

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

PROGRAM 01:

/*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.*/

//Required Header files

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define the structure for a day

struct Day {

 char *name; // Dynamic string for day name

 int date; // Integer for date

 char *description; // Dynamic string for activity description

};

// Function to create the calendar

void create(struct Day calendar[7]) {

 for (int i = 0; i < 7; i++) {

 // dynamically allocate memory for the day name and description

 calendar[i].name = (char *)malloc(50 * sizeof(char)); // Assuming max length of day name is 50 characters

 calendar[i].description = (char *)malloc(100 * sizeof(char)); // Assuming max length of description is 100 characters

 }

```
}

// Function to read the weekly activity details
void read(struct Day calendar[7]) {
    for (int i = 0; i < 7; i++) {
        printf("Enter name for Day %d: ", i + 1);
        scanf("%s", calendar[i].name);
        printf("Enter date for Day %d: ", i + 1);
        scanf("%d", &calendar[i].date);
        printf("Enter description for Day %d: ", i + 1);
        scanf("%s", calendar[i].description);
    }
}

// Function to display the weekly activity details
void display(struct Day calendar[7]) {
    for (int i = 0; i < 7; i++) {
        printf("Day Name: %s\n", calendar[i].name);
        printf("Date: %d\n", calendar[i].date);
        printf("Description: %s\n", calendar[i].description);
        printf("\n");
    }
}

int main() {
    struct Day calendar[7];

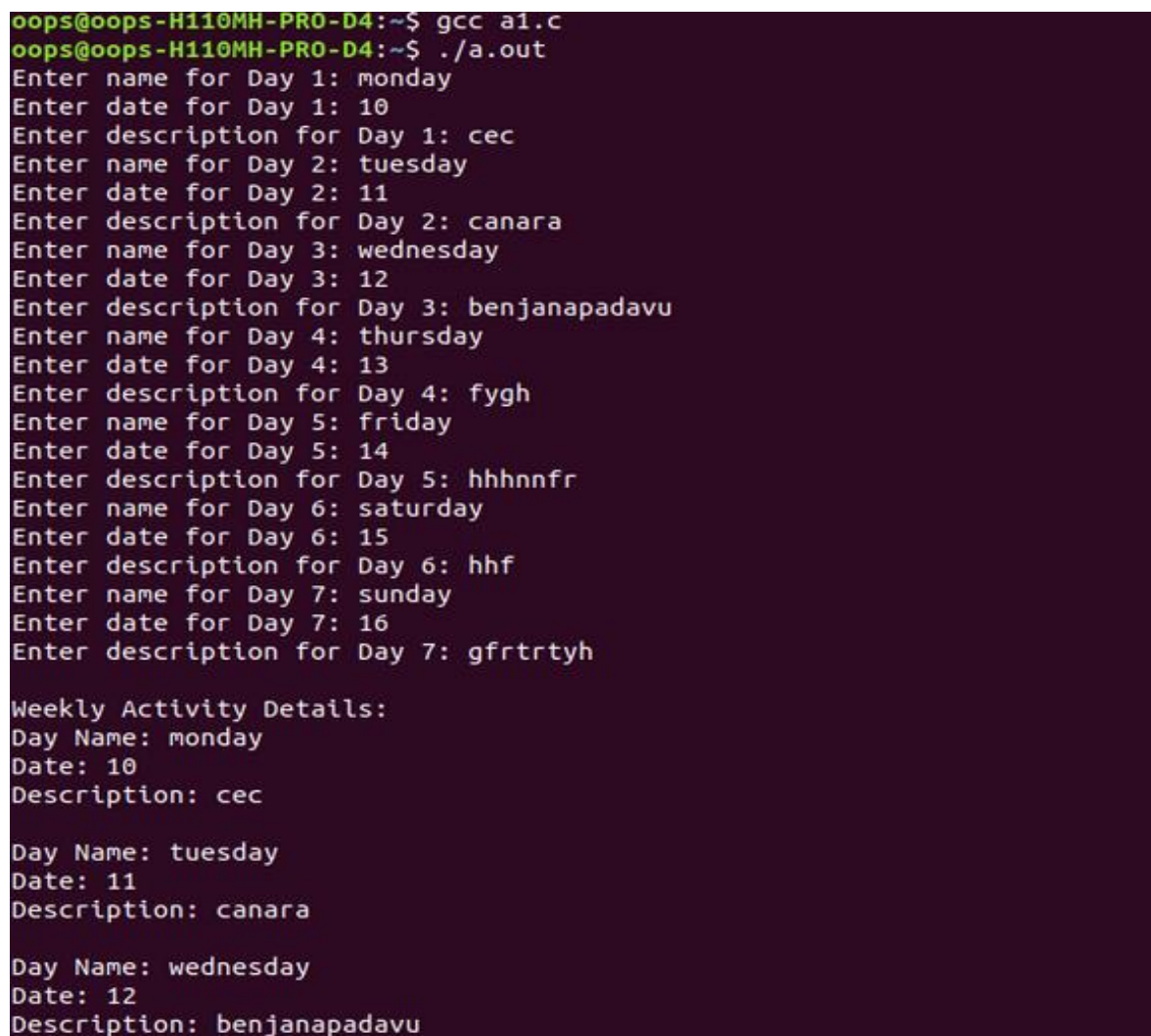
    // Call the create function to create the calendar
    create(calendar);

    read(calendar);

    // Display the weekly activity details
```

```
printf("\nWeekly Activity Details:\n");  
display(calendar);  
  
// Free dynamically allocated memory  
for (int i = 0; i < 7; i++) {  
    free(calendar[i].name);  
    free(calendar[i].description);  
}  
  
return 0;  
}
```

OUTPUT:



```
oops@oops-H110MH-PRO-D4:~$ gcc a1.c  
oops@oops-H110MH-PRO-D4:~$ ./a.out  
Enter name for Day 1: monday  
Enter date for Day 1: 10  
Enter description for Day 1: cec  
Enter name for Day 2: tuesday  
Enter date for Day 2: 11  
Enter description for Day 2: canara  
Enter name for Day 3: wednesday  
Enter date for Day 3: 12  
Enter description for Day 3: benjanapadavu  
Enter name for Day 4: thursday  
Enter date for Day 4: 13  
Enter description for Day 4: fygh  
Enter name for Day 5: friday  
Enter date for Day 5: 14  
Enter description for Day 5: hhhnnfr  
Enter name for Day 6: saturday  
Enter date for Day 6: 15  
Enter description for Day 6: hhf  
Enter name for Day 7: sunday  
Enter date for Day 7: 16  
Enter description for Day 7: gfrtrtyh  
  
Weekly Activity Details:  
Day Name: monday  
Date: 10  
Description: cec  
  
Day Name: tuesday  
Date: 11  
Description: canara  
  
Day Name: wednesday  
Date: 12  
Description: benjanapadavu
```

```
Enter description for Day 5: hhhnnfr
Enter name for Day 6: saturday
Enter date for Day 6: 15
Enter description for Day 6: hhf
Enter name for Day 7: sunday
Enter date for Day 7: 16
Enter description for Day 7: gfrtrtyh
```

Weekly Activity Details:

Day Name: monday

Date: 10

Description: cec

Day Name: tuesday

Date: 11

Description: canara

Day Name: wednesday

Date: 12

Description: benjanapadavu

Day Name: thursday

Date: 13

Description: fygh

Day Name: friday

Date: 14

Description: hhhnnfr

Day Name: saturday

Date: 15

Description: hhf

Day Name: sunday

Date: 16

Description: gfrtrtyh

PROGRAM 02:

/* Design, Develop and Implement 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>
```

```
// user defined parameterized functions prototype
```

```
int length(char[]);
```

```
int match(char[], char[], int, int);
```

```
int compare(char[], char[]);
```

```
// main ( ) to drive the code
```

```
void main()
```

```
{
```

```
// local declarations
```

```
    char str[100], pat[100], rep[100], t[100], d[100];
```

```
        int ls, lp, lr;
```

```
        int i, j, l, ll=0, k=0;
```

```
// Read strings
```

```
    printf("\n Enter main String :");
```

```
    gets(str);
```

```
    printf("\n Enter pattern String :");
```

```
    gets(pat);
```

```
    printf("\n Enter Replace String :");
```

```
    gets(rep);
```

```
// find the length of strings
```

```
ls = length(str);
lp = length(pat);
lr = length(rep);

// check weather pattern matches or not
if (ls < lp || !match(str, pat, ls, lp))
    printf("\n Pattern Not found!!!\n");
else // if pattern matches
{
    // Replaces all occurrences of pattern in string
    for (i=0;i<ls;i++) // loop for the size of main string
    {
        for(j=0; j<lp; j++) // copy the first chunk form main string of pattern length
            t[j]=str[i+j]; // store in temporary string
        t[j]='\0'; // end the temporary string
        if (compare(t,pat)==0) // compare temporary string with pattern string
        {
            k++; // count number of replaces
            for(l=0;l<lr;l++)
                d[l1++]=rep[l];
            i=i+(lp-1);
        }
        else
            d[l1++]=str[i];
    }

    d[l1]='\0'; // end of destination string

// display the results

printf("\n The number of string replaced =%d \n", k);
```

```
    printf("\n Original String =%s \n", str);
    printf("\n Pattern String =%s \n", pat);
    printf("\n Replace String =%s \n", rep);
    printf("\n String after replacement =%s \n", d);
}
}

// function to find the length of string
int length(char str[])
{
    int i, len =0;
    for (i = 0; str[i] != '\0'; i++)
        len++;
    return len;
}

// function to compare two strings
int compare(char str1[],char str2[])
{
    int i=0;
    while (str1[i] == str2[i] && str1[i] != '\0')
        i++;
    if (str1[i] > str2[i])
        return 1;
    else if (str1[i] < str2[i])
        return -1;
    else
        return 0; // both strings are same
}
```

```
// function to match pattern string in given string
int match(char text[], char pattern[], int ls, int lp)
{
    int c, d, e;
    for (c = 0; c <= ls - lp; c++) { e = c;
        for (d = 0; d < lp; d++)
        {
            if (pattern[d] == text[e])
                e++;
            else
                break;
        }
        if (d == lp)
            return 1; // return 1 if there is a pattern match
    }
    return 0; // return 0 if pattern does not found
}
```

OUTPUT:

```
admin-pc@admin-31:~$ gedit program2.c
admin-pc@admin-31:~$ gcc program2.c
admin-pc@admin-31:~$ ^C
admin-pc@admin-31:~$ ./a.out

Enter main String :good morning
Enter pattern String :morning
Enter Replace String :evening
The number of string replaced =1
Original String =good morning
Pattern String =morning
Replace String =evening
String after replacement =good evening
```


PROGRAM 03:

/* Design, Develop and Implement 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>

#define MAX 5

int stack[MAX];

int top=-1;

//a. Push an Element on to Stack

void push()

{

    int item;

    // Stack Overflow situations

    if(top==(MAX-1))

        printf("\n Stack Overflow");

    else

    {

        printf("\n Enter the element to be pushed :");

        scanf("%d",&item);

        // pushing element to the top of stack

        stack[++top]=item;
```

```
    }  
}  
  
//b. Pop an Element from Stack  
  
void pop()  
{  
    // Stack Underflow situations  
  
    if(top== -1)  
        printf("\n Stack Underflow");  
  
    else  
        printf(" \nPoped element is %d ",stack[top--]); // popping element from the top of stack  
}  
  
//e. Display the status of Stack  
  
void display()  
{  
    int i;  
  
    if(top== -1)  
        printf("\n Sorry Empty Stack");  
  
    else  
    {  
        printf("\nThe elements of the stack are\n");  
        for(i=top;i>=0;i--)  
            printf("stack[%d] = %d\n",i, stack[i]);  
    }  
}  
  
//c. Demonstrate how Stack can be used to check Palindrome  
  
void palindrome()  
{
```

```
int i,count=0;

for(i=0; i<=(top/2); i++)

{

    if(stack[i] == stack[top-i])

        count++;

}

if((top/2+1)==count)

    printf("\n Stack contents are Palindrome");

else

    printf("\nStack contents are not palindrome");

}

void main()

{

    int choice;

    while(1)

    {

        printf("\n STACK OPERATIONS\n");

        printf("1.Push\n 2.Pop\n 3.Display\n 4.Palindrome\n 5.Exit\n");

        printf("Enter your choice\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:push();

                break;

            case 2:pop();

                break;

            case 3:display();
```

```
        break;

    case 4:palindrome();

        break;

    case 5:return;

    default: printf("Invalid choice\n");

    }

}

}
```

OUTPUT:

```
admin-pc@admin-31:~$ gedit program3.c
admin-pc@admin-31:~$ gcc program3.c
admin-pc@admin-31:~$ ./a.out

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :10

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :20

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :30
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :40
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1
```

```
Enter the element to be pushed :50
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1
```

```
Stack Overflow
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
3
```

```
The elements of the stack are
stack[4] = 50
stack[3] = 40
stack[2] = 30
stack[1] = 20
stack[0] = 10
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Poped element is 50
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Poped element is 40
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Poped element is 30
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Poped element is 20
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Poped element is 10
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
2

Stack Underflow
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
3

Sorry Empty Stack
```

```
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :1

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :2

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :1

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
4

Stack contents are Palindrome
STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
1

Enter the element to be pushed :4

STACK OPERATIONS
1.Push
2.Pop
3.Display
4.Palindrome
5.Exit
Enter your choice
4

Stack contents are not palindrome
```

PROGRAM 04:

/*Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression.

Program should support for both parenthesized and free parenthesized expressions with the operators:

+, -, *, /, %(Remainder), ^(Power) and alphanumeric operands. */

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

```
void infix_to_postfix();
```

```
void push(char);
```

```
char pop();
```

```
int priority(char);
```

```
char infix[30], postfix[30],stack[30];
```

```
int top=-1;
```

```
void main()
```

```
{
```

```
    printf("Enter the valid Infix expression \n");
```

```
    scanf("%s",infix);
```

```
    infix_to_postfix();
```

```
    printf("\n Infix expression : %s",infix);
```

```
    printf("\n Postfix expression : %s\n",postfix);
```

```
}
```

```
// push symbol to stack
```

```
void push(char item)
```

```
{
```

```
    stack[++top]=item;
```

```
} // end of function push
```

```
// pop symbol from stack
```

```
char pop()
```



```
{  
    return stack[top--];  
} // end of function pop  
  
// check the priority of operator  
int priority(char symb)  
{  
    int p;  
    switch(symb)  
    {  
        case '+':  
        case '-': p=1;  
        break;  
        case '*':  
        case '/':  
        case '%':    p=2;  
        break;  
        case '^':  
        case '$':    p=3;  
        break;  
        case '(':  
        case ')': p=0;  
        break;  
        case '#':    p=-1;  
        break;  
    } // end of switch  
    return p;  
} // end of function priority
```

//converting an Infix Expression to Postfix Expression

void infix_to_postfix()

{

int i=0,j=0;

char symb,temp;

push('#');

for(i=0;infix[i]!='\0';i++)

{

symb=infix[i];

switch(symb)

{

case '(': push(symb); // push all symbols inside the (to top of stack

break;

case ')': temp=pop(); //pop symbol from top of stack

while(temp!='(') //pop all symbols from top of stack and store in postfix until (

{

postfix[j++]=temp;

temp=pop();

} // end of while

break;

case '+':

case '-':

case '*':

case '/':

case '%':

case '^':

case '\$': while(priority(stack[top])>=priority(symb)) // check for priority of operator

```

        {
            temp=pop();
            postfix[j++]=temp;
        }

        push(symb);

        break;

        default: postfix[j++]=symb;
    } // end of switch

    } // end of for

    while(top>0) // pop remaining all symbols form top of stack and store to
postfix

    {

        temp=pop();

        postfix[j++]=temp;

    } // end of while

    postfix[j]='\0'; // end string postfix

    } // end of function infix_to_postfix

```

OUTPUT:

```

admin-pc@admin-31:~$ gedit program4.c
admin-pc@admin-31:~$ gcc program4.c
admin-pc@admin-31:~$ ./a.out
Enter the valid Infix expression
a+b

Infix expression : a+b
Postfix expression : ab+
admin-pc@admin-31:~$ ./a.out
Enter the valid Infix expression
(a*b)+c

Infix expression : (a*b)+c
Postfix expression : ab*c+
admin-pc@admin-31:~$ ./a.out
Enter the valid Infix expression
(a+(b*c))/(d-e))

Infix expression : (a+(b*c))/(d-e))
Postfix expression : abc*de-/+
```

PROGRAM 05:

/* Design, Develop and Implement a Program in C for the following Stack Applications

- a. Evaluation of postfix expression with single digit operands and operators: +, -, *, /, %, ^
- b. Solving Tower of Hanoi problem with N disks

*/

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

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

```
void push(float);
```

```
float pop();
```

```
void evaluate(char[]);
```

```
float stack[20];
```

```
int top=-1;
```

```
void main()
```

```
{
```

```
    int choice,n;
```

```
    char postfix[100];
```

```
    while(1) // infinite loop for menu
```

```
    {
```

```
        printf("\n STACK APPLICATIONS");
```

```
        printf("\n Enter your Choice: ");
```

```
        printf("\n 1. Evaluation of postfix expression with single digit operands and operators");
```

```
        printf("\n 2. Solving Tower of Hanoi problem with N disks");
```

```
        printf("\n 3. Exit \n");
```

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

```
        switch(choice)
```

```
        {
```

```
case 1 : printf("Enter a valid postfix expression\n");
        scanf("%s",postfix);
        evaluate(postfix);
        break;

case 2 : printf("\n Enter the number of discs:\n");
        scanf("%d",&n);
        tower(n,'A','C','B');
        printf("\n Total number of moves are %d",(int)pow(2,n)-1);
        break;

case 3 : return;

default : printf("\n Invalid Choice");
} // end of switch

} // end of menu

} // end of main

// push item to stack
void push(float item)
{
    stack[++top]=item;
} // end of push

// pop item from stack
float pop()
{
    return stack[top--];
} // end of pop

// function to postfix expression with single digit operands and operators: +, -, *, /, %, ^
void evaluate(char postfix[100])
```

```
{
    int i;
    float op1, op2, res;
    char symb;

    for(i=0;postfix[i]!='\0';i++) // repeate until end of string
    {
        symb=postfix[i];
        if(isdigit(symb)) // check for digit or not
            push(symb-'0'); // if digit push to top of stack -'0' is for ascii to number
conversion
        switch(symb)
        {
            case '+':op2=pop();
                op1=pop();
                res=op1+op2;
                push(res);
                break;
            case '-':op2=pop();
                op1=pop();
                res=op1-op2;
                push(res);
                break;
            case '*':op2=pop();
                op1=pop();
                res=op1*op2;
                push(res);
                break;
```

```
case '/':op2=pop();
    op1=pop();
    if(op2==0)
    {
        printf("Division by zero Error\n");
        return;
    }
    res=op1/op2;
    push(res);
    break;
case '%': op2=pop();
    op1=pop();
    if(op2==0)
    {
        printf("Division by zero Error\n");
        return;
    }
    res=(int)op1%(int)op2;
    push(res);
    break;
case '^':op2=pop();
    op1=pop();
    res=pow(op1,op2);
    push(res);
    break;

    } // end of switch

} // end of for
```

```
        res=pop(); // pop the final answer from top of stack

        if(top== -1)

            printf("\n Result: %f\n ",res); // Display the final answer
    else
    {
        printf("\nINVALID POSTFIX EXPRESSION\n");

        top=-1;
    }
} // end of evaluate function

// Recursive function for Solving Tower of Hanoi problem with N disks

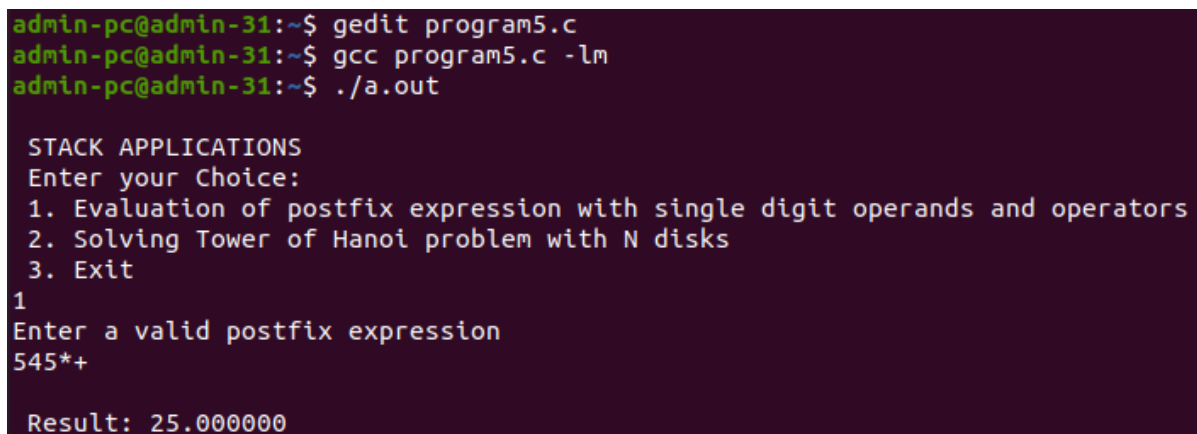
void tower(int n,int source,int destination,int aux)
{
    if(n==0)

        return;

    tower(n-1,source,aux,destination);

    printf("\n Move disc %d from %c to %c",n,source,destination);

    tower(n-1,aux,destination,source);
} // end of tower function
```



```
admin-pc@admin-31:~$ gedit program5.c
admin-pc@admin-31:~$ gcc program5.c -lm
admin-pc@admin-31:~$ ./a.out

STACK APPLICATIONS
Enter your Choice:
1. Evaluation of postfix expression with single digit operands and operators
2. Solving Tower of Hanoi problem with N disks
3. Exit
1
Enter a valid postfix expression
545*+

Result: 25.000000
```


STACK APPLICATIONS

Enter your Choice:

1. Evaluation of postfix expression with single digit operands and operators
2. Solving Tower of Hanoi problem with N disks
3. Exit

1

Enter a valid postfix expression

123++

Result: 6.000000

STACK APPLICATIONS

Enter your Choice:

1. Evaluation of postfix expression with single digit operands and operators
2. Solving Tower of Hanoi problem with N disks
3. Exit

2

Enter the number of discs:

3

Move disc 1 from A to C

Move disc 2 from A to B

Move disc 1 from C to B

Move disc 3 from A to C

Move disc 1 from B to A

Move disc 2 from B to C

Move disc 1 from A to C

Total number of moves are 7

STACK APPLICATIONS

Enter your Choice:

1. Evaluation of postfix expression with single digit operands and operators
2. Solving Tower of Hanoi problem with N disks
3. Exit

3

PROGRAM 06:

/* 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>

#define MAX 6

int front=0,rear=0;

char q[MAX];

//a. Insert an Element on to Circular QUEUE

void insert()
{
    char item;

    if(front==(rear+1)%MAX) // Check for overflow
    {
        printf("\n Circular Queue Overflow\n");
        return;
    }

    rear=(rear+1)%MAX;

    printf("\n Enter the character to be inserted: ");

    fflush(stdin);

    scanf("%c",&item);
```

```
    q[rear]=item;
} // end of insert

//b. Delete an Element from Circular QUEUE

void delete()
{
    if(front==rear) // check for Underflow
    {
        printf("\n Circular Queue Underflow\n");
        return;
    }
    front=(front+1)%MAX;
    printf("\n Deleted character is: %c ",q[front]);
} // end of delete

// d. Display the status of Circular QUEUE

void display()
{
    int i;
    if(front==rear)
    {
        printf("\nCircular Queue is Empty\n");
        return;
    }
    printf("\n Contents of Queue is:\n");
    for(i=(front+1)%MAX; i!=rear; i=(i+1)%MAX)
        printf("%c\t",q[i]);
    printf("%c\t",q[i]);
} // end of display
```

```
void main()
{
    int choice;
    while(1)
    {
        printf("\n CIRCULAR QUEUE OPERATIONS");
        printf("\n Enter the choice");
        printf("\n 1.Insert\n 2.Delete\n 3.Display\n 4.Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                insert();
                break;
            case 2:delete();
                break;
            case 3:display();
                break;
            case 4:return;
            default : printf("\n Invalid Choice \n");
        }// end of switch
    } // end of menu loop
} //end of main
```

OUTPUT:

```
admin-pc@admin-31:~$ gedit program6.c
admin-pc@admin-31:~$ gcc program6.c
admin-pc@admin-31:~$ ./a.out
```

```
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
3
```

```
Circular Queue Empty
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
1
```

```
Enter the element to be inserted: 1
```

```
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
1
```

```
Enter the element to be inserted: 2
```

```
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
1
```

```
Enter the element to be inserted: 3
```

```
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
1
```

```
Enter the element to be inserted: 4
```

```
~~Circular Queue Overflow~~
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
3
```

```
Contents of Queue is:
1 2 3
```

```
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
2

Deleted element from the queue is: 1
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
3

Contents of Queue is:
2 3
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
2

Deleted element from the queue is: 2
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
3

Contents of Queue is:
3
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
2

Deleted element from the queue is: 3
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
3

Circular Queue Empty
CIRCULAR QUEUE OPERATIONS
Enter the choice
1.Insert
2.Delete
3.Display
4.Exit
2

~~Circular Queue Underflow~~
```

PROGRAM 07:

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

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

*/

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

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

```
//NODE for singly linked list
```

```
struct node
```

```
{
```

```
char usn[20],name[10],branch[5];
```

```
unsigned long long int phno;
```

```
int sem;
```

```
struct node *link; // link for node of type node
```

```
};
```

```
typedef struct node * NODE;
```

```
NODE temp,FIRST=NULL;
```

```
// function to create a node
```

```
NODE getnode()
```

```
{
```

```
NODE x;
```

```
x=(NODE)malloc(sizeof(struct node)); //Dynamic memory allocation for NODE
```

```
x->link=NULL; //by node default link is set to NULL to avoid dangling pointer

return x;

}

//Function to read linked list node information

void read()

{

temp=getnode();

printf("Enter USN: ");

scanf("%s",temp->usn);

printf("Enter NAME: ");

scanf("%s",temp->name);

printf("Enter Branch: ");

scanf("%s",temp->branch);

printf("Enter phone Number: ");

scanf("%llu",&temp->phno);

printf("Enter Semester: ");

scanf("%d",&temp->sem);

}

// Create a SLL of N Students Data by using front insertion

void create_SLL()

{

int n,i;

printf("Enter the number of students: ");

scanf("%d",&n);

for(i=1;i<=n;i++) // read in student details

{

printf("\n Enter the details of %d students\n",i);
```



```
read(); // call read() to read student details

if(FIRST==NULL) // check for empty list

    FIRST=temp; // if true set new node as FIRST node of list

else{

    temp->link=FIRST; // otherwise link the old node to new node

    FIRST=temp; // make new node as FIRST (INSERT FRONT)

}

}

}

//Display the status of SLL and count the number of nodes in it

void display_count()

{

    int count=1;

    temp=FIRST;

    printf("Student details:\n");

    if(FIRST==NULL) // check for empty list

        printf("Student detail is NULL and count is 0\n");

    else

    {

        printf("\nUSN\tNAME\tBRANCH\tPHNO\tSEMESTER\n");

        while(temp->link!=NULL) // iterate nodes of SLL until end node

        {

            count++;

            printf("%s\t%s\t%s\t%llu\t%d\n",temp->usn,temp->name,temp->branch,temp-

            >phno,temp->sem);

            temp=temp->link; // next node

        }

    }
```

```
printf("%s\t%s\t%s\t%llu\t%d",temp->usn,temp->name,temp->branch,temp->phno,temp->sem);
```

```
printf("\n Student count is %d\n",count);
```

```
}
```

```
return;
```

```
}
```

```
// Perform Insertion at front of SLL
```

```
void insert_front()
```

```
{
```

```
printf("Enter the details of student\n");
```

```
read();
```

```
if(FIRST==NULL)
```

```
    FIRST=temp;
```

```
else{
```

```
    temp->link=FIRST;
```

```
    FIRST=temp;
```

```
}
```

```
}
```

```
// Perform Insertion at End of SLL
```

```
NODE insert_end()
```

```
{
```

```
    NODE last=FIRST;
```

```
printf("Enter the details of student\n");
```

```
read();
```

```
if(FIRST==NULL)
```

```
    FIRST=temp;
```

```
else
```

```
{  
    while(last->link!=NULL) // find last node iteratively  
        last=last->link;  
    last->link=temp; // connect new node to last node link  
}  
return FIRST;  
}
```

// Perform Deletion at front of SLL

```
void delete_front()
```

```
{  
    temp=FIRST;  
    if(FIRST==NULL) // Check for empty list  
        printf("List is empty\n");  
    else  
    {  
        printf("deleted element is %s\n",temp->usn);  
        FIRST=FIRST->link; //set 2nd node as FIRST node  
        free(temp); // delete previous 1st node  
    }  
    return;  
}
```

// Perform Deletion at End of SLL

```
void delete_end()
```

```
{  
    NODE last=NULL;  
    temp=FIRST;  
    if(FIRST==NULL) // Check for empty list
```

```
    printf("List is empty\n");
else if(FIRST->link==NULL) // // Check for single node in SLL
{
    printf("Deleted element is %s\n",temp->usrn);
    free(FIRST);
    FIRST=NULL;
}
else
{
    while(temp->link!=NULL)
    {
        last=temp;
        temp=temp->link;
    }
    last->link=NULL;
    printf("Deleted element is %s\n",temp->usrn);
    free(temp); // delete last node
}
return;
}

void main()
{
    int choice;
    while(1)
    {
        printf("\n1. Create SLL\n2.Display SSL\n3.Insert front\n4.Insert end\n5.Delete
front\n6.Delete end\n7.EXIT\n");
```

```
printf("Enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
    case 1:create_SLL();
        break;
    case 2:display_count();
        break;
    case 3:insert_front();
        break;
    case 4:insert_end();
        break;
    case 5:delete_front();
        break;
    case 6:delete_end();
        break;
    case 7:return;
    default:printf("\nInvalid choice\n");
}
}
}
```

OUTPUT:

```
admin-pc@admin-31:~$ gedit program7.c
admin-pc@admin-31:~$ gcc program7.c
admin-pc@admin-31:~$ ./a.out

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
1
Enter the number of students: 3

Enter the details of 1 students
Enter USN: 4CB22CG001
Enter NAME: ADITI
Enter Branch: CSD
Enter phone Number: 2754376252
Enter Semester: 3

Enter the details of 2 students
Enter USN: 4CB22CG002
Enter NAME: BHAVANA
Enter Branch: CSD
Enter phone Number: 35185178
Enter Semester: 3

Enter the details of 3 students
Enter USN: 4CB22CG003
Enter NAME: CHETHAN
Enter Branch: CSD
Enter phone Number: 233333333678
Enter Semester: 3

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
2
Student details:

USN      NAME      BRANCH  PHNO      SEMESTER
4CB22CG003    CHETHAN CSD      233333333678    3
4CB22CG002    BHAVANA CSD      35185178        3
4CB22CG001    ADITI   CSD      2754376252      3
Student count is 3
```

```
1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
3
Enter the details of student
Enter USN: 4CB22CG100
Enter NAME: NEWSTUDENT
Enter Branch: CSD
Enter phone Number: 32465879
Enter Semester: 3

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
2
Student details:

USN      NAME      BRANCH  PHNO      SEMESTER
4CB22CG100  NEWSTUDENTCSD  CSD      32465879      3
4CB22CG003  CHETHAN CSD      233333333678      3
4CB22CG002  BHAVANA CSD      35185178      3
4CB22CG001  ADITI CSD      2754376252      3
Student count is 4

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
5
deleted element is 4CB22CG100

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
2
Student details:

USN      NAME      BRANCH  PHNO      SEMESTER
4CB22CG003  CHETHAN CSD      233333333678      3
4CB22CG002  BHAVANA CSD      35185178      3
4CB22CG001  ADITI CSD      2754376252      3
Student count is 3
```

```

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
4
Enter the details of student
Enter USN: 4CB22CG200
Enter NAME: XYZ
Enter Branch: CSD
Enter phone Number: 325564
Enter Semester: 3

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
2
Student details:

USN      NAME      BRANCH  PHNO      SEMESTER
4CB22CG003  CHETHAN CSD      233333333678  3
4CB22CG002  BHAVANA CSD      35185178      3
4CB22CG001  ADITI   CSD      2754376252    3
4CB22CG200  XYZ     CSD      325564  3
Student count is 4

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
6
Deleted element is 4CB22CG200

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
2
Student details:

USN      NAME      BRANCH  PHNO      SEMESTER
4CB22CG003  CHETHAN CSD      233333333678  3
4CB22CG002  BHAVANA CSD      35185178      3
4CB22CG001  ADITI   CSD      2754376252    3
Student count is 3

1. Create SLL
2.Display SSL
3.Insert front
4.Insert end
5.Delete front
6.Delete end
7.EXIT
Enter your choice
7

```


PROGRAM 08:

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

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

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

```
struct node // structure to store employee details
```

```
{  
  
    char ssn[12],name[20],dept[25],desig[20];  
  
    unsigned long long int phno;  
  
    float sal;  
  
    struct node *prev;  
  
    struct node *next;  
  
};
```

```
typedef struct node *NODE; // renaming node as NODE
```

```
NODE temp, FIRST=NULL,END=NULL;
```

```
NODE getnode() // Create node
```

```
{  
  
    NODE x;  
  
    x=(NODE)malloc(sizeof(struct node));  
  
    x->prev=NULL;  
  
    x->next=NULL;
```

```
        return x;
    }

void read() // read details of employee
{
    temp=getnode();
    printf("Enter SSN:");
    scanf("%s",temp->:ssn);
    printf("Enter Name:");
    scanf("%s",temp->name);
    printf("Enter Dept:");
    scanf("%s",temp->dept);
    printf("Enter Designation:");
    scanf("%s",temp->desig);
    printf("Enter Phno:");
    scanf("%llu",&temp->phno);
    printf("Enter Salary:");
    scanf("%f",&temp->sal);
    return;
}

void Create_DLL() // Create a DLL of N Employees Data by using end insertion.
{
    int n,i=1;
    printf("Enter the number of Employees \n");
    scanf("%d",&n);
    while(i<=n)
    {
        printf("Enter the details of the %d employee\n", i++);
```

```

        read();

        if(FIRST==NULL) // if empty list new node will be the first node
        {

            FIRST=temp;

            END=temp;

        }

        else //otherwise find the last node and insert the new node
        {

            END->next=temp;

            temp->prev=END;

            END=temp;

        }

    } //e nd of while
} // end of create()

void display_count() //Display the status of DLL and count the number of nodes in it
{

    temp=FIRST;

    int count=0;

    if(FIRST==NULL) // check for empty list

        printf("the Employee detail is NULL and count is %d\n", count);

    else

    {

        printf("Employee details:\n");

        printf("SSN \t EMPLOYEE NAME \t DEPARTMENT \t DESIGNATION \t\n");
        printf("PHONE NUMBER \t SALARY");

        while(temp!=NULL) // display all nodes in the list
        {

```

```
        count++;

        printf("\n%s\t%s\t%s\t%s\t%llu\t%0.2f",temp->:ssn, temp->name, temp-
>dept,temp->desig,temp->phno,temp->sal);

        temp=temp->next;

    }

    printf("\n Employee count is %d\n",count);

    } // end of else

return;

} // end of display()

void Insertionfront() //Perform Insertion at front of DLL
{

    printf("Enter the details of the employee\n");

    read();

    if(FIRST==NULL) // if empty list new node will be the first node

    {

        FIRST=temp;

        END=temp;

    }

    else // otherwise insert the new node at front

    {

        temp->next=FIRST;

        FIRST->prev=temp;

        FIRST=temp;

    }

} // end of insert front

void Insertionend() //Perform Insertion at End of DLL

{
```

```
printf("Enter the details of the new employee\n");
read();
if(FIRST==NULL) // check for empty list
{
    FIRST=temp;
    END=temp;
}
else // otherwise find the last node and insert the new node
{
    END->next=temp;
    temp->prev=END;
    END=temp;
}
return ;
} // end of insert end

void Deletionfront() //Delete node from front of DLL
{
    temp=FIRST;
    if(FIRST==NULL) // check for empty list
        printf("List is empty\n");
    else if(FIRST==END) // otherwise check for single node in list
    {
        printf("deleted element is %s\n", temp->ssn);
        FIRST=NULL;
        END=NULL;
        free(temp);
    }
}
```

```
else // otherwise delete node from front of DLL
{
    printf("deleted element is %s\n", temp->:ssn);
    FIRST =FIRST->next;
    FIRST->prev=NULL;
    free(temp);
}
return;
} // end of deletefront

void Deletionend() // delete node at end of DLL
{
    temp = END;

    if(FIRST==NULL) // check for empty list
        printf("List is empty\n");
    else if(FIRST==END) // otherwise check for single node in list
    {
        printf("deleted element is %s\n", temp->:ssn);
        FIRST=NULL;
        END=NULL;
        free(temp);
    }
    else // otherwise delete end node from DLL
    {
        printf("deleted element is %s\n", temp->:ssn);
        END=END->prev;
        END->next=NULL;
        free(temp);
    }
}
```

```
    }

    return ;

} // end of deletionend

void main()
{
    int choice;

    while(1)
    {
        printf("\n 1 - Create DLL of N Employees \n 2 - Display DLL \n 3 - Insertion\n at front \n 4 - Insertion at end");

        printf("\n 5 - Deletion at front \n 6 - Deletion at end \n 7 - Exit\n");

        printf("Enter Your Choice: ");

        scanf("%d",&choice);

        switch(choice)
        {
            case 1 : Create_DLL();

            break;

            case 2 : display_count();

            break;

            case 3 : Insertionfront();

            break;

            case 4 : Insertionend();

            break;

            case 5 : Deletionfront();

            break;

            case 6 : Deletionend();

            break;
```

```

        case 7 : return;

        default: printf("Invalid Choice\n");

    } //end of switch

} // end of while

} // end of main

```

OUTPUT:

```

admin-pc@admin-31:~$ gedit program8.c
admin-pc@admin-31:~$ gcc program8.c
1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 1
Enter the number of Employees
2
Enter the details of the 1 employee
Enter SSN:1001
Enter Name:akash
Enter Dept:cash
Enter Designation:accountant
Enter Phno:54788987654
Enter Salary:30000
Enter the details of the 2 employee
Enter SSN:1002
Enter Name:deepak
Enter Dept:office
Enter Designation:clerk
Enter Phno:345678956789
Enter Salary:20000

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 2
Employee details:
SSN      EMPLOYEE NAME  DEPARTMENT  DESIGNATION  PHONE NUMBER  SALARY
1001    akash  cash    accountant    54788987654    30000.00
1002    deepak  office  clerk    345678956789    20000.00
Employee count is 2

```



```

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 3
Enter the details of the employee
Enter SSN:1003
Enter Name:nithin
Enter Dept:sales
Enter Designation:manger
Enter Phno:1234567890
Enter Salary:40000

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 2
Employee details:
SSN      EMPLOYEE NAME  DEPARTMENT  DESIGNATION  PHONE NUMBER  SALARY
1003     nithin  sales      manger      1234567890    40000.00
1001     akash   cash       accountant   54788987654   30000.00
1002     deepak  office     clerk        345678956789  20000.00
Employee count is 3

```

```

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 4
Enter the details of the new employee
Enter SSN:1004
Enter Name:poorva
Enter Dept:sales
Enter Designation:supervisor
Enter Phno:234567890
Enter Salary:30000

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 2
Employee details:
SSN      EMPLOYEE NAME  DEPARTMENT  DESIGNATION  PHONE NUMBER  SALARY
1003     nithin  sales      manger      1234567890    40000.00
1001     akash   cash       accountant   54788987654   30000.00
1002     deepak  office     clerk        345678956789  20000.00
1004     poorva  sales      supervisor   234567890     30000.00
Employee count is 4

```

```
1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 5
deleted element is 1003

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 2
Employee details:
SSN      EMPLOYEE NAME  DEPARTMENT  DESIGNATION  PHONE NUMBER  SALARY
1001    akash  cash  accountant  54788987654  30000.00
1002    deepak office clerk  345678956789  20000.00
1004    poorva sales supervisor  234567890  30000.00
Employee count is 3

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 6
deleted element is 1004

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 2
Employee details:
SSN      EMPLOYEE NAME  DEPARTMENT  DESIGNATION  PHONE NUMBER  SALARY
1001    akash  cash  accountant  54788987654  30000.00
1002    deepak office clerk  345678956789  20000.00
Employee count is 2

1 - Create DLL of N Employees
2 - Display DLL
3 - Insertion at front
4 - Insertion at end
5 - Deletion at front
6 - Deletion at end
7 - Exit
Enter Your Choice: 7
```

PROGRAM 09:

/* Develop a Program in C for the following operations on Singly Circular

Linked List (SCLL) with header nodes

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

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

Support the program with appropriate functions for each of the above operation */

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

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

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

```
#include <stdio_ext.h>
```

```
struct node // polynomial node
```

```
{
```

```
    int coef;
```

```
    int x,y,z;
```

```
    struct node *link;
```

```
};
```

```
typedef struct node *NODE;
```

```
NODE getnode() // create a node
```

```
{
```

```
    NODE x;
```

```
    x=(NODE)malloc(sizeof(struct node));
```

```
    return x;
```

```
} // end of getnode
```

```
NODE readpoly()
```

```
{
```

```
    NODE temp,head,cur;
```

```

    char ch;

    head=getnode(); // create a head node and set all values to -1 it is similar to FIRST in
SLL program

    head->coef=-1;

    head->x=-1;

    head->y=-1;

    head->z=-1;

    head->link=head; // self reference

    do

    {

        temp=getnode(); // create a polynomial node

        printf("\nEnter the coefficient and exponent in decreasing order\n");

        scanf("%d%d%d%d",&temp->coef,&temp->x,&temp->y,&temp->z );

        cur=head;

        while(cur->link!=head) // find the last node

            cur=cur->link;

        cur->link=temp; // connect new node to the last node

        temp->link=head; // point back to head

        printf("\nDo you want to enter more coefficients(y/n)");

        __fpurge(stdin); // to clear the stdin buffer

        scanf("%c",&ch);

    } while(ch == 'y' || ch == 'Y');

    return head; // return the polynomial list

} // end of readpoly

int compare(NODE a,NODE b) // function to compare the A and B polynomial nodes

{

    if(a->x > b->x)

```

```
        return 1;
    else if(a->x < b->x)
        return -1;
    else if(a->y > b->y)
        return 1;
    else if(a->y < b->y)
        return -1;
    else if(a->z > b->z)
        return 1;
    else if(a->z < b->z)
        return -1;
    return 0;
} // end of compare

void attach(int cf,int x1,int y1, int z1, NODE *ptr) // function to attach the A and B
polynomial node to C Polynomial
{
    NODE temp;
    temp=getnode();
    temp->coef=cf;
    temp->x=x1;
    temp->y=y1;
    temp->z=z1;
    (*ptr)->link=temp;
    *ptr=temp;
} // end of attach

NODE addpoly(NODE a,NODE b) // function to add polynomial A and B i.e, C=A+B
{
```

```
NODE starta,c ,lastc;

int sum,done=0;

starta=a;

a=a->link;

b=b->link;

c=getnode(); // create list C to store A+B

c->coef=-1;

c->x=-1;

c->y=-1;

c->z=-1;

lastc=c;

do{

switch(compare(a,b))

{

case -1:attach(b->coef,b->x,b->y,b->z,&lastc);

        b=b->link;

        break;

case 0:if(starta==a) done=1;

        else{

            sum=a->coef+b->coef;

            if(sum)

                attach(sum,a->x, a->y,a->z,&lastc);

            a=a->link;b=b->link;

        }

        break;

case 1: if(starta==a) done=1;

        attach(a->coef,a->x, a->y,a->z,&lastc);
```

```
        a=a->link;

        break;

    }

    }while(!done); // repeate until not done

    lastc->link=c; // point back to head of C

    return c; // return answer

}

void print(NODE ptr) // to print the polynomial

{

    NODE cur;

    cur=ptr->link;

    while(cur!=ptr) // To print from HEAD node till END node

    {

        printf("%d*x^%d*y^%d*z^%d",cur->coef,cur->x, cur->y, cur->z);

        cur=cur->link; // move to next node

        if (cur!=ptr)

            printf(" + ");

    }

    } // end of print

void evaluate(NODE ptr) // function to evaluate the final polynomial

{

    int res=0;

    int x,y,z, ex,ey,ez,cof;

    NODE cur;

    printf("\nEnter the values of x, y,z"); // read values of X, Y and Z

    scanf("%d", &x);

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



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

cur=ptr->link; // start with HEAD

while(cur!=ptr) // Repeat until the end of list
{
    ex=cur->x; // exponent of x
    ey=cur->y; // exponent of y
    ez=cur->z; // exponent of z
    cof=cur->coef; // coefficient
    res+=cof*pow(x,ex)*pow(y,ey)*pow(z,ez); // compute result for each
polynomial

    cur=cur->link; // move to next node
}

printf("\nresult: %d",res);
} // end of evaluate

void main(void)
{
    int i, ch;

    NODE a=NULL,b,c;

    while(1)
    {
        printf("\n1: Represent first polynomial A");
        printf("\n2: Represent Second polynomial B");
        printf("\n3: Display the polynomial A");
        printf("\n4: Display the polynomial B");
        printf("\n5: Add A & B polynomials"); // C=A+B
        printf("\n6: Evaluate polynomial C");
        printf("\n7: Exit");
```

```
printf("\n Enter your choice: ");
scanf("%d",&ch);
switch(ch)
{
    case 1: printf("\nEnter the elements of the polynomial A");
            a=readpoly();
            break;
    case 2: printf("\nEnter the elements of the polynomial B");
            b= readpoly();
            break;
    case 3: print(a); // display polynomial A
            break;
    case 4: print(b); // display polynomial A
            break;
    case 5: c=addpoly(a,b); // C=A+B
            printf("\nThe sum of two polynomials is: ");
            print(c); // display polynomial C
            printf("\n");
            break;
    case 6: evaluate(c); // Evaluate polynomial C
            break;
    case 7: return;
    default: printf("\nInvalid choice!\n");
} //end of switch
} // end of while
} // end of main
```

OUTPUT:

```

admin-pc@admin-31:~$ ./a.out

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 1

Enter the elements of the polynomial A
Enter the coefficient and exponent in decreasing order
7 5 2 1

Do you want to enter more coefficients(y/n)y

Enter the coefficient and exponent in decreasing order
9 5 3 1

Do you want to enter more coefficients(y/n)n

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 3
7*x^5*y^2*z^1 + 9*x^5*y^3*z^1

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 2

Enter the elements of the polynomial B
Enter the coefficient and exponent in decreasing order
5 5 2 1

Do you want to enter more coefficients(y/n)y

Enter the coefficient and exponent in decreasing order
1 3 3 1

Do you want to enter more coefficients(y/n)n

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 4
5*x^5*y^2*z^1 + 1*x^3*y^3*z^1

```

```
1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 5

The sum of two polynomials is:  $12x^5y^2z^1 + 9x^5y^3z^1 + 1x^3y^3z^1$ 

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 6

Enter the values of x, y,z 1 1

result: 22

1: Represent first polynomial A
2: Represent Second polynomial B
3: Display the polynomial A
4: Display the polynomial B
5: Add A & B polynomials
6: Evaluate polynomial C
7: Exit
Enter your choice: 7
```

PROGRAM 10:

Design, Develop and Implement 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
- e. Exit.

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

```
struct node // DLL node for Tree with left_link value and right_link
```

```
{
```

```
    int value;
```

```
    struct node *ltree;
```

```
    struct node *rtree;
```

```
};
```

```
typedef struct node* NODE;
```

```
NODE getnode() // create a node
```

```
{
```

```
    NODE x;
```

```
    x=(NODE) malloc (sizeof(struct node));
```

```
    x->ltree=NULL;
```

```
    x->rtree=NULL;
```

```
    return x;
```

```
} // end of getnode
```

```
NODE create(int item, NODE root) // create a BST for the given values
```

```
{
```

```
    NODE temp,cur,prev;
```

```
    int i , flag=0;
```

```
    temp=getnode(); // create node
```

```
temp->value=item; // store value

if(root==NULL) // if empty tree then new node itself is ROOT of BST
{
    root=temp;
    return root;
}

else // otherwise add new node to BST
{
    prev=NULL;
    cur=root;
    while (cur!=NULL) // find the leaf node
    {
        prev=cur;
        if(temp->value==cur->value)
        {
            flag=1;
            break;
        }

        cur=(temp->value<cur->value)?cur->ltree:cur->rtree; // condition to
choose left or right subtree
    }

    if(flag==0)
    {
        if(temp->value<prev->value) // if new node smaller than ROOT then make it
as left child
            prev->ltree=temp;

        else if(temp->value>prev->value) // otherwise if new node greater than ROOT
then make it as right child
```

```
        prev->rtree=temp;

    }

}

return root;

}

void in(NODE IN) // Recursive INORDER traversal function
{
    if(IN!=NULL) // if tree is not empty then traverse
    {
        in(IN->ltree); // left child

        printf("%d\t",IN->value); // Root

        in(IN->rtree); //right child
    }
}

void pre(NODE PRE) // Recursive PREORDER traversal function
{
    if(PRE!=NULL) // if tree is not empty then traverse
    {
        printf("%d\t",PRE->value); // Root

        pre(PRE->ltree); // left child

        pre(PRE->rtree); //right child
    }
}

void post(NODE POST) // Recursive POSTORDER traversal function
{
    if(POST!=NULL) // if tree is not empty then traverse
    {
```

```
        post(POST->ltree); // left child
        post(POST->rtree); //right child
        printf("%d\t",POST->value); // Root
    }
}

void search(NODE root) // Binary Search on BST for given Key element
{
    int item, found=0;
    NODE cur;
    printf("enter the element to be searched\n");
    scanf("%d",&item); // read key
    if(root==NULL) // check for empty tree
    {
        printf("tree is empty\n");
        return;
    }
    // for non-empty tree search for key in BST
    cur=root;
    while(cur!=NULL) // repeat until end
    {
        if(item==cur->value) // check key and root value is same SUCESSFULL
        SEARCH
        {
            found = 1; // set flag
            printf("Found key %d in tree\n",cur->value);
        }
        if(item<cur->value) // check key is greater than root value then choose right
        subtree
```



```
        cur=cur->ltree;

        else // otherwise choose left subtree

        cur=cur->rtree;

    } // end of while

    if(found==0) // if found flag is Zero then UNSUCCESSFUL SEARCH

        printf("Key not found\n");

} // end of search

void main()

{

    int choice, item, n, i;

    NODE root=NULL;

    while(1)

    {

        printf("\n1. Create BST of N Integers");

        printf("\n2. BST Traversal");

        printf("\n3. Binary Search");

        printf("\n4. Exit");

        printf("\nEnter Your Choice: ");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1: printf("\nEnter number elements : ");

                scanf("%d",&n);

                for (i =0; i<n; i++)

                {

                    printf("Enter the item to be inserted\n");

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

```
        root=create(item,root);
    }
    break;

    case 2:  if(root==NULL) // check for empty tree

        printf("Tree is empty\n");
    else
    { // for non-empty tree traverse BST
        printf("\n\nPREORDER traversal\n");
        pre(root);
        printf("\n\nINORDER traversal\n");
        in(root);
        printf("\n\nPOSTORDER traversal\n");
        post(root);
    }
    break;

    case 3:  search(root);

    break;

    case 4:  return;

    default: printf("\n Invalid Choice\n");

    } // end of switch

    } // end of while

} // end of main
```

OUTPUT:

```
admin-pc@admin-31:~$ gedit program10.c
admin-pc@admin-31:~$ gcc program10.c
admin-pc@admin-31:~$ ./a.out

1. Create BST of N Integers
2. BST Traversal
3. Binary Search
4. Exit
Enter Your Choice: 1

Enter number elements : 12
Enter the item to be inserted
6
Enter the item to be inserted
5
Enter the item to be inserted
9
Enter the item to be inserted
2
Enter the item to be inserted
8
Enter the item to be inserted
15
Enter the item to be inserted
24
Enter the item to be inserted
14
Enter the item to be inserted
7
Enter the item to be inserted
8
Enter the item to be inserted
5
Enter the item to be inserted
2

1. Create
Enter Your Choice: 4
```

PROGRAM 11:

/* Design, Develop and Implement 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>

int a[20][20], q[20], visited[20];

int n, i, j, f=0, r=-1;

void create_graph() // Create the Digraph using Adjacency matrix
{
    printf("\n Enter the number of cities: ");

    scanf("%d",&n);

    printf("\n Enter graph data in matrix form:\n");

    for(i=1;i<=n;i++)

        for(j=1;j<=n;j++)

            scanf("%d",&a[i][j]); // read adjacency matrix

    return;
}

void bfs(int v) // Reachability using Breadth First Search
{
    for(i=1;i<=n;i++)

        if(a[v][i] && !visited[i]) // check weather node is visited

            q[++r]=i; // if not add it Queue

    if(f<=r) // check for non empty Queue

        {

            visited[q[f]]=1; // set visited status for front node of Queue
```

```
        bfs(q[f++]); // recursive call BSF

    }

} // end of BSF

void main()
{
    int v, choice;

    while(1)
    {
        printf("\n1. Create a Digraph of N cities using Adjacency Matrix");

        printf("\n2. Print all the nodes reachable from a given starting node in a digraph using
BFS method") ;

        printf("\n3. Exit");

        printf("\n Enter Your Choice: ");

        scanf("%d",&choice);

        switch(choice)
        {
            case 1: create_graph();

                break;

            case 2: printf("Enter the source vertex: ");

                scanf("%d",&v);

                if((v<1)|| (v>n)) // check for valid source entry

                    printf("\nEnter a valid source vertex");

                else // if valid begin test for reachability

                {

                    for(i=1;i<=n;i++) // begin with assuming all cities are not visited

                        visited[i]=0;

                    visited[v]=1; // source is visited
```

```

        bfs(v); // cal BFS to check reachability

        printf("The reachable nodes from node %d:\n",v);

        for(i=1;i<=n;i++) // display reachable cities from the source city

            if(visited[i] && i !=v)

                printf("node %d\n",i);

    }

    break;

case 3:return;

default:printf("\nInvalid Choice");

} // end of switch

} // end of while

} // end of main

```

OUTPUT:

```

admin-pc@admin-31:~$ gedit program11.c
admin-pc@admin-31:~$ gcc program11.c
admin-pc@admin-31:~$ ./a.out

1. Create a Digraph of N cities using Adjacency Matrix
2. Print all the nodes reachable from a given starting node in a digraph using BFS method
3. Exit
Enter Your Choice: 1

Enter the number of cities: 5

Enter graph data in matrix form:
0 1 0 1 0
0 0 0 0 1
0 0 0 0 0
0 1 0 0 0
1 0 0 0 0

1. Create a Digraph of N cities using Adjacency Matrix
2. Print all the nodes reachable from a given starting node in a digraph using BFS method
3. Exit
Enter Your Choice: 2
Enter the source vertex: 1
The reachable nodes from node 1:
node 2
node 4
node 5

1. Create a Digraph of N cities using Adjacency Matrix
2. Print all the nodes reachable from a given starting node in a digraph using BFS method
3. Exit
Enter Your Choice: 3

```

PROGRAM 12:

/* Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Design and 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>
```

```
#define MAX 10
```

```
void linear_prob(int a[MAX])
```

```
{
```

```
    int flag, key, i, addrs, count;
```

```
    char ans;
```

```
    do
```

```
    {
```

```
        flag=0;
```

```
        count=0;
```

```
        printf("\n Enter 4 digit Key : ");
```

```
        scanf("%4d", &key); // read a key
```

```
        addrs=key%10; // generate single digit key for given key
```

```
        if(a[addrs]== -1) // check for empty entry in Hash table
```

```
            a[addrs] = key; // if yes the add to HT
```

```
        else // if entry exists then avoid collision
```

```
        {
```

```
            printf("\nCollision Detected...!!!\n");
```

```
            i=0;
```

```
            while(i<MAX) // check for next available empty location in HT
```

```
{  
    if (a[i]!=-1)  
count++; // count empty location in HT  
    i++;  
} // end of while  
if(count == MAX) // if HT is full then display HT and return  
{  
    printf("\n Hash table is full \n");  
    display(a); // Display HT  
    return;  
} // end of if  
printf("\nCollision avoided successfully using LINEAR PROBING\n");  
for(i=addr+1; i<MAX; i++) // if there is empty space after key in HT then  
make a entry in HT  
    if(a[i] == -1)  
    {  
        a[i] = key;  
        flag=1;  
        break;  
    } // end of if  
i=0;  
while((i<addr) && (flag==0)) // check for empty space before key in HT then  
make a entry in HT  
    {  
        if(a[i] == -1)  
        {  
            a[i] = key;  
            flag=1;  
        }
```



```
                break;

            } // end of if

            i++;

        } // end of while

    } // end of else

    printf("\n Do you wish to continue ? (y/n) ");

    fflush(stdin);

    scanf("%c",&ans);

    } while(ans=='y' || ans == 'Y') ;// end of do while
} // end of linear probe

void display(int a[MAX]) // display the HT
{
    int i;

    printf("\n the HASH TABLE is\n Addr's \t Key");

    for(i=0; i<MAX; i++)

        printf("\n %d \t %d ", i, a[i]);

} // end of display

void main()
{
    int a[MAX], i, choice;

    for (i=0; i<MAX; i++) // initialize HT with no entries

        a[i] = -1;

    while(1)

    {

        printf("\n Collision handling by Linear Probing ");

        printf("\n1. Insert into Hash table");

        printf("\n2. Display Hash table");
```

```
printf("\n3. Exit");  
printf("\n Enter your Choice : ");  
scanf("%d",&choice);  
switch (choice)  
{  
    case 1:  linear_prob(a);  
            break;  
    case 2:  display(a);  
            break;  
    case 3: return;  
    default: printf("\n Invalid Choice");  
} // end of switch  
} // end of while  
} // end of main  
.....
```

```
admin-pc@admin-31:~$ gedit program12.c
admin-pc@admin-31:~$ gcc program12.c
```

```
admin-pc@admin-31:~$ ./a.out

Collision handling by Linear Probing
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 1

Enter 4 digit Key : 1234

Do you wish to continue ? (y/n)
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 1

Enter 4 digit Key : 9789

Do you wish to continue ? (y/n)
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 1

Enter 4 digit Key : 1773
```

.....

```
Do you wish to continue ? (y/n)
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 2

the HASH TABLE is
Addr  Key
0      -1
1      -1
2      -1
3      1773
4      1234
5      -1
6      -1
7      -1
8      -1
9      9789
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 1

Enter 4 digit Key : 1663

Collision Detected...!!!

Collision avoided successfully using LINEAR PROBING

Do you wish to continue ? (y/n)
1. Insert into Hash table
2. Display Hash table
3. Exit
Enter your Choice : 2

the HASH TABLE is
Addr  Key
0      -1
1      -1
2      -1
3      1773
4      1234
5      1663
6      -1
7      -1
8      -1
9      9789
1. Insert into Hash table
2. Display Hash table
3. Exit
```

VIVA QUESTIONS:

PROGRAM 01:

1. What is a structure in C? How do you define a structure in C? Provide an example.
2. What is the difference between a structure and a union in C?
3. How do you access members of a structure?
4. Can a structure have another structure as a member? Explain with an example.
5. How do you pass a structure to a function? Can you pass it by value or do you have to pass it by reference?
6. What is the difference between an array of structures and a structure of arrays?
7. How can you dynamically allocate memory for a structure in C?
8. Explain the concept of bit-fields in structures.
9. What is the significance of the dot operator (.) when working with structures?
10. How do you define a pointer to a structure? How is it different from a pointer to a simple data type?
11. Explain the concept of dynamic memory allocation in C and how it relates to data structures?
12. How do you free memory allocated dynamically in C to avoid memory leaks?
13. How is dynamic memory allocated in C? Explain the functions used for allocation.
14. What is the role of the malloc function, and how is it different from calloc and realloc?
15. What is the purpose of the sizeof operator when working with dynamic memory allocation?

PROGRAM 02:

1. What is pattern matching in the context of string processing?
2. How can you perform a basic pattern matching operation in C without using any built-in functions or libraries?
3. How can you handle case-insensitive pattern matching when replacing strings in C?
4. What are the applications of pattern matching and string replacement in real-world programming scenarios?
5. Explain the difference between string comparison using == and strcmp()
6. What is the significance of the null-terminator ('\0') in string comparison?
7. Can you compare two strings using the less than (<) or greater than (>) operators directly? Why or why not?

PROGRAM 03:

1. What is a stack, and how is it implemented using an array in C?
2. Explain the basic operations of a stack (push and pop) and how they are performed in an array-based stack.
3. What is the role of the top pointer (or index) in an array-based stack? How do you initialize an empty stack using an array?
4. How do you check if a stack is empty or full when using an array-based implementation?
5. Discuss the advantages and limitations of using an array to implement a stack.
6. Can you implement a dynamic array-based stack that can resize itself when needed? If so, how?
7. Explain how to handle stack overflow and underflow in an array-based stack.
8. How can you retrieve the top element of the stack without popping it?
9. Describe a real-world scenario where you might use an array-based stack in a C program.
10. Explain how to implement multiple stacks using a single array.

PROGRAM 04:

1. What is infix notation, and what is postfix notation?
2. Explain the process of converting an infix expression to postfix notation using a stack data structure.
3. How do you handle operators of different precedence levels during the conversion process?
4. What should be done when encountering an open parenthesis "(" during the conversion? How do you handle the close parenthesis ")" during the infix to postfix conversion?
5. What is the significance of the right-associative operators in the conversion process?
6. How can you handle errors or invalid expressions during the conversion process?
7. What are the applications or use cases for infix to postfix conversion in programming?
8. How does the infix to postfix conversion algorithm handle unary operators and functions?
9. Can you convert complex infix expressions with nested parentheses to postfix notation using the same algorithm?
10. What are some advantages of postfix notation over infix notation for expression evaluation?

PROGRAM 05:**Postfix Expression Evaluation:**

1. Explain the process of evaluating a postfix expression.
2. How do you handle operands and operators in the postfix evaluation algorithm? Can you provide a step-by-step example of evaluating a postfix expression?
3. What is the role of a stack in the postfix expression evaluation?
4. What happens if the postfix expression is invalid or incomplete?
5. Can you evaluate postfix expressions with multi-digit operands? If so, how?
6. How do you handle division by zero or other potential errors during postfix evaluation?
7. Discuss the advantages of postfix notation in mathematical expressions.

Tower of Hanoi:

8. What is the Tower of Hanoi problem, and what are its rules?
9. Explain the recursive approach to solving the Tower of Hanoi problem.
10. How many moves are required to solve the Tower of Hanoi problem with N disks?
11. How does the number of moves required to solve the Tower of Hanoi increase with the number of disks?
12. Are there alternative methods to solve the Tower of Hanoi problem besides recursion?

PROGRAM 06:

1. What is a circular queue, and how does it differ from a regular queue?
2. Explain the concept of front and rear pointers in a circular queue.

3. How is memory allocated for a circular queue in an array-based implementation?
4. What is the maximum size (MAX) of the circular queue in this program?
5. Describe the basic operations of a circular queue, including enqueueing and dequeuing elements.
6. How do you determine if the circular queue is empty or full during enqueue and dequeue operations?
7. What are the advantages of using a circular queue over a regular queue in certain scenarios?
8. What happens when you attempt to enqueue an element into a full circular queue?
9. How do you handle dequeuing from an empty circular queue?
10. Can you demonstrate the concept of circular queue traversal?
11. How can you implement additional operations like checking the size of the circular queue and clearing it?
12. What is the importance of using a circular queue in situations where data continuously cycles through a fixed-size buffer?
13. Discuss potential issues or challenges that may arise when working with circular queues

PROGRAM 07:

1. What is a singly linked list, and how does it differ from other data structures like arrays?
2. Describe the structure of a node in a singly linked list.
3. How do you represent an empty singly linked list? How do you check if a linked list is empty in C?
4. What is the significance of the "head" pointer in a singly linked list?
5. How can you insert a new node at the beginning of a singly linked list?
6. Describe the process of inserting a new node at the end of a singly linked list.
7. What is the difference between inserting a node in the middle of the list and at a specific position in the list?
8. How do you delete a node from a singly linked list?
9. Discuss the importance of properly updating pointers when inserting or deleting nodes from a linked list.
10. Explain how to traverse a singly linked list from the beginning to the end.
11. What is a singly linked list's linear data structure and how is it useful in problem-solving?
12. What is a self-referential structure, and how is it used in a singly linked list node?
13. Discuss potential issues or challenges that may arise when working with singly linked lists.
14. Can you provide an example of a real-world application where a singly linked list is used in C?

PROGRAM 08:

1. What is a doubly linked list, and how does it differ from a singly linked list?
2. Explain the structure of a doubly linked list node. What information does it typically contain?
3. Describe the advantages of using a doubly linked list in comparison to a singly linked list.
4. What is the purpose of having both "next" and "previous" pointers in a doubly linked list?

5. Explain the process of inserting a new node at the beginning of a doubly linked list.
6. How do you insert a new node at the end of a doubly linked list?
7. Discuss the steps to insert a new node at a specific position within a doubly linked list.
8. How can you delete a node from a doubly linked list?
9. What precautions should be taken when deleting nodes from a doubly linked list?
10. How do you traverse a doubly linked list in both forward and reverse directions?

PROGRAM 09:

1. What is a singly circular linked list, and how does it differ from a singly linked list and a doubly circular linked list?
2. Describe the structure of a singly circular linked list node. What information does it typically contain?
3. Explain the concept of a circular linked list. How does it differ from a linear linked list?
4. How does a singly circular linked list ensure that there is no "end" or "beginning" of the list?
5. What are the advantages of using a circular linked list in comparison to a linear linked list?
6. Explain the process of inserting a new node at the beginning of a singly circular linked list.
7. How do you insert a new node at the end of a singly circular linked list?
8. Discuss the steps to insert a new node at a specific position within a singly circular linked list.
9. How can you delete a node from a singly circular linked list?
10. What precautions should be taken when deleting nodes from a singly circular linked list?
11. How do you traverse a singly circular linked list? What is the exit condition for traversal?
12. What are the potential challenges or drawbacks of using a singly circular linked list in C?
13. How can you represent a polynomial using a linked list in C?
14. Explain the structure of a node in a linked list representation of a polynomial?
15. How do you handle sparse polynomials where many coefficients are zero?

PROGRAM 10:

1. What is a binary search tree (BST), and how is it structured?
2. Explain the fundamental properties of a BST that differentiate it from other binary tree structures.
3. How does the BST property ensure efficient searching of elements in a BST?
4. What is the significance of the left and right sub trees in a BST?
5. Can a binary search tree contain duplicate values? Why or why not?
6. Describe the process of inserting a new element into a BST while maintaining the BST property.
7. How do you delete a node from a BST while preserving the BST property?
8. What is the difference between in-order, pre-order, and post-order traversal of a BST, and what are their use cases?
9. How do you find the minimum and maximum elements in a BST?
10. What is the height (depth) of a binary search tree, and how does it relate to the efficiency of operations?

PROGRAM 11:

1. What is a graph, and how is it represented using an adjacency matrix?
2. Explain the concept of directed and undirected graphs in the context of adjacency matrices.
3. How do you represent weighted edges in an adjacency matrix?
4. What is Breadth-First Search (BFS), and what is its primary purpose in graph traversal?
5. Explain the process of performing BFS on a graph using an adjacency matrix.
6. What data structure is commonly used to implement the BFS algorithm?
7. How do you find the shortest path between two nodes in an unweighted graph using BFS?
8. What is Depth-First Search (DFS), and how does it differ from BFS in terms of traversal strategy?
9. Explain the process of performing DFS on a graph using an adjacency matrix.
10. What data structure is commonly used to implement the DFS algorithm?
11. Compare and contrast the time complexities of BFS and DFS on graphs represented by adjacency matrices.

PROGRAM 12

1. What is hashing, and how does it relate to the mapping of keys to addresses?
2. Explain the concept of a hash function. What does it do, and why is it important in hashing?
3. What is a collision in the context of hashing, and why does it occur?
4. Can you provide examples of situations where collisions might occur in hash tables?
5. What is linear probing, and how does it work to resolve collisions in a hash table?
6. What is the basic idea behind linear probing when a collision occurs?
7. How do you search for a key in a hash table using linear probing?
8. When inserting a key-value pair with linear probing, what steps do you follow to find the appropriate location in the hash table?
9. What are the advantages and disadvantages of using linear probing as a collision resolution method?
10. Discuss potential scenarios in which linear probing may not be an ideal choice for collision resolution.
11. Can you explain any issues or challenges that may arise when implementing a hash table with linear probing?
12. What are the applications of hashing techniques with linear probing in real-world scenarios?