred for Linked lists?
3. Mention what is traversal in linked lists?
4. Mention what are the applications of Linked Lists?
5. What does the dummy header in linked list contain?
6. Mention what is the difference between singly and doubly linked lists?
7. Write an algorithm to insert a node in front of the SLL.
8. Write an algorithm to insert a node in end of the SLL
9. Write an algorithm to delete a node in SLL
10. Write an algorithm to display all the nodes in SLL

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

**a. Create a DLL of N Employees Data by using end insertion.**

**b. Display the status of DLL and count the number of nodes in it.**

**c. Perform Insertion and Deletion at End of DLL .**

**d. Perform Insertion and Deletion at Front of DLL .**

**e. Demonstrate how this DLL can be used as Double Ended Queue.**

**f. Exit**

**PROGRAM:**

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct employeeData
{
      struct employeeData *prev;
      int SSN;
      char name[20];
      char dept[20];
      char designation[20];
      float sal;
      char PhNo[50];
      struct employeeData *next;
  };
  typedef struct employeeData employee;
  employee *first=NULL,*last=NULL;
  int n;
  void CreateList(int n){
            employee *t;
            int i;

            first=(employee*) malloc (sizeof(employee)) ;
            first->prev=NULL;
            printf("\nEnter employee id:");
            scanf("%d",&first->SSN);
            printf("\nEnter the employee name:");
            scanf("%s",first->name);
            printf("\nEnter the employee department:");
            scanf("%s",first->dept);
            printf("\nEnter the employee designation:");
```

```
 scanf("%s",first->designation);
 printf("\nEnter employee salary:");
 scanf("%f",&first->sal);
 printf("\nEnter employee phone number:");
 scanf("%s",first->PhNo);
 first->next=NULL;
 last = first;

 for(i=1;i<n;i++) {
        t=(employee*) malloc (sizeof(employee));
        t->prev=last;
        last->next=t;
        printf("\nEnter employee id:");
        scanf("%d",&t->SSN);
        printf("\nEnter the employee name:");
        scanf("%s",t->name);
        printf("\nEnter the employee department:");
        scanf("%s",t->dept);
        printf("\nEnter the employee designation:");
        scanf("%s",t->designation);
        printf("\nEnter employee salary:");
        scanf("%f",&t->sal);
        printf("\nEnter employee phone number:");
        scanf("%s",t->PhNo);
        t->next=NULL;
        last = t;
 }

}

void display(){
 if ( first == NULL)
        printf("No Data is entered, list is empty \n");
 else {
        employee *t;
        t=first;
        while(t != NULL) {
                printf("\nEmployee id:%d",t->SSN);
                printf("\nEmployee name:%s",t->name);
                printf("\nEmployee department:%s",t->dept);
                printf("\nEmployee designation:%s",t->designation);
                printf("\nEmployee salary:%f",t->sal);
                printf("\nEmployee phone number:%s",t->PhNo);
                t=t->next;
                printf("\n\n");
        }
 }
}
```

```
void insertEnd(){
        employee *t = (employee*) malloc (sizeof(employee));
        printf("\nEnter employee id:");
        scanf("%d",&t->SSN);
        printf("\nEnter the employee name:");
        scanf("%s",t->name);
        printf("\nEnter the employee department:");
        scanf("%s",t->dept);
        printf("\nEnter the employee designation:");
        scanf("%s",t->designation);
        printf("\nEnter employee salary:");
        scanf("%f",&t->sal);
        printf("\nEnter employee phone number:");
        scanf("%s",t->PhNo);
        if(first == NULL) {
                t->prev=NULL;
                t->next=NULL;
                first=t;
                last = first;
        }
        else {
                t->next=NULL;
                t->prev=last;
                last->next=t;
                last=t;
        }
 }
void insertFront() {
        employee *t = (employee*) malloc (sizeof(employee));
        printf("\nEnter employee id:");
        scanf("%d",&t->SSN);
        printf("\nEnter the employee name:");
        scanf("%s",t->name);
        printf("\nEnter the employee department:");
        scanf("%s",t->dept);
        printf("\nEnter the employee designation:");
        scanf("%s",t->designation);
        printf("\nEnter employee salary:");
        scanf("%f",&t->sal);
        printf("\nEnter employee phone number:");
        scanf("%s",t->PhNo);
        if(first == NULL) {
                t->prev=NULL;
                t->next=NULL;
                first=t;
```

```
            last=first
        }
        else {
                t->next=first;
                t->prev=NULL;
                first->prev=t;
   }            first=t;
}

void deleteEnd() {
        employee *t = last;
        employee *d;
        d = last;
        t = last->prev;
        t->next=NULL;
        last = t;
        free(d);
}

void deleteFront(){
        employee *t = first;
        employee *d;
        d = first;
        t = first->next;
        t->prev=NULL;
        first=t;
        free(d);
}
  main(){
    int choice,key,info;
    while(1){
      printf("\nDLIST OPERATIONS\n");
      printf("1.Create a list \n ");
            printf("2.INSERT at End of DLL \n ");
            printf("3.INSERT at Front of DLL\n ");
            printf("4.DELETE at End of DLL \n ");
            printf("5.DELETE at Front of DLL \n ");
            printf("6. DISPLAY\n ");
            printf("7.EXIT\n");
      printf("Enter your choice:\n");
      scanf("%d",&choice);
      switch(choice) {
          case 1:                   printf("\nEnter the number of elements to be included in
                                    list:");
                                        scanf("%d",&n);
                                    CreateList(n); break;
```

```
                    case 2:
                            insertEnd();
                            break;
                    case 3:
                            insertFront();
                            break;
                    case 4:
                            deleteEnd();
                            break;
                    case 5:
                            deleteFront();
                            break;
                    case 6:
                            display();
                            break;
                    case 7:
                            exit(0);
        }
    }
}
```

**Sample Output:**

```
DLIST OPERATIONS
 1.Create a list
 2.INSERT at End of DLL
 3.INSERT at Front of DLL
 4.DELETE at End of DLL
 5.DELETE at Front of DLL
 6. DISPLAY
 7.EXIT
Enter your choice:1

Enter the number of elements to be included in list:1

Enter employee id:12

Enter the employee name:Preeti

Enter the employee department:cse

Enter the employee designation:AP

Enter employee salary:12345

Enter employee phone number:12345678
```

DLIST OPERATIONS
1. Create a list
2. INSERT at End of DLL
3. INSERT at Front of DLL
4. DELETE at End of DLL
5. DELETE at Front of DLL
6. DISPLAY
7. EXIT

Enter your choice:6

Employee id:12 Employee
name:Preeti Employee
department:cse Employee
designation:AP
Employee salary:12345.000000
Employee phone number:12345678

DLIST OPERATIONS
1. Create a list
2. INSERT at End of DLL
3. INSERT at Front of DLL
4. DELETE at End of DLL
5. DELETE at Front of DLL
6. DISPLAY
7. EXIT

Enter your choice:7

**VIVA Questions:**

1. Mention what is Linked lists?
2. What type of memory allocation is referred for Linked lists?
3. Advantages of DLL
4. Disadvantages of DLL
5. What does the dummy header in linked list contain?
6. Mention what is the difference between singly and doubly linked lists?
7. Write an algorithm to insert a node in front of the DLL.
8. Write an algorithm to insert a node in end of the DLL
9. Write an algorithm to delete a node in DLL
10. Write an algorithm to display all the nodes in DLL

**9. Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes**

a. **Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$**

b. **Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store theresult in POLYSUM(x,y,z)**

**PROGRAM**

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

// Node structure for a term in the polynomial
struct PolyTerm{
   int coefficient;
   int pow_x;
   int pow_y;
   int pow_z;
   struct PolyTerm* next;
};

typedef struct PolyTerm* POLYPTR;

POLYPTR fnInsertTerm(POLYPTR poly, int coef, int pow_x, int pow_y, int pow_z)
{
        POLYPTR cur;
   POLYPTR newNode = (POLYPTR)malloc(sizeof(struct PolyTerm));
   newNode->coefficient = coef;
   newNode->pow_x = pow_x;
   newNode->pow_y = pow_y;
   newNode->pow_z = pow_z;
   newNode->next = NULL;

   cur = poly;
   while(cur->next != poly)
   {
        cur = cur->next;
   }
        cur->next = newNode;
        newNode->next = poly;
        return poly;
}

void fnDispPolynomial(POLYPTR poly)
{
   if (poly->next == poly)
   {
      printf("Polynomial is empty.n");
```

```
return;
    }

    POLYPTR cur = poly->next;
    do
    {
        printf("%dx^%dy^%dz^%d ", cur->coefficient, cur->pow_x, cur->pow_y, cur->pow_z);
        cur = cur->next;
        if (cur != poly)
        {
            printf("+ ");
        }
    } while (cur != poly);
    printf("n");
}


int fnEvaluatePolynomial(POLYPTR poly, int x, int y, int z)
{
    int result = 0;
    if (poly->next == poly)
    {
        return result;
    }

    POLYPTR cur = poly->next;
    do
    {
        int termValue = cur->coefficient;
        termValue *= pow(x, cur->pow_x);
        termValue *= pow(y, cur->pow_y);
        termValue *= pow(z, cur->pow_z);

        result += termValue;

        cur = cur->next;
    } while (cur != poly);

    return result;
}

bool fnMatchTerm(POLYPTR p1, POLYPTR p2)
{
        bool bMatches = true;
        if(p1->pow_x != p2->pow_x)
                bMatches = false;
        if(p1->pow_y != p2->pow_y)
                bMatches = false;if(p1->pow_z != p2->pow_z)
```

```
if(p1->pow_z != p2->pow_z)
bMatches = false;

return bMatches;
}

POLYPTR fnAddPolynomials(POLYPTR poly1, POLYPTR poly2, POLYPTR polySum)
{
   POLYPTR cur1 = poly1->next;
   POLYPTR cur2 = poly2->next;

   do
   {
         polySum = fnInsertTerm(polySum, cur1->coefficient, cur1->pow_x, cur1->pow_y, cur1->pow_z);
         cur1 = cur1->next;
   }while(cur1 != poly1);

   do
   {
      cur1 = polySum->next;
      bool bMatchFound = false;
      do
      {
         if(fnMatchTerm(cur1, cur2))
         {
            cur1->coefficient += cur2->coefficient;
            bMatchFound = true;
            break;
         }
         cur1 = cur1->next;

      }while(cur1 != polySum);
      if(!bMatchFound)
      {
         polySum = fnInsertTerm(polySum, cur2->coefficient, cur2->pow_x, cur2->pow_y, cur2->pow_z);
      }

      cur2 = cur2->next;

   }while(cur2 != poly2);

   return polySum;

}

int main()
{
```

```
    POLYPTR poly1 = (POLYPTR)malloc(sizeof(struct PolyTerm));
    poly1->next = poly1;
    POLYPTR poly2 = (POLYPTR)malloc(sizeof(struct PolyTerm));
    poly2->next = poly2;

    POLYPTR polySum = (POLYPTR)malloc(sizeof(struct PolyTerm));
    polySum->next = polySum;

    // Represent and evaluate the polynomial P(x, y, z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3
    poly1 = fnInsertTerm(poly1, 6, 2, 2, 1);
    poly1 = fnInsertTerm(poly1, 4, 0, 1, 5);
    poly1 = fnInsertTerm(poly1, 3, 3, 1, 1);
    poly1 = fnInsertTerm(poly1, 2, 1, 5, 1);
    poly1 = fnInsertTerm(poly1, 2, 1, 1, 3);

    // Display the polynomial P(x, y, z)
    printf("POLY1(x, y, z) = ");
    fnDispPolynomial(poly1);

    // Read and evaluate the second polynomial POLY2(x, y, z)
    // Represent the polynomial P(x, y, z) = xyz + 4x^3yz
    poly2 = fnInsertTerm(poly2, 1, 1, 1, 1);  // Example term
    poly2 = fnInsertTerm(poly2, 4, 3, 1, 1);
    // Display the second polynomial POLY2(x, y, z)
    printf("POLY2(x, y, z) = ");
    fnDispPolynomial(poly2);
    // Add POLY1(x, y, z) and POLY2(x, y, z) and store the result in POLYSUM(x, y, z)
    polySum = fnAddPolynomials(poly1, poly2, polySum);
    // Display the sum POLYSUM(x, y, z)
    printf("nPOLYSUM(x, y, z) = ");
    fnDispPolynomial(polySum);

    // Evaluate POLYSUM(x, y, z) for specific values
    int x = 1, y = 2, z = 3;
    int iRes = fnEvaluatePolynomial(polySum, x, y, z);
    printf("nResult of POLYSUM(%d, %d, %d): %dn", x, y, z, iRes);
    return 0;
}
```

**Sample Output**

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

POLY2(x, y, z) = 1x^1y^1z^1 + 4x^3y^1z^1

POLYSUM(x, y, z) = 6x^2y^2z^1 + 4x^0y^1z^5 + 7x^3y^1z^1 + 2x^1y^5z^1 + 2x^1y^1z^3 + 1x^1y^1z^1

Result of POLYSUM(1, 2, 3): 2364

**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 recursively in inorder, preorder & postorder**

**c. Search the BST for a given element (KEY) and report the appropriate message**

## Program

```
# include <stdio.h> #
include <stdlib.h> struct
node {
        int value;
        struct node* left;
        struct node* right;
};
struct node* root;
struct node* insert(struct node* r, int data); void
inorder(struct node* r);
void preorder(struct node* r); void
postorder(struct node* r);
struct node* search(struct node ** tree, int val); struct
node* delete1(struct node* root, int key);
main() {
        int choice,key,i;
        struct node* tmp;
        root = NULL;
        int n, v;
        printf("\nProgram For Binary Search Tree ");
        printf("How many data's do you want to insert ?\n");
        scanf("%d", &n);
        while(1) {
          printf("\n1.Create");
          printf("\n2.Search");
          printf("\n3.Recursive Traversals");
          printf("\n4.Delete");
          printf("\n5.Exit");
          printf("\nEnter your choice :");
          scanf("%d", &choice);
          switch (choice) {
          case 1:

            for(i=0; i<n; i++){
                        printf("Data %d: ", i+1);
                        scanf("%d", &v);
                        root = insert(root, v);
              }
                  break;
```

```
            Case 2: printf("\nEnter Element to be searched :");
              scanf("%d", &key);
             tmp = search(&root, key);
             if (tmp) {
                    printf("Searched node=%d\n", tmp->value);
             }
             else {
                    printf("Data Not found in tree.\n");
             }
          break;
       case 3:
          if (root == NULL)
            printf("Tree Is Not Created");
          else {
            printf("\nThe Inorder display : ");
            inorder(root);
            printf("\nThe Preorder display : ");
            preorder(root);
            printf("\nThe Postorder display : ");
            postorder(root);
          }
          break;
      case 4:printf("\nEnter Element to be deleted :");
        scanf("%d", &key);
            tmp=delete1(root, key);
            if (tmp) {
                    printf("deleted node=%d\n", tmp->value);
            }
            else {
                    printf("Data Not found in tree.\n");
          }
       break;
       default:
       exit(0);
       }
     }
}
struct node* insert(struct node* r, int data)
{
     if(r==NULL)
     {
```

```
        r = (struct node*) malloc(sizeof(struct node));
        r->value = data;
        r->left = NULL;
        r->right = NULL;
    }
    else if(data < r->value){
        r->left = insert(r->left, data);
    }
    else {
        r->right = insert(r->right, data);
    }
    return r;

}

struct node* search(struct node ** tree, int val)
{
    if(!(*tree))
    {
        return NULL;
    }

    if(val < (*tree)->value)
    {
        search(&((*tree)->left), val);
    }
    else if(val > (*tree)->value)
    {
        search(&((*tree)->right), val);
    }
    else if(val == (*tree)->value)
    {
        return *tree;
    }
}

void inorder(struct node* r)
{
    if(r!=NULL){
        inorder(r->left);
        printf("%d ", r->value);
        inorder(r->right);
    }
}

void preorder(struct node* r)
{
    if(r!=NULL){
```

```
        printf("%d ", r->value);
        preorder(r->left);
        preorder(r->right);
    }
}

void postorder(struct node* r)
{
    if(r!=NULL){
        postorder(r->left);
        postorder(r->right);
        printf("%d ", r->value);

    }
}
struct node * minValueNode(struct node* node)
{
    struct node* current = node;

      while (current->left != NULL)
       current = current->left;

    return current;
}

 struct node* delete1(struct node* root, int key) {
    if (root == NULL) return root;

    if (key < root->value)
        root->left = delete1(root->left, key);

    else if (key > root->value)
        root->right = delete1(root->right, key);

    else {

        if (root->left == NULL)
        {
            struct node *temp = root->right;
              return temp;
        }
        else if (root->right == NULL)
        {
            struct node *temp = root->left;
             return temp;
        }
            struct node* temp = minValueNode(root->right);
```

```
        root->value = temp->value;

        root->right = delete1(root->right, temp->value);
    }
    return root;
}
```

## Sample Output:

Program For Binary Search Tree How many data's do you want to insert ?

12

1.Create

2.Search

3.Recursive Traversals

Enter your choice :1 Data

1: 6

Data 2: 9

Data 3: 5

Data 4: 2

Data 5: 8

Data 6: 15

Data 7: 24

Data 8: 14

Data 9: 7

Data 10: 8

Data 11: 5

Data 12: 2

1.Create

2.Search

3.Recursive Traversals

Enter your choice :3

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

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

1 Create

2 Search

3 Recursive Traversals

Enter your choice :2

Enter Element to be searched :7

Searched node=7

1.Create

2.Search

3.Recursive Traversals

Enter your choice :2

Enter Element to be searched :10

Data Not found in tree.

1.Create

2.Search

3.Recursive Traversals

Enter your choice :

**VIVA Questions:**

1. What do you understand by binary search?
2. Explain the steps of algorithm of binary search?
3. Define the complexity of binary search?
4. Explain the difference between linear search and binary search?
5. Define tree?
6. Explain tree terminology? 1> root, 2> node.
7. Explain the degree of node?
8. Explain the terminal node & nonterminal node?
9. What do you understand by binary tree?
10. Explain the application of binary tree?

**11. Develop a Program in C for the following operations on Graph(G) of Cities**

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

**b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method**

**Program**

```
#include<stdio.h>
#include<stdlib.h>
void BFS(int [20][20],int,int [20],int);
void DFS(int [20][20],int,int [20],int);
void main()
{
       int n,a[20][20],i,j,visited[20],source;
       printf("Enter the number of vertices:");
       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 source node:");
       scanf("%d",&source);
       visited[source]=1;
       BFS(a,source,visited,n);
       for(i=1;i<=n;i++)
       {
              if(visited[i]!=0)
                     printf("\n Node %d is reachable",i);
              else
                     printf("\n Node %d is not reachable",i);
       }
      DFS(a,source,visited,n);
       for(i=1;i<=n;i++)
       {
              if(visited[i]==0)
          {
                     printf("\nGraph is not connected");

                     exit(0);
              }
       }
```

```
        printf("\nGraph is connected\n");
}
void BFS(int a[20][20],int source,int visited[20],int n)
{
        int queue[20],f,r,u,v;
        f=0;
        r=-1;
        queue[++r]=source;
        while(f<=r)
        {
                u=queue[f++];
                for(v=1;v<=n;v++)
                {
                        if(a[u][v]==1 && visited[v]==0)
                        {
                    queue[++r]=v;
                    visited[v]=1;
                  }
          }
   }
}
        void DFS(int a[20][20],int u,int visited[20],int n)
{
        int v;
        visited[u]=1;
        for(v=1;v<=n;v++)
        {
                if(a[u][v]==1 && visited[v]==0)
                        DFS(a,v,visited,n);
        }
}
```

## Sample Output:

```
[exam@cpl1-8 ~]$ cc ds11.c
[exam@cpl1-8 ~]$ ./a.out
Enter the number of vertices:4

Enter the adjacency matrix:
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0

 Enter the source node:1
  Node 1 is reachable
  Node 2 is reachable
  Node 3 is reachable
```

Node 4 is reachable
Graph is connected

**VIVA Questions:**

1. What is a graph?
2. What do you mean by directed acyclic graph?
3. What is adjacent matrix?
4. What is a path?
5. What do you mean by a cycle?
6. What is a connected graph?
7. What is a degree of a graph?
8. What do you understand by the term complete graph?
9. What is a weighted graph?
10. What is a tree?

**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) usinglinear probing.**

**Program**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int tableSize = 0, totEle = 0; struct
node *hashTable = NULL; struct
node
   { int salary, key;
    char name[100];
    int marker;
    };
  void insertInHash(int key, char *name, int age)
  { int hashIndex = key % tableSize;
          if (tableSize == totEle)
                  {
                  printf("CAN'T PERFORM INSERTION..HASH TABLE IS FULL!!");
                          return;
                   }
  while (hashTable[hashIndex].marker == 1)
          { hashIndex = (hashIndex + 1)%tableSize;
           }
  hashTable[hashIndex].key = key;
  hashTable[hashIndex].salary = age;
  strcpy(hashTable[hashIndex].name, name);
  hashTable[hashIndex].marker = 1;
  totEle++;
  return;
}

void deleteFromHash(int key)
  {
int hashIndex = key % tableSize, count = 0, flag = 0;
if (totEle == 0)
```

```
            {
            printf("HASH TABLE IS EMPTY!!\n");
            return;
            }
    while (hashTable[hashIndex].marker != 0 && count <= tableSize) {
    if (hashTable[hashIndex].key == key)
            {
            hashTable[hashIndex].key = 0;
            /* set marker to -1 during deletion operation*/
            hashTable[hashIndex].marker = -1;
            hashTable[hashIndex].salary = 0;
            strcpy(hashTable[hashIndex].name, "\0");
            totEle--;
            flag = 1;
            break;
            }
    hashIndex = (hashIndex + 1)%tableSize;
    count++;
    }
    if (flag)
            printf("GIVEN DATA DELETED FROM HASH TABLE\n");
    else
                printf("GIVEN DATA IS NOT AVAILABLE IN HASH TABLE\n");
                return;
}
void searchElement(int key)
{
      int hashIndex = key % tableSize, flag = 0, count = 0;
      if (totEle == 0) {
      printf("HASH TABLE IS EMPTY!!");
      return;
      }
    while (hashTable[hashIndex].marker != 0 && count <= tableSize)
             {
              if (hashTable[hashIndex].key == key)
                    {
                    printf("EMPLOYEE ID : %d\n", hashTable[hashIndex].key);
                    printf("NAME : %s\n", hashTable[hashIndex].name);
                    printf("SALARY : %d\n", hashTable[hashIndex].salary);
                    flag = 1;
                    break;
                    }
            count++;
            hashIndex = (hashIndex + 1)%tableSize;
            }
    if (!flag)
            printf("GIVEN DATA IS NOT PRESENT IN HASH TABLE\n");
            return;
```

```
        }
void display()
{
int i;
        if (totEle == 0)
                {
                printf("HASH TABLE IS EMPTY!!\n");
                return;
                }
printf("EmployeeID  Name          Salary Index \n");
printf("..........\n");
for (i = 0; i < tableSize; i++) {
        if (hashTable[i].marker == 1) {
                printf("%-13d", hashTable[i].key);
                printf("%-15s", hashTable[i].name);
                printf("%-2d\t", hashTable[i].salary);
                printf(" %d\n", i);
                }
        }
        printf("\n");
        return;
        }

int checkduplicate(int key)
{
  int hashIndex = key % tableSize, flag = 0,count;
 while (hashTable[hashIndex].marker != 0 && count <= tableSize) {
   if (hashTable[hashIndex].key == key)
        {
         flag = 1;
        return 1;
        }
  count++;
hashIndex = (hashIndex + 1)%tableSize;
}
if (!flag)
        return 0;
}

int main()
{
int key, salary, ch;
char name[100];
printf("ENTER THE NO OF ELEMENTS:");
scanf("%d", &tableSize);
```

```
hashTable = (struct node *)calloc(tableSize, sizeof(struct node));
while (1
        {
        printf("1. INSERTION\t2. DELETION\n");
        printf("3. SEARCHIN\t4. DISPLAY\n");
        printf("5. EXIT\t\n ENTER YOUR CHOICE");
        scanf("%d", &ch);
        switch (ch)
        { case 1:
                printf("ENTER THE KEY VALUE BETWEEN 0 AND 9999:");
                scanf("%d", &key);
                if (key>9999)
                        {printf("KEY VALUE SHOULD BE <10000 \n");
                        break;
                        }
                if(checkduplicate(key))
                        {printf("KEY VALUE ALREADY EXIST\n");
                        break;
                        }
                else
                        {
                        getchar();
                        printf("NAME:");
                        fgets(name, 100, stdin);
                        name[strlen(name) - 1] = '\0';
                        printf("SALARY");
                        scanf("%d", &salary);
                        insertInHash(key, name, salary);
                        break;
                        }
        case 2:
                printf("ENTER THE KEY VALUE:");
                scanf("%d", &key);
                deleteFromHash(key);
                break;
        case 3:
                printf("ENTER THE KEY VALUE:");
                scanf("%d", &key);
                searchElement(key);
                break;
        case 4:
                display();
                break;
        case 5:
                exit(0);
                default:
                printf("U HAVE ENTERED WRONG OPTION!!\n");
```

```
                    break;
                    }
            }
return 0;
}
```

**Sample Output:**

```
ENTER THE NUMBER OF ELEMENTS:3
        1. INSERTION
        2. DELETION
        3.SEARCHING
        4.DISPLAY
        5.EXIT

ENTER YOUR CHOICE:1
ENTER THE KEY VALUE BETWEEN 0 AND 9999:34
NAME:ANITHA
SALARY:10000
1. INSERTION            2.DELETION
3.SEARCHING             4.DISPLAY
5.EXIT

ENTER YOUR CHOICE:1
ENTER THE KEY VALUE BETWEEN 0 AND 9999:35
NAME:PRANAV
SALARY:15000
ENTER YOUR CHOICE:4
EMPLOYEE  ID            NAME SALARY  INDEX
34                      ANITHA  10000       1
35                      PRANAV  15000       2

ENTER YOUR CHOICE:2 ENTER
THE KEY VALUE:40
GIVEN DATA IS NOT AVAILABLE IN HASH TABLE
1. INSERTION            2.DELETION
3.SEARCHING             4.DISPLAY
5.EXIT

ENTER YOUR CHOICE:2 ENTER
THE KEY VALUE:34
GIVEN DATA DELETED FROM HASH TABLE
        1. INSERTION    2.DELETION
        3.SEARCHING     4.DISPLAY
        5.EXIT
```

**VIVA Questions**:

1. What are the advantages of hashing?
2. What is a hash table?
3. Explain Collision.
4. How do we handle Collision in Hash table?
5. What is Linear Probing?

# SAFETY INSTRUCTIONS

## Do's

1. Do wear ID card and follow dress code.
2. Be on time to Lab sessions.
3. Do log off the computers when you finish.
4. Do ask the staff for assistance if you need help.
5. Do keep your voice low when speaking to others in the LAB.
6. Do ask for assistance in downloading any software.
7. Do make suggestions as to how we can improve the LAB.
8. In case of any hardware related problem, ask LAB in charge for solution.
9. If you are the last one leaving the LAB , make sure that the staff in charge of the LAB is informed to close the LAB.
10. Be on time to LAB sessions.
11. Do keep the LAB as clean as possible.

## Don'ts

1. Do not use mobile phone inside the lab.
2. Don't do anything that can make the LAB dirty (like eating, throwing waste papers etc).
3. Do not carry any external devices without permission.
4. Don't move the chairs of the LAB.
5. Don't interchange any part of one computer with another.
6. Don't leave the computers of the LAB turned on while leaving the LAB.
7. Do not install or download any software or modify or delete any system files on any lab computers.
8. Do not damage, remove, or disconnect any labels, parts, cables, or equipment.
9. Don't attempt to bypass the computer security system.
10. Do not read or modify other user's file.
11. If you leave the lab, do not leave your personal belongings unattended. We are not responsible for any theft.