

Visvesvaraya Technological University, Belagavi
POST GRADUATION CENTRE, KALABURAGI
BACHELOR OF TECHNOLOGY (B.Tech)
IN
COMPUTER SCIENCE & ENGINEERING



A
Laboratory Report
ON

DATA STRUCTURES LABORATORY
Department of Computer Science & Engineering, Post
graduation Centre, Visvesvaraya Technological University,
KALABURAGI

2023-2024

SL	Name of Program	Page No
1	<p>Develop a Program in C for the following:</p> <p>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).</p> <p>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.</p>	
2	<p>Develop a Program in C for the following operations on Strings.</p> <p>a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)</p> <p>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.</p>	
3	<p>Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)</p> <p>a. Push an Element on to Stack</p> <p>b. Pop an Element from Stack</p> <p>c. Demonstrate how Stack can be used to check Palindrome</p> <p>d. Demonstrate Overflow and Underflow situations on Stack</p> <p>e. Display the status of Stack</p> <p>f. Exit</p> <p>Support the program with appropriate functions for each of the above operations</p>	
4	<p>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.</p>	
5	<p>Develop a Program in C for the following Stack Applications</p> <p>a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^.</p> <p>b. Solving Tower of Hanoi problem with n disks</p>	

6	<p>Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)</p> <ol style="list-style-type: none"> Insert an Element on to Circular QUEUE Delete an Element from Circular QUEUE Demonstrate Overflow and Underflow situations on Circular QUEUE Display the status of Circular QUEUE Exit <p>Support the program with appropriate functions for each of the above operations</p>	
7	<p>Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: <i>USN, Name, Programme, Sem, PhNo</i></p> <ol style="list-style-type: none"> Create a SLL of N Students Data by using <i>front insertion</i>. Display the status of SLL and count the number of nodes in it Perform Insertion / Deletion at End of SLL Perform Insertion / Deletion at Front of SLL(Demonstration of stack) Exit 	
8	<p>Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: <i>SSN, Name, Dept, Designation, Sal, PhNo</i></p> <ol style="list-style-type: none"> Create a DLL of N Employees Data by using <i>end insertion</i>. Display the status of DLL and count the number of nodes in it Perform Insertion and Deletion at End of DLL Perform Insertion and Deletion at Front of DLL Demonstrate how this DLL can be used as Double Ended Queue. Exit 	
9	<p>Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes</p> <ol style="list-style-type: none"> Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3$ Find the sum of two polynomials $POLY1(x,y,z)$ and $POLY2(x,y,z)$ and store the result in $POLYSUM(x,y,z)$ <p>Support the program with appropriate functions for each of the above operations</p>	
10	<p>Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .</p>	

	<p>a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2</p> <p>b. Traverse the BST in Inorder, Preorder and Post Order</p> <p>c. Search the BST for a given element (KEY) and report the appropriate message d. Exit</p>	
11	<p>Develop a Program in C for the following operations on Graph(G) of Cities</p> <p>a. Create a Graph of N cities using Adjacency Matrix.</p> <p>b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method</p>	
12	<p>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 $H: K \rightarrow L$ as $H(K)=K \bmod m$ (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.</p>	

PROGRAMS

1. Develop a Program in C for the following:

a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (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>
struct calElement
{
    char *day;
    int date;
    char *activity;
};
struct calElement* create() // it creates calendar structure for 7 days
{
    struct calElement *calendar;
    // dynamic allocation for calendar
    calendar = (struct calElement *)malloc(sizeof(struct
    calElement)*7); return calendar;
}
void read(struct calElement *calendar)
{
    // Local Variable to store string
    elements char day[10];
    char activity[25];
    int i, date;
    for(i = 0; i<7; i++)
    {
        printf("Enter the day :
        "); scanf("%s",day);
```

```

        calendar[i].day = (char *)malloc(strlen(day)+1); // dynamically allocated memory for
        day strcpy(calendar[i].day, day); // copy day from local variable to heap printf("Enter the
        date : ");
        scanf("%d",&date);
        calendar[i].date = date;
        getchar();
        printf("Enter description of the activity : ");
        scanf("%[^\n]s",activity);
        // dynamically allocate memory for activity calendar[i].activity = (char
        *)malloc(strlen(activity)+1); strcpy(calendar[i].activity, activity); // copy
        activity from local variable to heap
    }
}
void display(struct calElement *calendar)
{
    int i;
    printf("\n\nYour calendar\n");
    printf("Day\t\tDate\t\tActivity");
    //Display the calendar
    for(i = 0; i<7; i++){
        printf("\n%s\t\t%d\t\t%s",calendar[i].day,calendar[i].date,calendar[i].activity );
    }
}
void main()
{
    struct calElement *calendar; // create structure variable of type
    pointer calendar=create(); //call create function
    read(calendar); // read function to read all inputs
    display(calendar); // Function to print calendar
    free(calendar); // Release the memory allocated dynamically
}

```

OUTPUT :

```

Enter the day : Mon
Enter the date : 10
Enter description of the activity : DS Class
Enter the day : Tue
Enter the date : 11
Enter description of the activity : Visit Library

```

Enter the day : Wed
Enter the date : 12
Enter description of the activity : OS Class
Enter the day : Thu
Enter the date : 13
Enter description of the activity : Research
Enter the day : Fri
Enter the date : 14
Enter description of the activity : Tutorial
Enter the day : Sat
Enter the date : 15
Enter description of the activity : Presentation on DD
Enter the day : Sun
Enter the date : 16
Enter description of the activity : Go to Movie

Your calender

Day	Date	Activity
Mon	10	DS Class
Tue	11	Visit Library
Wed	12	OS Class
Thu	13	Research
Fri	14	Tutorial
Sat	15	Presentation on DD
Sun	16	Go to Movie

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<stdio.h>
char STR[100],PAT[100],REP[100],ANS[100];
int s, p, r, a, flag = 0;
//Function Declaration
void read();
void replace();
void display();
void main()
{
    read();
```

```

        replace();
        void display();
    }
    void read()
    {
        printf("Enter the MAIN string: \n");
        scanf("%[^\\n]s",STR);
        printf("Enter a PATTERN string: \n");
        scanf("%s",PAT);
        printf("Enter a REPLACE string: \n");
        scanf("%s",REP);
    }
    void replace ()
    {
        s = p = a = 0;
        while ( STR[s] != '\0')
        {
            // Checking for Match
            if ( STR[s] == PAT[p] )
            {
                p++;
                s++;
                //if the first character is matched check for entire PAT string
                if ( PAT[p] == '\0')
                {
                    flag=1;
                    for(r = 0; REP[r] != '\0';r++, a++)
                        ANS[a] = REP[r];
                    p = 0;
                }
            }
            else //Mismatch
            {
                ANS[a] = STR[s];
                s++;
                a++;
                p = 0;
            }
        }
    }
    void print()
    {

```



```

    if(flag==0)
        printf("Pattern doesn't found!!!");
    else
    {
        ANS[a] = '\0';
        printf("\nThe RESULTANT string is:\n%s\n" ,ANS);
    }
}

```

OUTPUT 1:

1. Enter the MAIN string:
 good morning
 Enter a PATTERN string:
 morning
 Enter a REPLACE string:
 day
 The RESULTANT string is:
 good day

OUTPUT 2:

2. Enter the MAIN string:
 ACHARYA
 Enter a PATTERN string:
 COLLEGE
 Enter a REPLACE string:
 INSTITUTE
 Pattern doesn't found!!!

3. Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

- a. Push an Element on to Stack
- b. Pop an Element from Stack
- c. Demonstrate how Stack can be used to check Palindrome
- d. Demonstrate Overflow and Underflow situations on Stack
- e. Display the status of Stack
- f. Exit

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

```

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

```

```

#include<math.h>
#define MAX 5
int s[MAX],top = -1,item;
int IsFull()
{
    if(top>=MAX-1)
        return 1;
    return 0;
}
int IsEmpty()
{
    if(top== -1)
        return 1;
    return 0;
}
void push(int item)
{
    top++;
    s[top]=item;
}
void pop()
{
    item=s[top];
    top--;
}
void display()
{
    int i;
    printf("\n the elements of the stack are");
    for(i=top;i>=0;i--)
        printf("\n %d",s[i]);
}

void check_pal()
{
    int num=0,temp,digit,revnum=0,k=0;
    if(top== -1)
    {
        printf("Stack is empty\n");
        return;
    }
    else
    {
        while(top!=-1)
        {
            pop();
            num=num*10+item;
            revnum=item*pow(10,k)+revnum;

```

```

        k=k+1;
    }
    printf("\nReverse Number of %d is is: %d\n",num, revnum);
    if(num == revnum)
        printf("The stack contains a Palindrome number\n");
    else
        printf("The stack does not contain a Palindrome number\n");
    }
}
void main()
{
    int ch;

    do
    {
        printf("\n1. Push \n2. Pop \n3. Display\n4. Check Palindrome 5. Exit\n");
        printf("\n Enter the choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\n Enter the element to be
                        inserted"); scanf("%d",&item);
                    if(IsFull())
                        printf("Stack Overflow\n");
                    else
                        push(item);
                    break;
            case 2: if(IsEmpty())
                    printf("Stack Underflow\n");
                    else
                    {
                        pop();
                        printf("The item deleted is %d\n",item);
                    }
                    break;
            case 3: if(IsEmpty())
                    printf("Stack is Empty\n");
                    else
                        display();
                    break;
            case 4: check_pal();
                    break;
            case 5: printf("Program Terminated \n");
                    exit(0);
            default:printf("Invalid choice: \n");
        }
    }while(ch!=5);
}

```

OUTPUT:

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 1

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 9

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 9

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 1

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 3

the elements of the stack are

1

9

9

1

1. Push

2. Pop

3. Display

4. Check Palindrome 5. Exit

Enter the choice: 4

Reverse Number of 1991 is is: 1991

The stack contains a Palindrome number

1. Push

2. Pop

3. Display

4. Check Palindrome 5. Exit

Enter the choice: 2

Stack Underflow

1. Push

2. Pop

3. Display

4. Check Palindrome 5. Exit

1. Push

2. Pop

3. Display

4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 10

10

1. Push

2. Pop

3. Display

4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 20

20

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 30

30

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 40

40

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 50

50

1. Push
2. Pop
3. Display
4. Check Palindrome 5. Exit

Enter the choice: 1

Enter the element to be inserted 60

60

Stack Overflow

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

```
#include <stdio.h>
#define SIZE 20
char s[SIZE];
int top = -1;
void push(char elem)
{
    s[++top] = elem;
}
char pop()
{
    return (s[top--]);
}
int precedence(char elem) /* Function for precedence */
{
    switch (elem)
    {
        case '#': return 0;
        case '(': return 1;
        case '+':
        case '-': return 2;
        case '*':
        case '/':
        case '%': return 3;
        case '$':
        case '^': return 4;
    }
    return elem;
}
void main()
{
    char infix[50], postfix[50], ch, elem;
    int i = 0, k = 0;
    printf("\n\nEnter the Infix Expression: ");
    scanf("%s", infix);
    push('#');
    while ((ch = infix[i++]) != '\0')
    {
        if (ch == '(')
            push(ch);
```

```

else if (isdigit(ch))
    postfix[k++] = ch;
else if (ch == ')')
{
    while (s[top] != '(')
        postfix[k++] = pop();
    elem = pop(); /* Remove ( */
}
else /* Operator */
{
    while (precedence(s[top]) >= precedence(ch))
        postfix[k++] = pop();
    push(ch);
}
}
while (s[top] != '#'){ /* Pop from stack till empty */
    postfix[k++] = pop();
}
postfix[k] = '\0'; /* Make postfix as valid string */
printf("\nGiven Infix Expression is: %s\n",infix);
printf("\nPostfix Expression is: %s\n", postfix);
}

```

OUTPUT 1:

Enter the Infix Expression: A+B*C+D

Given Infix Expression is: A+B*C+D

Postfix Expression is: ABC*+D+

OUTPUT 2:

Enter the Infix Expression: ((4+8)(6-5))/((3-2)(2+2))

Given Infix Expression is: ((4+8)(6-5))/((3-2)(2+2))

Postfix Expression is: ((4+8)(6-5))/((3-2)(2+2))

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

5a.

```

#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
int compute(char symbol, int op1, int op2)
{

```



```

switch(symbol)
{
    case '+': return op1+op2; /* Perform addition */
    case '-': return op1-op2; /* Perform subtraction */
    case '*': return op1*op2; /* Perform multiplaction */
    case '/': return op1/op2; /* Perform division */
    case '%': return op1%op2; /* Perform division and gives reminder
    */ case '$':
    case '^': return pow(op1,op2); /* Compute power */
}
}
void main()
{
    int s[20]; /* Place for stack elements */
    int res; /* Holds partial or final result */
    int op1; /* First operand */
    int op2; /* Second operand */
    int top;
    /* Points to the topmost element */
    int i;
    /* Index value */
    char postfix[20]; /* Input expression */
    char symbol; /* Scanned postfix symbol */
    printf("Enter the postfix expression\n");
    scanf("%s",postfix);
    top=-1;
    for(i=0;i<strlen(postfix);i++)
    {
        symbol=postfix[i]; /* Obtains the next character */
        if(isdigit(symbol)) /* If character is a digit or not */
            s[++top]=symbol-'0';
        else
        {
            op2=s[top--];
            /* Obtain second operand from stack */
            op1=s[top--];
            /* Obtain first operand from stack */
            /* Perform specified operation */
            res=compute(symbol,op1,op2);
            /* Push partial results on the stack */
            s[++top]=res;
        }
    }
    res=s[top--];
    printf("the result is %d\n",res);
}

```

OUTPUT 1:

Enter the postfix expression

23^

the result is 8

OUTPUT 2:

Enter the postfix expression

234+*6-

the result is 8

5b.

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

```
void towers(int, char, char, char);
```

```
void main()
```

```
{
```

```
    int num;
```

```
    printf("Enter the number of disks : ");
```

```
    scanf("%d", &num);
```

```
    printf("The sequence of moves involved in the Tower of Hanoi are :\n");
```

```
    towers(num, 'A', 'B', 'C');
```

```
}
```

```
void towers(int n, char source, char temp, char dest)
```

```
{
```

```
    if (n == 1)
```

```
    {
```

```
        printf("\n Move disk 1 from peg %c to peg %c", source, dest);
```

```
        return;
```

```
    }
```

```
    towers(n - 1, source, dest, temp);
```

```
    printf("\n Move disk %d from peg %c to peg %c", n, source, dest);
```

```
    towers(n - 1, temp, source, dest);
```

```
}
```

OUTPUT 1:

Enter the number of disks : 2

The sequence of moves involved in the Tower of Hanoi are :

Move disk 1 from peg A to peg B

Move disk 2 from peg A to peg C

Move disk 1 from peg B to peg C

OUTPUT 2:

Enter the number of disks : 1

The sequence of moves involved in the Tower of Hanoi are :

Move disk 1 from peg A to peg C

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

```
#include<stdio.h>
#include<stdlib.h>
#define QSIZE 4
int q[QSIZE], r=-1, f=0, count=0, item;
/* Insert Operation */
void insert()
{
    /* Check for queue overflow */
    if(count == QSIZE)
    {
        printf("Queue Overflow\n");
        return;
    }
    r = (r+1) % QSIZE; /* Increment rear by 1 */
    q[r] = item; /* Insert into queue */
    count++;
}
/* Delete Operation */
void del()
{
    /* Check for Queue Underflow */
    if(count == 0)
    {
        printf("Queue Underflow\n");
        return;
    }
    printf("The item deleted is: %d\n", q[f]);
    f = (f+1) % QSIZE;
    count--;
}
/* Display Operation */
void display(int front)
{
    int i;
    /* Check for Empty Queue */
    if(count == 0)
    {
```

```

        printf("Queue is Empty\n");
        return;
    }
    /* Display the contents of the queue */
    printf("Contents of the queue\n");
    for(i=1; i<=count; i++)
    {
        printf("%d\n",q[front]);
        front = (front+1) % QSIZE;
    }
}
void main()
{
    int choice;
    do
    {
        printf("*****\n");
        printf("Circular Queue Operations\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Quit\n");
        printf("Enter your choice:\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the item to be inserted\n");
                    scanf("%d",&item);
                    insert();
                    break;
            case 2: del();
                    break;
            case 3: display(f);
                    break;
            case 4: exit(0);
            default:printf("Invalid choice\n");
        }
    }while(choice!=4);
}

```

OUTPUT

Circular Queue Operations

1. Insert

2. Delete

3. Display

4. Quit

Enter your choice:

1
Enter the item to be inserted

10

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

1

Enter the item to be inserted

20

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

1

Enter the item to be inserted

30

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

1

Enter the item to be inserted

40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

3

Contents of the queue

10

20

30

40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

1

Enter the item to be inserted

50

Queue Overflow

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

3

Contents of the queue

10

20

30

40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

The item deleted is: 10

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

1

Enter the item to be inserted

60

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

3

Contents of the queue

60

20

30

40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

The item deleted is: 20

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

3

Contents of the queue

60

30

40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

The item deleted is: 30

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

The item deleted is: 40

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

The item deleted is: 60

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

2

Queue Underflow

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

3

Queue is Empty

Circular Queue Operations

1. Insert
2. Delete
3. Display
4. Quit

Enter your choice:

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>
#include<string.h>
int count=0;
struct node
{
    int sem;
    long int phno;
    char name[20], branch[10], usn[20];
    struct node *next;
}*first=NULL,*last=NULL,*temp=NULL, *temp1;
void create()
{
    int sem;
    long int phno;
    char name[20],branch[10],usn[20];
    temp=(struct node*)malloc(sizeof(struct node));
    printf("\n Enter USN, NAME, BRANCH, SEMESTER, PHNUM of student :
"); scanf("%s %s %s %d %ld", usn, name,branch, &sem,&phno); strcpy(temp-
>usn,usn);
    strcpy(temp->name,name);
    strcpy(temp->branch,branch);
    temp->sem = sem;
    temp->phno = phno;
    temp->next=NULL;
    count++;
}
void insert_atfirst()
{
    if (first == NULL)
    {
        create();
        first = temp;
        last = first;
    }
    else
    {
        create();
        temp->next = first;
        first = temp;
    }
}
```

```

    }
}
void insert_atlast()
{
    if(first==NULL)
    {
        create();
        first = temp;
        last = first;
    }
    else
    {
        create();
        last->next = temp;
        last = temp;
    }
}
void display()
{
    temp1=first;
    if(temp1 == NULL)
    {
        printf("List empty to display \n");
        return;
    }
    printf("\n Linked list elements from begining : \n");
    printf("USN\t NAME\t BRANCH\t SEMESTER\t PH.NUM\n");
    while (temp1 != NULL)
    {
        printf("%s\t %s\t %s\t %d\t %ld\n", temp1->usn, temp1->name, temp1->branch, temp1->sem, temp1->phno);
        temp1 = temp1->next;
    }
    printf(" No of students = %d ", count);
}
void delete_end()
{
    struct node *temp;
    temp=first;
    if(first==NULL)
    /* List is Empty */
    {
        printf("List is Empty\n");
        return;
    }
    if(temp->next==NULL)
    /* If only there is one node in the List */
    {

```

```

        printf("%s %s %s %d %ld\n", temp->usn, temp->name,temp->branch, temp->sem, temp->phno );
        free(temp);
        first=NULL;
    }
    else
    /* If more than one node in the List */
    {
        while(temp->next!=last)
            temp=temp->next;
        printf("%s %s %s %d %ld\n", last->usn, last->name,last->branch, last->sem, last->phno ); free(last);
        temp->next=NULL;
        last=temp;
    }
    count--;
}
void delete_front()
{
    struct node *temp;
    temp=first;
    if(first==NULL)
    /* List is Empty */
    {
        printf("List is Empty\n");
        return;
    }
    if(temp->next==NULL) /* If only there is one node in the List */
    {
        printf("%s %s %s %d %ld\n", temp->usn, temp->name,temp->branch, temp->sem, temp->phno );
        free(temp);
        first=NULL;
    }
    else
    /* If more than one node in the List */
    {
        first=temp->next;
        printf("%s %s %s %d %ld", temp->usn, temp->name,temp->branch,temp->sem, temp->phno );
        free(temp);
    }
    count--;
}
void main()
{
    int ch,n,i;
    first=NULL;

```

```

temp = temp1 = NULL;
printf(".....MENU.....\n");
printf("\n 1 Create a SLL of n Employees");
printf("\n 2 - Display from beginning");
printf("\n 3 - Insert at end");
printf("\n 4 - delete at end");
printf("\n 5 - Insert at beg");
printf("\n 6 - delete at beg");
printf("\n 7 - exit\n");
printf(".....\n");
while (1)
{
    printf("\n Enter choice : ");
    scanf("%d", &ch);
    switch (ch)
    {
        case 1: printf("\n Enter no of students : ");
                scanf("%d", &n);
                for(i=0;i<n;i++)
                    insert_atfirst();
                break;
        case 2: display();
                break;
        case 3: insert_atlast();
                break;
        case 4: delete_end();
                break;
        case 5: insert_atfirst();
                break;
        case 6: delete_front();
                break;
        case 7: exit(0);
        default: printf("Wrong Choice\n");
    }
}
:

```

OUTPUT.....MENU.....

```

1 Create a SLL of n Employees
2 - Display from beginning
3 - Insert at end
4 - delete at end
5 - Insert at beg
6 - delete at beg
7 - exit

```

Enter choice : 1

Enter no of students : 1

Enter USN, NAME, BRANCH, SEMESTER, PHNUM of student : 01 anu cs 3 123456789

Enter choice : 2

Linked list elements from begining :

USN	NAME	BRANCH	SEMESTER	PH.NUM
01	anu	cs	3	123456789

No of students = 1

Enter choice : 3

Enter USN, NAME, BRANCH, SEMESTER, PHNUM of student : 02 bhayana cse 3 123456789

Enter choice : 2

Linked list elements from begining :

USN	NAME	BRANCH	SEMESTER	PH.NUM
01	anu	cs	3	123456789
02	bhavana	cse	3	123456789

No of students = 2

Enter choice : 5

Enter USN, NAME, BRANCH, SEMESTER, PHNUM of student : 03 chandana cs 3 123456789

Enter choice : 2

Linked list elements from begining :

USN	NAME	BRANCH	SEMESTER	PH.NUM
03	chandana	cs	3	123456789
01	Anu	cs	3	123456789
02	bhavana	cse	3	123456789

No of students = 3

Enter choice : 4

02 bhavana cse 3 123456789

Enter choice : 2

Linked list elements from begining :

USN	NAME	BRANCH	SEMESTER	PH.NUM
03	chandana	cs	3	123456789
01	anu	cs	3	123456789

No of students = 2

Enter choice : 6

03 chandana cs 3 123456789

Linked list elements from beginning :

USN	NAME	BRANCH	SEMESTER	PH.NUM
-----	------	--------	----------	--------

01 anu cs 3

No of students = 1 12345678

Enter choice : 7 9

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

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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct node
{
    struct node *prev;
    int ssn;
    long int phno;
    float sal;
    char name[20],dept[10],desg[20];
    struct node *next;
}*first,*temp,*last;
void create()
{
    int ssn;
    long int phno;
    float sal;
    char name[20],dept[10],desg[20];
    temp =(struct node *)malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter ssn,name,department, designation, salary and phno of employee :
"); scanf("%d %s %s %s %f %ld", &ssn, name,dept,desg,&sal, &phno);
    temp->ssn = ssn;
    strcpy(temp->name,name);
    strcpy(temp->dept,dept);
    strcpy(temp->desg,desg);
    temp->sal = sal;
    temp->phno = phno;
    count++;
}
```

```

void display()
{
    temp = first;
    if(temp == NULL)
    {
        printf("List is Empty\n");
        return;
    }
    printf("\n Linked list elements from begining : \n");
    while (temp != NULL)
    {
        printf("%d %s %s %s %f %ld\n", temp->:ssn, temp->name,temp->dept,temp-
        >desg,temp->sal, temp->phno );
        temp = temp->next;
    }
    printf(" No of employees = %d", count);
}
void insert_front()
{
    if (first == NULL)
    {
        create();
        first = temp;
        last = first;
    }
    else
    {
        create();
        temp->next = first;
        first->prev = temp;
        first = temp;
    }
}
void delete_front()
{
    struct node *cur=first;
    if(first == NULL) /* If the List is Empty */
    {
        printf("List is Empty\n");
        return;
    }
    if(first->next == NULL) /*If there is only one node in the List */
    {
        printf("%d %s %s %s %f %ld\n", first->:ssn, first->name,first->dept, first->desg,first-
        >sal,first->phno );
        free(first);
        first = NULL;
    }
}

```

```

        else
        {
            first = first->next;
            printf("%d %s %s %s %f %ld", cur->ssn, cur->name, cur->dept, cur->desg, cur->sal, cur->phno );
            free(cur);
        }
        count--;
    }
void insert_rear()
{
    if(first == NULL)
    {
        create();
        first = temp;
        last = first;
    }
    else
    {
        create();
        last->next = temp;
        temp->prev = last;
        last = temp;
    }
}
void delete_rear()
{
    if(first == NULL) /*If the list is Empty */
    {
        printf("List is Empty\n");
        return;
    }
    if(first->next == NULL) /*If there is only one node in the List */
    {
        printf("%d %s %s %s %f %ld\n", first->ssn, first->name, first->dept, first->desg, first->sal, first->phno );
        free(first);
        first = NULL;
    }
    else
    {
        temp = last->prev;
        temp->next = NULL;
        printf("%d %s %s %s %f %ld\n", last->ssn, last->name, last->dept, last->desg, last->sal, last->phno );
        free(last);
        last=temp;
    }
}

```



```

        count--;
    }
void main()
{
    int ch,n,i;
    first = NULL;
    temp = last = NULL;
    printf(".....MENU.....\n");
    printf("\n 1 - Create a DLL of n emp");
    printf("\n 2 - Display from beginning");
    printf("\n 3 - Insert at front end");
    printf("\n 4 - Delete at front end");
    printf("\n 5 - Insert at rear end");
    printf("\n 6 - Delete at rear end");
    printf("\n 7 - exit\n");
    printf(".....\n");
    while (1)
    {
        printf("\n Enter Choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1: printf("\n Enter no of employees : ");
                    scanf("%d", &n);
                    for(i=0;i<n;i++)
                        insert_rear();
                    break;
            case 2: display();
                    break;
            case 3: insert_front();
                    break;
            case 4: delete_front();
                    break;
            case 5: insert_rear();
                    break;
            case 6: delete_rear();
                    break;
            case 7: exit(0);
            default: printf("Wrong Choice\n");
        }
    }
}

```

OUTPUT:

```

.....
M
.....
1 - Create a DLL of n emp
2 - Display from beginning
E

```

- 3 - Insert at front end
- 4 - Delete at front end
- 5 - Insert at rear end
- 6 - Delete at rear end
- 7 - exit

Enter Choice : 1

Enter no of employees : 2

Enter ssn,name,department, designation, salary and phno of employee : 01 anusha cse hr
15000 1234567890

Enter ssn,name,department, designation, salary and phno of employee : 02 bhavya cse manager 20000
1234567890

Enter Choice : 2

Linked list elements from begining :

1 sha cse hr 15000.000000 1234567890

2 bhavya cse manager 20000.000000 1234567890

No of employees = 2

Enter Choice : 3

Enter ssn,name,department, designation, salary and phno of employee : 03 maha sales assistant
20000 1234567890

Enter Choice : 2

Linked list elements from begining :

3 maha sales assistant 20000.000000 1234567890

1 sha cse hr 15000.000000 1234567890

2 bhavya cse manager 20000.000000 1234567890

No of employees = 3

Enter Choice : 5

Enter ssn,name,department, designation, salary and phno of employee : 04 shruthi research professor
30000 1234567890

Enter Choice : 2

Linked list elements from begining :

3 maha sales assistant 20000.000000 1234567890

1 sha cse hr 15000.000000 1234567890

2 bhavya cse manager 20000.000000 1234567890

4 shruthi research professor 30000.000000 1234567890

No of employees = 4

Enter Choice : 4

3 maha sales assistant 20000.000000 1234567890

Enter Choice : 2

Linked list elements from begining :

1 sh cse hr 15000.000000 1234567890

2 bhavya cse manager 20000.000000 1234567890

4 shruthi research professor 30000.000000 1234567890

No of employees = 3

Enter Choice : 6

4 shruthi research professor 30000.000000 1234567890

Enter Choice : 2

Linked list elements from begining :

1 sha cse hr 15000.000000 1234567890

2 bhavya cse manager 20000.000000 1234567890

No of employees = 2

Enter Choice : 7

9. Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2 y^2 z - 4yz^5 + 3x^3 yz + 2xy^5 z - 2xyz^3$ 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<math.h>
#include<stdlib.h>
struct node
{
    int cf, px, py, pz;
    int flag;
    struct node *link;
};
typedef struct node NODE;
NODE* getnode()
{
    NODE *x;
    x = (NODE*)malloc(sizeof(NODE));
    if(x == NULL)
    {
        printf("Insufficient memory\n");
        exit(0);
    }
}
```

```

        return x;
    }
void display(NODE *head)
{
    NODE *temp;
    if(head->link == head)
    {
        printf("Polynomial does not exist\n");
        return;
    }
    temp = head->link;
    printf("\n");
    while(temp != head)
    {
        printf("%d x^%d y^%d z^%d",temp->cf,temp->px,temp->py,temp->pz);
        if(temp->link != head)
            printf(" + ");
        temp = temp->link;
    }
    printf("\n");
}
NODE* insert_rear(int cf,int x,int y,int z,NODE *head)
{
    NODE *temp, *cur;
    temp = getnode();
    temp->cf = cf;
    temp->px = x;
    temp->py = y;
    temp->pz = z;
    cur = head->link;
    while(cur->link != head)
        cur = cur->link;
    cur->link = temp;
    temp->link = head;
    return head;
}
NODE* read_poly(NODE *head)
{
    int px, py, pz, cf, ch;
    printf("\nEnter coeff: ");
    scanf("%d",&cf);
    printf("\nEnter x, y, z powers(0-indicate NO term): ");
    scanf("%d%d%d", &px, &py, &pz);
    head = insert_rear(cf,px,py,pz,head);
    printf("\nIf you wish to continue press 1 otherwise 0: ");
    scanf("%d", &ch);
    while(ch != 0)
    {

```

```

        printf("\nEnter coeff: ");
        scanf("%d",&cf);
        printf("\nEnter x, y, z powers(0-indiacate NO term):
        "); scanf("%d%d%d",&px, &py, &pz);
        head = insert_rear(cf,px,py,pz,head);
        printf("\nIf you wish to continue press 1 otherwise 0:
        "); scanf("%d", &ch);
    }
    return head;
}
NODE* add_poly(NODE *h1,NODE *h2,NODE *h3)
{
    NODE *p1, *p2;
    int cf;
    p1 = h1->link;
    while(p1 != h1)
    {
        p2=h2->link;
        while(p2 != h2)
        {
            if(p1->px == p2->px && p1->py == p2->py && p1->pz == p2->pz)
                break;
            p2 = p2->link;
        }
        if(p2 != h2)
        {
            cf = p1->cf + p2->cf;
            p2->flag = 1;
            if(cf != 0)
                h3 = insert_rear(cf,p1->px,p1->py,p1->pz,h3);
        }
        else
            h3 = insert_rear(p1->cf,p1->px,p1->py,p1->pz,h3);
        p1 = p1->link;
    }
    p2 = h2->link;
    while(p2 != h2)
    {
        if(p2->flag == 0)
            h3 = insert_rear(p2->cf,p2->px,p2->py,p2->pz,h3);
        p2 = p2->link;
    }
    return h3;
}
void evaluate(NODE *head)
{
    NODE *h1=head->link;
    int x, y, z;

```

```

float result = 0.0;
printf("\nEnter x, y, z, terms to evaluate:\n");
scanf("%d%d%d", &x, &y, &z);
while(h1 != head)
{
    result = result + (h1->cf * pow(x,h1->px) * pow(y,h1->py) * pow(z,h1->pz)); h1 = h1->link;
}
printf("\nPolynomial result is: %f", result);
}
void main()
{
    NODE *h1, *h2, *h3, *eval;
    int ch;
    while(1)
    {
        eval = getnode();
        h1 = getnode();
        h2 = getnode();
        h3 = getnode();
        eval->link = eval;
        h1->link = h1;
        h2->link = h2;
        h3->link = h3;
        printf("\nn1.Evaluate polynomial\n2.Add two polynomials\n3.Exit\n"); printf("Enter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: printf("\nEnter polynomial to evaluate:\n");
                    eval = read_poly(eval);
                    display(eval);
                    evaluate(eval);
                    free(eval);
                    break;
            case 2: printf("\nEnter the first polynomial: ");
                    h1 = read_poly(h1);
                    printf("Flag = %d\n",h1->flag);
                    printf("\nEnter the second polynomial: ");
                    h2 = read_poly(h2);
                    h3 = add_poly(h1,h2,h3);
                    printf("\nFirst polynomial is: ");
                    display(h1);
                    printf("\nSecond polynomial is: ");
                    display(h2);
                    printf("\nThe sum of 2 polynomials is: ");
                    display(h3);
                    free(h1);

```

```

        free(h2);
        free(h3);
        break;
    case 3: exit(0);
        break;
    default: printf("\nInvalid entry");
    }
}
}

```

OUTPUT:

- 1.Evaluate polynomial
- 2.Add two polynomials
- 3.Exit

Enter your choice: 1

Enter polynomial to evaluate:

Enter coeff: 2

Enter x, y, z powers(0-indiacate NO term): 1 1 1

If you wish to continue press 1 otherwise 0: 0

2 x^1 y^1 z^1

Enter x, y, z, terms to evaluate:

1 1 1

Polynomial result is: 2.000000

- 1.Evaluate polynomial
- 2.Add two polynomials
- 3.Exit

Enter your choice: 2

Enter the first polynomial:

Enter coeff: 2

Enter x, y, z powers(0-indiacate NO term): 1 1 1

If you wish to continue press 1 otherwise 0: 0

Flag = 0

Enter the second polynomial:

Enter coeff: 4

Enter x, y, z powers(0-indiacate NO term): 1 1 1

If you wish to continue press 1 otherwise 0: 0

First polynomial is:

2 x^1 y^1 z^1

Second polynomial is:

4 x^1 y^1 z^1

The sum of 2 polynomials is:

$6x^1y^1z^1$

1.Evaluate polynomial
2.Add two polynomials
3.Exit
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 *left;
    struct BST *right;
};

typedef struct BST NODE;
NODE* createtree(NODE *root, int data)
{
    if (root == NULL)
    {
        NODE *temp;
        temp = (NODE*) malloc (sizeof(NODE));
        temp->data = data;
        temp->left = temp->right = NULL;
        return temp;
    }
    if (data < (root->data))
        root->left = createtree(root->left, data);
    else if (data > root->data)
        root->right = createtree(root->right, data);
    return root;
}

void inorder(NODE *root)
{
    if(root != NULL)
    {
        inorder(root->left);
        printf("%d\t", root->data);
        inorder(root->right);
    }
}
```



```

}
void preorder(NODE *root)
{
    if(root != NULL)
    {
        printf("%d\t", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void postorder(NODE *root)
{
    if(root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d\t", root->data);
    }
}
NODE *search(NODE *root, int data)
{
    if(root == NULL)
        printf("\nElement not found");
    else if(data < root->data)
        root->left = search(root->left, data);
    else if(data > root->data)
        root->right = search(root->right, data);
    else
        printf("\nElement found is: %d", root->data);
    return root;
}
void main()
{
    int data, ch, i, n;
    NODE *root = NULL;
    while (1)
    {
        printf("\n1.Creation of Binary Search Tree");
        printf("\n2.Inorder\n3.Preorder\n4.Postorder\n5.Search\n6.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1: printf("\nEnter N value: ");
                    scanf("%d", &n);
                    printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");
                    for(i=0; i<n; i++)
                    {

```

```

        scanf("%d", &data);
        root = createtree(root, data);
    }
    break;
case 2: printf("\nInorder Traversal: \n");
        inorder(root);
        break;
case 3: printf("\nPreorder Traversal: \n");
        preorder(root);
        break;
case 4: printf("\nPostorder Traversal: \n");
        postorder(root);
        break;
case 5: printf("\nEnter the element to Search: ");
        scanf("%d", &data);
        root=search(root, data);
        break;
case 6: exit(0);
default: printf("\nWrong Option");
        break;
    }
}
}

```

OUTPUT:

```

1.Creation of Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Search
6.Exit
Enter your choice: 1
Enter N value: 12
Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)
695281524147852
1.Creation of Binary Search Tree
2.Inorder
3.Preorder
4.Postorder
5.Search
6.Exit
Enter your choice: 2
Inorder Traversal:
2    5    6    7    8    9    14    15    24
1.Creation of Binary Search Tree
2.Inorder
3.Preorder

```

4.Postorder
 5.Search
 6.Exit
 Enter your choice: 3
 Preorder Traversal:
 6 5 2 9 8 7 15 14 24

1.Creation of Binary Search Tree
 2.Inorder
 3.Preorder
 4.Postorder
 5.Search
 6.Exit

Enter your choice: 4
 Postorder Traversal:
 2 5 7 8 14 24 15 9 6

1.Creation of Binary Search Tree
 2.Inorder
 3.Preorder
 4.Postorder
 5.Search
 6.Exit

Enter your choice: 5
 Enter the element to Search: 8
 Element found is: 8
 1.Creation of Binary Search Tree
 2.Inorder
 3.Preorder
 4.Postorder
 5.Search
 6.Exit

Enter your choice: 5
 Enter the element to Search: 10
 Element not found
 1.Creation of Binary Search Tree
 2.Inorder
 3.Preorder
 4.Postorder
 5.Search
 6.Exit

Enter your choice: 6

11. Develop a Program in C for the following operations on Graph(G) of Cities a. Create a Graph of N cities using Adjacency Matrix. b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method

```
#include<stdio.h>
#include<stdlib.h>
int n, a[20][20], visited1[20], visited2[20], source;
```

```

void read_data()
{
    int i,j;
    printf("enter the number of nodes\n");
    scanf("%d",&n);
    printf("enter the adjacency matrix\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
}
void print_data(int visited[])
{
    int i;
    for(i=0;i<n;i++)
    {
        if(visited[i]==0)
            printf("\nvertex %d is not reachable\n",i);
        else
            printf("\nvertex %d is reachable\n",i);
    }
}
void BFS()
{
    int f = 0, r = 0, q[20], i, j;
    q[r]=source;
    visited1[source]=1;
    while(f<=r)
    {
        i=q[f++];
        printf("--%d--",i);
        for(j=0;j<n;j++)
        {
            if(a[i][j]==1 && visited1[j]==0)
            {
                visited1[j]=1;
                q[++r]=j;
            }
        }
    }
}
void DFS(int src, int *cnt)
{
    int i,j;
    printf("--%d--",src);
    visited2[src]=1;
    for(j=0;j<n;j++)
    {
        if(a[src][j]==1 && visited2[j]==0)

```

```

        {
            (*cnt)++;
            DFS(j,cnt);
        }
    }
}

void main()
{
    int i,choice,a,*count=&a;
    read_data();
    printf("\t\t**ADJANCEY MATRIX FOR CITIES HAS CREATED SUCCESSFULLY**\n");
    while(1)
    {
        printf("\n1.BFS\n2.DFS\n3.Exit");
        printf("\nEnter Your Choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: for(i=0;i<n;i++)
                    visited1[i]=0;
                    printf("Enter the source vertex between 0 to %d\n",n-1);
                    scanf("%d",&source);
                    BFS();
                    print_data(visited1);
                    break;
            case 2: for(i=0;i<n;i++)
                    visited2[i]=0;
                    printf("Enter the source vertex between 0 to %d\n",n-1);
                    scanf("%d",&source);
                    a=0;
                    DFS(source,count);
                    print_data(visited2);
                    if(*count==n-1)
                        printf("graph is connected\n");
                    else
                        printf("graph is not connected\n");
                    break;
            case 3: exit(0);
            default:printf("\nEnter proper Choice");
        }
    }
}

```

OUTPUT:

enter the number of nodes

3

enter the adjacency matrix

0 1 1

1 0 1

1 1 0

ADJANCEY MATRIX FOR CITIES HAS CREATED SUCCESSFULLY

1.BFS

2.DFS

3.Exit

Enter Your Choice: 1

Enter the source vertex between 0 to 2

0

--0.....1.....2

vertex 0 is reachable

vertex 1 is reachable

vertex 2 is reachable

1.BFS

2.DFS

3.Exit

Enter Your Choice: 2

Enter the source vertex between 0 to 2

1

--1__0__2

vertex 0 is reachable

vertex 1 is reachable

vertex 2 is reachable

graph is connected

1.BFS

2.DFS

3.Exit

Enter Your Choice: 3

12. Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function $H: K \rightarrow L$ as $H(K)=K \bmod m$ (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>
```

```
#define MAX 5
```

```

struct employee
{
    int id;
    char name[15];
};
typedef struct employee EMP;
EMP emp[MAX];
int a[MAX];
int create(int num)
{
    int key;
    key = num % 100;
    return key;
}
int getemp(EMP emp[],int key)
{
    printf("\nEnter emp id: ");
    scanf("%d",&emp[key].id);
    printf("\nEnter emp name: ");

    gets(emp[key].name);
    return key;
}
void display()
{
    int i, ch;
    while(1)
    {
        printf("\n1.Display ALL\n2.Filtered Display");
        printf("\nEnter the choice: ");
        scanf("%d",&ch);
        if(ch == 1)
        {
            printf("\nThe hash table is:\n");
            printf("\nHTKey\tEmpID\tEmpName");
            for(i=0; i<MAX; i++)
                printf("\n%d\t%d\t%s", i, emp[i].id, emp[i].name);
        }
        else if (ch==2)
        {
            printf("\nThe hash table is:\n");
            printf("\nHTKey\tEmpID\tEmpName");
            for(i=0; i<MAX; i++)
                if(a[i] != -1)
                {
                    printf("\n%d\t%d\t%s", i, emp[i].id, emp[i].name);
                    continue;
                }
        }
    }
}

```

```

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

```



```

    }
}
void main()
{
    int num, key, i;
    int ans = 1;
    printf("\nCollision handling by linear probing: ");
    for (i=0; i < MAX; i++)
        a[i] = -1;
    do
    {
        printf("\nEnter the data: ");
        scanf("%d", &num);
        key=create(num);
        linear_prob(key,num);
        printf("\nDo you wish to continue? (1/0): ");
        scanf("%d",&ans);
    }while(ans);
    display(emp);
}

```

OUTPUT:

Collision handling by linear probing:
Enter the data: 10
Collision Detected...!!!

Collision avoided successfully using LINEAR PROBING

A. Lab write-up and execution rubrics(Max marks 8)

		Good	Average
a.	Understanding of problem (3 marks)	Demonstrate good knowledge of language constructs and programming practice (3)	Moderate understanding of language constructs (1)
b.	Execution and testing (3marks)	Program handles all possible conditions and results with satisfying results. (3)	Partial executions /poor error handling (1)
c.	Result and documentation (2 marks)	Meticulous documentation of changes made and results obtained are in proper format (2)	Moderate formatting of output and average documentation (1)

B. VIVA Rubrics: (Max marks 2)

		Good	Average
	Conceptual understanding (2 marks)	Explain the complete program with the related concepts.(5)	Adequately provides explanation.(3)

C. Marks for Lab Record:10