DATA STRUCTURES WITH C LABORATORY (15CSL38)

No.	Program	Page no.
1	Design, Develop and Implement a menu driven Program in C for the following **Array** operations a. Creating an Array of **N** Integer Elements b. Display of Array Elements with Suitable Headings c. Inserting an Element (**ELEM**) at a given valid Position (**POS**) d. Deleting an Element at a given valid Position(**POS**) e. Exit. Support the program with functions for each of the above operations.	
2	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.	
3	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	
4	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**.**	
5	Design, Develop and Implement 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	
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	
7	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	
8	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	
9	Design, Develop and Implement 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 operations	
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 d. Exit	
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	
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 →L** as H(**K**)=**K** mod **m (remainder** method), and implement hashing technique to map a given key **K** to the address space **L.** Resolve the collision (if any) using **linear probing**.	

1. Design, Develop and Implement a menu driven Program in C for the following **Array** operations
   a. Creating an Array of **N** Integer Elements
   b. Display of Array Elements with Suitable Headings
   c. Inserting an Element (**ELEM**) at a given valid Position (**POS**)
   d. Deleting an Element at a given valid Position(**POS**)
   e. Exit.
   Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5

int a[MAX], pos, elem;
int n = 0;
void create();
void display();
void insert();
void delete();

void main()
{
        int choice;
        while(1)
        {
                printf("\n\n~~~~MENU~~~~");
                printf("\n=>1. Create an array of N integers");
                printf("\n=>2. Display of array elements");
                printf("\n=>3. Insert ELEM at a given POS");
                 printf("\n=>4. Delete an element at a given POS");
                 printf("\n=>5. Exit");
                 printf("\nEnter your choice: ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:         create();
                                        break;
                        case 2:         display();
                                        break;
                        case 3:         insert();
                                        break;
                        case 4:         delete();
                                        break;
                        case 5:         exit(1);
                                        break;
                        default:        printf("\nPlease enter a valid choice:");
                }
        }
}

void create()
{
```

```
        int i;
        printf("\nEnter the number of elements: ");
        scanf("%d", &n);
        printf("\nEnter the elements: ");
        for(i=0; i<n; i++)
        {
                scanf("%d", &a[i]);
        }
}

void display()
{
        int i;
        if(n == 0)
        {
                printf("\nNo elements to display");
                return;
        }
        printf("\nArray elements are: ");
        for(i=0; i<n;i++)
                printf("%d\t ", a[i]);
}

void insert()
{
        int i;

        if(n == MAX)
        {
                printf("\nArray is full. Insertion is not possible");
                return;
        }

        do
        {
                printf("\nEnter a valid position where element to be inserted:  ");
                scanf("%d", &pos);
        }while(pos > n);

        printf("\nEnter the value to be inserted:  ");
        scanf("%d", &elem);

        for(i=n-1; i>=pos ; i--)
        {
                a[i+1] = a[i];
        }
        a[pos] = elem;
        n = n+1;
        display();
}

void delete()
{
```

```
        int i;

        if(n == 0)
        {
                printf("\nArray is empty and no elements to delete");
                return;
        }

        do
        {
                printf("\nEnter a valid position from where element to be deleted: ");
                scanf("%d", &pos);
        }while(pos>=n);

        elem = a[pos];

        printf("\nDeleted element is : %d \n", elem);
        for( i = pos; i< n-1; i++)
        {
                a[i] = a[i+1];
        }
        n = n-1;
        display();
}
```

Output:

```
~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 1
Enter the number of elements: 3
Enter the elements: 10 20 30

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 2
Array elements are: 10   20     30

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 3
```

Enter a valid position where element to be inserted: 5
Enter a valid position where element to be inserted: 4
Enter a valid position where element to be inserted: 3
Enter the value to be inserted: 40
Array elements are: 10 20 30 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **3**
Enter a valid position where element to be inserted: 4
Enter the value to be inserted: 50
Array elements are: 10 20 30 40 50

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **3**
Array is full. Insertion is not possible

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Enter a valid position from where element to be deleted: 5
Enter a valid position from where element to be deleted: 6
Enter a valid position from where element to be deleted: 4
Deleted element is: 50
Array elements are: 10 20 30 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Enter a valid position from where element to be deleted: 2
Deleted element is: 30
Array elements are: 10 20 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
~~~~

=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Enter a valid position from where element to be deleted: 1
Deleted element is: 20
Array elements are: 10 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Enter a valid position from where element to be deleted: 0
Deleted element is: 10
Array elements are: 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Enter a valid position from where element to be deleted: 0
Deleted element is: 40
No elements to display

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **4**
Array is empty and no elements to delete

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: **5**

2 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>
#include<string.h>
#include<stdlib.h>

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

void replacepattern()
{
        int i, j;
        lastr = strlen(rep)-2;
        if(lastp != lastr)
        {
                printf("\nInvalid length of replace string");
                exit(0);
        }
        else
        {
                i = start;
                for(j=0; j<=lastr; j++)
                {
                        str[i] = rep[j];
                        i++;
                }
        }
        return;
}

void findpattern()
{
        int i, j, inmatch;
        lasts = (strlen(str)) - 2;
        lastp = (strlen(pat)) - 2;
        int endmatch;
        for(endmatch = lastp; endmatch<=lasts; endmatch++, start++)
        {
                if(str[endmatch] == pat[lastp])
                {
                        inmatch = start;
                        j=0;
                        while(j<lastp)
                        {
                                if(str[inmatch] == pat[j])
```

```
                        {
                                inmatch++;
                                j++;
                        }
                        else
                        {
                                break;
                        }
                }
                if(j == lastp)
                {
                        patfound = 1;
                        replacepattern();
                }
        }
    }
    return;
}

void main()
{
    printf("\nEnter the main string(STR): ");
    fgets(str, 50, stdin);
    printf("\nEnter the pattern to be matched(PAT): ");
    fgets(pat, 50, stdin);
    printf("\nEnter the string to be replaced(REP): ");
    fgets(rep, 50, stdin);
    printf("\nThe string before pattern match is:\n %s", str);
    findpattern();
    if(patfound == 0)
        printf("\nThe pattern is not found in the main string");
    else
        printf("\nThe string after pattern match and replace is: \n %s ", str);
}
```

Output:

Case 1:

Enter the main string(STR): **Hello hii how are you hii**
Enter the pattern to be matched(PAT): **hii**
Enter the string to be replaced(REP): **xyz**
The string before pattern match is:
Hello hii how are you hii
The string after pattern match and replace is:
Hello xyz how are you xyz

Case 2:

Enter the main string(STR): **Hello hii how are you**
Enter the pattern to be matched(PAT): **abc**
Enter the string to be replaced(REP): **xyz**
The string before pattern match is:
Hello hii how are you
The pattern is not found in the main string

3 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>
#include<stdlib.h>
#define MAX 5

int s[MAX];
int top = -1;

void push(int item);
int pop();
void palindrome();
void display();

void main()
{
        int choice, item;
        while(1)
        {
                printf("\n\n\n\n~~~~~~Menu~~~~~~ : ");
                printf("\n=>1.Push an Element to Stack and Overflow demo ");
                printf("\n=>2.Pop an Element from Stack and Underflow demo");
                printf("\n=>3.Palindrome demo ");
                printf("\n=>4.Display ");
                printf("\n=>5.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:         printf("\nEnter an element to be pushed: ");
                                        scanf("%d", &item);
                                        push(item);
                                        break;
                        case 2:         pop();
                                        break;
                        case 3:         palindrome();
                                        break;
                        case 4:         display();
                                        break;
                        case 5:         exit(1);
                        default:        printf("\nPlease enter valid choice ") ;
                                        break;
                }
        }
}
```

```
void push(int item)
{
        if(top == MAX-1)
        {
                printf("\n~~~~Stack overflow~~~~");
                return;
        }

        top = top + 1 ;
        s[top] = item;
}

void pop()
{
        int item;
        if(top == -1)
        {
                printf("\n~~~~Stack underflow~~~~");
                return -1;
        }
        item = s[top];
        printf("\nElement popped is: %d", item);
        top = top - 1;
}

void display()
{
        int i;
        if(top == -1)
        {
                printf("\n~~~~Stack is empty~~~~");
                return;
        }
        printf("\nStack elements are:\n ");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);
}

void palindrome()
{
        int flag=1,i;
        printf("\nStack content are:\n");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);

        printf("\nReverse of stack content are:\n");
        for(i=0; i<=top; i++)
                printf("| %d |\n", s[i]);

        for(i=0; i<=top/2; i++)
        {
                if( s[i] != s[top-i] )
                {
```

```
                        flag = 0;
                        break;
                }
        }
        if(flag == 1)
                printf("\nIt is palindrome number");
        else
                printf("\nIt is not a palindrome number");
}
```

Output:

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 12

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 13

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 14

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
~~~~~~

=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 15

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 16
~~~~Stack overflow~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **4**

Stack elements are:
| 15 |
| 14 |
| 13 |
| 12 |
| 11 |

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 15

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **4**
Stack elements are:
| 14 |
| 13 |
| 12 |
| 11 |
~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
~~~~~~

=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 14
~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 13

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 12

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
~~~~Stack underflow~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **4**
~~~~Stack is empty~~~~

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo

=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **3**
Stack content are:
| 11 |
| 22 |
| 11 |

Reverse of stack content are:
| 11 |
| 22 |
| 11 |

It is palindrome number

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **2**
Element popped is: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 11

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 22

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **1**
Enter an element to be pushed: 33

~~~~~~Menu~~~~~
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: **3**
Stack content are:

| 33 |
| 22 |
| 11 |

Reverse of stack content are:
| 11 |
| 22 |
| 33 |

It is not a palindrome number

4 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>
#include<stdlib.h>

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

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

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

void evaluate()
{
        int i = 0, j = 0;
        char symb, temp;
        push('#');
        for(i=0; infix[i] != '\0'; i++)
        {
                symb = infix[i];
                switch(symb)
                {
                        case '(' :      push(symb);
                                        break;

                        case ')' :      temp = pop();
                                        while(temp != '(' )
                                        {
                                                postfix[j] = temp;
                                                j++;
                                                temp = pop();
                                        }
                                        break;
                        case '+' :
                        case '-' :
                        case '*' :
                        case '/' :
                        case '%' :
                        case '^' :
                        case '$' :      while( prec(stack[top]) >= prec(symb) )
```

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

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

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

int prec(char symb)
{
        int p;
        switch(symb)
        {
                case '#' :      p = -1;
                                break;
                case '(' :
                case ')' :      p = 0;
                                break;
                case '+' :
                case '-' :      p = 1;
                                break;
                case '*' :
                case '/' :
                case '%' :      p = 2;
                                break;
                case '^' :
                case '$' :      p = 3;
```

```
                    break;
        }
        return p;
}
```

Output:

Enter the valid infix expression: **(a+b)+c/d*e**

The entered infix expression is :
(a+b)+c/d*e

The corresponding postfix expression is :
ab+cd/e*+

5 Design, Develop and Implement 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

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

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

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

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

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

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

Output:

To compile in Linux: cc 5.c -lm

Enter a valid postfix expression:
623+-382/+*2$3+
Result = 52

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

Tower of Hanoi

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

void tower(int n, char from_peg, char aux_peg, char to_peg);
void main()
{
        int n;
        printf("\nEnter the number of disks: ");
        scanf("%d", &n);

        tower(n, 'A', 'B', 'C');                    // A -> from_peg  B -> aux_peg  C -> to_peg
        printf("\nTotal number of moves = %.0lf", pow(2,n)-1 );
}

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

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

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

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

Output:

```
Enter the number of disks: 3

Move disk 1 from A peg to C peg
Move disk 2 from peg A to B peg
Move disk 1 from C peg to B peg
Move disk 3 from peg A to C peg
Move disk 1 from B peg to A peg
Move disk 2 from peg B to C peg
Move disk 1 from A peg to C peg
```

Total number of moves = 7

6 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>
#include<stdio_ext.h>

#define MAX 3
char cq[MAX];
int front = -1,rear = -1;

void inset(char);
void delete();
void display();

int main()
{
        int ch;
        char item;

        while(1)
        {
                printf("\n\n~~Main Menu~~");
                printf("\n==> 1. Insertion and Overflow Demo");
                printf("\n==> 2. Deletion and Underflow Demo");
                printf("\n==> 3. Display");
                printf("\n==> 4. Exit");
                printf("\nEnter Your Choice: ");
                scanf("%d", &ch);
                __fpurge(stdin);
                switch(ch)
                {
                        case 1:         printf("\n\nEnter the element to be inserted: ");
                                        scanf("%c", &item);
                                        insert(item);
                                        break;
                        case 2:         delete();
                                        break;
                        case 3:         display();
                                         break;
                        case 4: exit(0);
                        default: printf("\n\nPlease enter a valid choice");
                }
        }
}
```

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

void delete()
{
        char item;
        if(front == -1)
        {
                printf("\n\n~~Circular Queue Underflow~~");
        }
        else
        {
                item = cq[front];
                if(front == rear) //only one element
                        front = rear = -1;
                else
                        front = (front+1)%MAX;
                printf("\n\nDeleted element from the queue is: %c ", item );
        }
}

void display()
{
        int i;
        if(front ==-1)
        {
                printf("\n\nCircular Queue Empty");
                return;
        }
        else
        {
                printf("\nCircular Queue contents are:\n");
                printf("\nFront[%d]-> ", front);
                for(i=front; i!=rear ; i=(i+1)%MAX)
                {
                        printf("  %c", cq[i]);
                }
```

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

Output:

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

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

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

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

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

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

==> 4. Exit
Enter Your Choice: **2**
Deleted element from the queue is: A

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

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

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

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

7 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<string.h>
#include<stdlib.h>

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

NODE first = NULL;
int count = 0;

NODE createStudentNode()
{
        char us[20], nam[20], bran[20];
        int se;
        long int pn;
        printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student: \n");
        scanf("%s %s %s %d %ld", us, nam, bran, &se, &pn);
        NODE studentNode;
        studentNode = (NODE)malloc(sizeof(struct node));
        if(studentNode == NULL)
        {
                printf("\nMemory is not available");
                exit(0);
        }
        studentNode->link = NULL;
        strcpy(studentNode->usn, us);
        strcpy(studentNode->name, nam);
        strcpy(studentNode->branch, bran);
        studentNode->sem = se;
        studentNode->phoneNo = pn;
        count++;
        return studentNode;
}

NODE insertAtFront()
{
        NODE temp;
        temp = createStudentNode();
```

```
        if(first  == NULL)
        {
                return  temp;
        }
        temp->link  = first;
        return  temp;
}

NODE deleteAtFront()
{
        NODE temp;
        if(first  == NULL)
        {
                printf("\nLinked  list  is empty");
                return  NULL;
        }
        if(first->link  == NULL)
        {
                printf("\nThe  Student  node with usn:%s  is deleted ", first->usn);
                count--;
                free(first);
                return  NULL;
        }
        temp  = first;
        first  = first->link;
        printf("\nThe  Student  node with usn:%s  is deleted",temp->usn);
        count--;
        free(temp);
        return  first;
}

NODE insertAtEnd()
{
        NODE cur,  temp;
        temp  = createStudentNode();
        if(first  == NULL)
        {
                return  temp;
        }

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

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

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

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

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

void displayStatus()
{
        NODE cur;
        int nodeNo = 1;
        cur = first;
        printf("\nThe contents of SLL: \n");
        if(cur == NULL)
                printf("\nNo Contents to display in SLL \n");
        while(cur!=NULL)
        {
                printf("\n||%d||", nodeNo);
                printf(" USN:%s|", cur->usn);
                printf(" Name:%s|", cur->name);
                printf(" Branch:%s|", cur->branch);
                printf(" Sem:%d|", cur->sem);
                printf(" Ph:%ld|", cur->phoneNo);
                cur = cur->link;
                nodeNo++;
        }
        printf("\n No of student nodes is %d \n",count);
```

```
}
void stackDemoUsingSLL()
{
        int ch;
        while(1)
        {
                printf("\n~~~Stack Demo using SLL~~~\n");
                printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
                printf("\nEnter your choice for stack demo");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:         first = insertAtFront();
                                        break;
                        case 2:         first = deleteAtFront();
                                        break;
                        case 3:         displayStatus();  break;
                        default:        return;
                }
        }
}
void main()
{
        int ch, i, n;
        while(1)
        {
                printf("\n~~~Menu~~~");
                printf("\nEnter your choice for SLL operation \n");
                printf("\n1:Create SLL of Student Nodes");
                printf("\n2:DisplayStatus");
                printf("\n3:InsertAtEnd");
                printf("\n4:DeleteAtEnd");
                printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
                printf("\n6:Exit \n");
                printf("\nEnter your choice:");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1 :        printf("\nEnter the no of students:    ");
                                        scanf("%d", &n);
                                        for(i=1; i<=n; i++)
                                                first = insertAtFront();
                                        break;
                        case 2:         displayStatus();
                                        break;
                        case 3:         first = insertAtEnd();
                                        break;
                        case 4:         first = deleteAtEnd();
                                        break;
                        case 5:         stackDemoUsingSLL();
                                        break;
                        case 6:         exit(0);
                        default:        printf("\nPlease enter the valid choice");
```

```
                }
        }
}
```

Output:

```
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:1
Enter the no of students:  3
Enter the usn, Name, Branch, sem, PhoneNo of the student:
111
aaa
cs
1
111111
Enter the usn, Name, Branch, sem, PhoneNo of the student:
222
bbb
ec
2
222222
Enter the usn, Name, Branch, sem, PhoneNo of the student:
333
ccc
ec
3
333333

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:2
The contents of SLL:
||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
 No of student nodes is 3

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
```

4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**3**

Enter the usn,Name,Branch, sem,PhoneNo of the student:
444
ddd
ec
4
444444

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**2**
The contents of SLL:
||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
||4|| USN:444| Name:ddd| Branch:ec| Sem:4| Ph:444444|
No of student nodes is 4

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**4**
The student node with the usn:444 is deleted

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**2**
The contents of SLL:
||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
No of student nodes is 3
~~~

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:**4**
The student node with the usn:111 is deleted

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

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **1**
Enter the usn,Name,Branch, sem,PhoneNo of the student:
555
eee
cs
1
555555

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **3**
The contents of SLL:
||1|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||2|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||3|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
No of student nodes is 3

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **1**
~~~

Enter the usn,Name,Branch, sem,PhoneNo of the student:
666
fff
cs
6
666666

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **3**
The contents of SLL:
||1|| USN:666| Name:fff| Branch:cs| Sem:6| Ph:666666|
||2|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||3|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||4|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
No of student nodes is 4

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **2**
The Student node with usn:666 is deleted

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: **3**
The contents of SLL:
||1|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||2|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||3|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
No of student nodes is 3

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo**4**

8 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.** f. Exit

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

struct node
{
        char ssn[25], name[25], dept[10], designation[25];
        int sal;
        long int phoneno;
        struct node *llink;
        struct node *rlink;
};
typedef struct node* NODE;
NODE first = NULL;
int count = 0;

NODE createEmployeeNode()
{
        char ssn[25], name[25], dept[10], designation[25];
        int sal;
        long int phoneno;
        printf("\nEnter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee: \n");
        scanf("%s %s %s %s %d %ld",ssn,name,dept,designation,&sal,&phoneno);
        NODE employeeNode;
        employeeNode = (NODE)malloc(sizeof(struct node));
        if( employeeNode== NULL)
        {
                printf("\nRunning out of memory");
                exit(0);
        }
        employeeNode->llink = NULL;
        employeeNode->rlink = NULL;
        strcpy(employeeNode->ssn, ssn);
        strcpy(employeeNode->name, name);
        strcpy(employeeNode->dept, dept);
        strcpy(employeeNode->designation, designation);
        employeeNode->sal = sal;
        employeeNode->phoneno = phoneno;
        count++;
        return employeeNode;
}
NODE insertAtFront()
{
```

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

NODE insertAtEnd()
{
        NODE cur, temp;
        temp = createEmployeeNode();
        if(first == NULL)
        {
                return temp;
        }
        cur = first;
        while(cur->rlink!=NULL)
        {
                cur = cur->rlink;
        }
        cur->rlink = temp;
        temp->llink = cur;
        return first;
}
NODE deleteAtEnd()
```

```
{
        NODE prev, cur;
        if(first == NULL)
        {
                printf("\nDoubly Linked List is empty");
                return NULL;
        }
        if(first->rlink == NULL)
        {
                printf("\nThe employee node with the ssn:%s is deleted", first->ssn);
                free(first);
                count--;
                return NULL;
        }
        prev = NULL;
        cur = first;
        while(cur->rlink!=NULL)
        {
                prev = cur;
        cur = cur->rlink;
        }
        cur->llink = NULL;
        printf("\nThe employee node with the ssn:%s is deleted", cur->ssn);
        free(cur);
        prev->rlink = NULL;
        count--;
        return first;
}
void displayStatus()
{
        NODE cur;
        int nodeno=1;
        cur = first;

        if(cur == NULL)
                printf("\nNo Contents to display in DLL");
        while(cur!=NULL)
        {
                printf("\nENode:%d|| SSN:%s| Name:%s| Department:%s| Designation:%s|
                        Salary:%d| Phone no:%ld", nodeno, cur->ssn, cur->name, cur->dept,
                        cur->designation, cur->sal, cur->phoneno);
                cur = cur->rlink;
                nodeno++;
        }
        printf("\nNo of employee nodes is %d",count);
}

void doubleEndedQueueDemo()
{
        int ch;
        while(1)
        {
                printf("\nDemo Double Ended Queue Operation");
```

```
                printf("\n1:InsertQueueFront \n2: DeleteQueueFront \n3:InsertQueueRear
                        \n4:DeleteQueueRear \n5:DisplayStatus \n6: Exit \n");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:         first = insertAtFront();
                                        break;
                        case 2:         first = deleteAtFront();
                                        break;
                        case 3:         first = insertAtEnd();
                                        break;
                        case 4:         first = deleteAtEnd();
                                        break;
                        case 5:         displayStatus();
                                        break;
                        default : return;
                 }
        }
 }

void main()
{
        int ch, i, n;
        while(1)
        {
                printf("\n\n~~~Menu~~~");
                printf("\n1:Create DLL of Employee Nodes");
                printf("\n2:DisplayStatus");
                printf("\n3:InsertAtEnd");
                printf("\n4:DeleteAtEnd");
                printf("\n5:InsertAtFront");
                printf("\n6:DeleteAtFront");
                printf("\n7:Double Ended Queue Demo using DLL");
                printf("\n8:Exit \n");
                printf("\nPlease enter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1 :        printf("\nEnter the no of Employees:  ");
                                        scanf("%d", &n);
                                        for(i=1;i<=n;i++)
                                                first = insertAtEnd();
                                        break;
                        case 2:         displayStatus();
                                        break;
                        case 3:         first = insertAtEnd();
                                        break;
                        case 4:         first = deleteAtEnd();
                                        break;
                        case 5:         first = insertAtFront();
                                        break;
                        case 6:         first = deleteAtFront();
                                        break;
```

```
                    case 7:     doubleEndedQueueDemo();
                                break;
                    case 8 :    exit(0);
                  default: printf("\nPlease Enter the valid choice");
            }
        }
}
```

Output:

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **1**
Enter the no of Employees: 2
Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
111
aaa
dept1
des1
1000
11111
Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
222
bbb
dept2
des2
2000
22222

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **2**
ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone no:22222
No of employee nodes is 2

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
~~~

4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **3**
Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
333
ccc
dept3
des3
3000
33333

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **2**
ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone no:22222
ENode:3||SSN:333|Name:ccc|Department:dept3|Designation:des3|Salary:3000|Phone no:33333
No of employee nodes is 3

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **5**
Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
444
ddd
dept4
des4
4000
44444

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
~~~

6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **2**
ENode:1||SSN:444|Name:ddd|Department:dept4|Designation:des4|Salary:4000|Phone no:44444
ENode:2||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone no:11111
ENode:3||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone no:22222
ENode:4||SSN:333|Name:ccc|Department:dept3|Designation:des3|Salary:3000|Phone no:33333
No of employee nodes is 4

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **4**
The employee node with the ssn:333 is deleted

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **6**
The employee node with the ssn:444 is deleted

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **2**
ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone no:22222
No of employee nodes is 2

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd

5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: **7**
Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
2
The employee node with the ssn:111 is deleted

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
4
The employee node with the ssn:222 is deleted

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
2
Doubly Linked List is empty

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
6

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
~~~

9 Design, Develop and Implement 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 operations

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

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

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

NODE attach(int coef, int xexp, int yexp, int zexp, NODE head)
{
        NODE temp, cur;
        temp = getnode();
        temp->coef = coef;
        temp->xexp = xexp;
        temp->yexp = yexp;
        temp->zexp = zexp;
        cur = head->link;
        while(cur->link != head)
        {
                cur = cur->link;
        }
        cur->link = temp;
        temp->link = head;
        return head;
}

NODE read_poly(NODE head)
{
        int i, j, coef, xexp, yexp, zexp, n;
```

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

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

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

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

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

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

NODE poly_sum(NODE head1, NODE head2, NODE head3)
```

```
{
   NODE a, b;
   int coef;
   a = head1->link;
   b = head2->link;

   while(a!=head1  && b!=head2)
   {
        while(1)
        {
             if(a->xexp  == b->xexp && a->yexp == b->yexp && a->zexp == b->zexp)
             {
                  coef = a->coef + b->coef;
                  head3 = attach(coef,  a->xexp, a->yexp, a->zexp, head3);
                  a = a->link;
                  b = b->link;
                  break;
             } //if ends here
             if(a->xexp!=0  || b->xexp!=0)
             {
                  switch(COMPARE(a->xexp,  b->xexp))
                  {
                   case -1 :  head3 = attach(b->coef,  b->xexp, b->yexp, b->zexp, head3);
                              b = b->link;
                              break;

                   case 0 :    if(a->yexp  > b->yexp)
                               {
                                    head3 = attach(a->coef,  a->xexp, a->yexp, a->zexp, head3);
                                    a = a->link;
                                    break;
                               }
                               else if(a->yexp  < b->yexp)
                               {
                                    head3 = attach(b->coef,  b->xexp, b->yexp, b->zexp, head3);
                                    b = b->link;
                                    break;
                               }
                               else if(a->zexp  > b->zexp)
                               {
                                    head3 = attach(a->coef,  a->xexp, a->yexp, a->zexp, head3);
                                    a = a->link;
                                    break;
                               }
                               else if(a->zexp  < b->zexp)
                               {
                                    head3 = attach(b->coef,  b->xexp, b->yexp, b->zexp, head3);
                                    b = b->link;
                                    break;
                               }
                   case 1 :    head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
                               a = a->link;
                               break;
```

```
                } //switch ends here
                break;
        } //if ends here
        if(a->yexp!=0  || b->yexp!=0)
        {
                switch(COMPARE(a->yexp,  b->yexp))
                {
                    case -1 :  head3 = attach(b->coef,  b->xexp, b->yexp, b->zexp, head3);
                               b = b->link;
                               break;
                     case 0 :  if(a->zexp  > b->zexp)
                               {
                                   head3 = attach(a->coef,  a->xexp, a->yexp, a->zexp, head3);
                                   a = a->link;
                                   break;
                                }
                               else if(a->zexp  < b->zexp)
                               {
                                   head3 = attach(b->coef,  b->xexp, b->yexp, b->zexp, head3);
                                   b = b->link;
                                   break;
                               }
                      case 1 :  head3 = attach(a->coef,  a->xexp, a->yexp, a->zexp, head3);
                                a = a->link;
                                break;
                }
                break;
        }
        if(a->zexp!=0  || b->zexp!=0)
        {
                switch(COMPARE(a->zexp,b->zexp))
                {
                        case -1 :  head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
                                   b = b->link;
                                   break;
                        case 1 :   head3 = attach(a->coef,  a->xexp, a->yexp, a->zexp, head3);
                                   a = a->link;
                                   break;
                }
                break;
        }
    }
}
while(a!=  head1)
{
     head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
    a = a->link;
}
while(b!=  head2)
{
    head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
    b = b->link;
}
```

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

 head->link=head;
 head1->link=head1;
 head2->link=head2;
 head3->link=  head3;

 while(1)
 {
     printf("\n~~~Menu~~~");
     printf("\n1.Represent  and Evaluate  a Polynomial  P(x,y,z)");
     printf("\n2.Find  the sum of two polynomials  POLY1(x,y,z)");
     printf("\nEnter  your choice:");
     scanf("%d",&ch);
     switch(ch)
     {
        case 1:          printf("\n~~~~Polynomial  evaluation  P(x,y,z)~~~\n");
                         head = read_poly(head);
                         printf("\nRepresentation  of Polynomial  for evaluation:  \n");
                         display(head);
                         res = poly_evaluate(head);
                         printf("\nResult  of polynomial  evaluation  is : %d \n", res);
                         break;

        case 2:          printf("\nEnter  the POLY1(x,y,z):  \n");
                         head1 = read_poly(head1);
                         printf("\nPolynomial  1 is:  \n");
                         display(head1);

                         printf("\nEnter  the POLY2(x,y,z):  \n");
                         head2 = read_poly(head2);
                         printf("\nPolynomial  2 is: \n");
                         display(head2);

                         printf("\nPolynomial  addition  result: \n");
                         head3 = poly_sum(head1,head2,head3);
                         display(head3);
                         break;
           case 3:       exit(0);
     }
   }
}
```

Output:

~~~Menu~~~

1.Represent and Evaluate a Polynomial P(x,y,z)

2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)

Enter your choice: **1**

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

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

Enter the 1 term:

Coef = **6**

Enter Pow(x) Pow(y) and Pow(z): **2 2 1**

Enter the 2 term:

Coef = **-4**

Enter Pow(x) Pow(y) and Pow(z): **0 1 5**

Enter the 3 term:

Coef = **3**

Enter Pow(x) Pow(y) and Pow(z): **3 1 1**

Enter the 4 term:

Coef = **2**

Enter Pow(x) Pow(y) and Pow(z): **1 5 1**

Enter the 5 term:

Coef = **-2**

Enter Pow(x) Pow(y) and Pow(z): **1 1 3**

Representation of Polynomial for evaluation:

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

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

Result of polynomial evaluation is : **5**

~~~Menu~~~

1.Represent and Evaluate a Polynomial P(x,y,z)

2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)

Enter your choice: **2**

Enter the POLY1(x,y,z):

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

Enter the 1 term:

Coef = **6**

Enter Pow(x) Pow(y) and Pow(z): **4 4 4**

Enter the 2 term:

Coef = **3**

Enter Pow(x) Pow(y) and Pow(z): **4 3 1**

Enter the 3 term:

Coef = **5**

Enter Pow(x) Pow(y) and Pow(z): **0 1 1**

Enter the 4 term:

Coef = **10**

Enter Pow(x) Pow(y) and Pow(z): **0 1 0**

Enter the 5 term:

Coef = **5**

Enter Pow(x) Pow(y) and Pow(z): **0 0 0**

Polynomial 1 is:

6x^4y^4z^4 + 3x^4y^3z^1 + 5x^0y^1z^1 + 10x^0y^1z^0 + 5x^0y^0z^0

Enter the POLY2(x,y,z):
Enter the no of terms in the polynomial: **5**

Enter the 1 term:
Coef = **8**
Enter Pow(x) Pow(y) and Pow(z): **4 4 4**

Enter the 2 term:
Coef = **4**
Enter Pow(x) Pow(y) and Pow(z): **4 2 1**

Enter the 3 term:
Coef = **30**
Enter Pow(x) Pow(y) and Pow(z): **0 1 0**

Enter the 4 term:
Coef = **20**
Enter Pow(x) Pow(y) and Pow(z): **0 0 1**

Enter the 5 term:
Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **0 0 0**

Polynomial 2 is:
8x^4y^4z^4 + 4x^4y^2z^1 + 30x^0y^1z^0 + 20x^0y^0z^1 + 3x^0y^0z^0

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

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

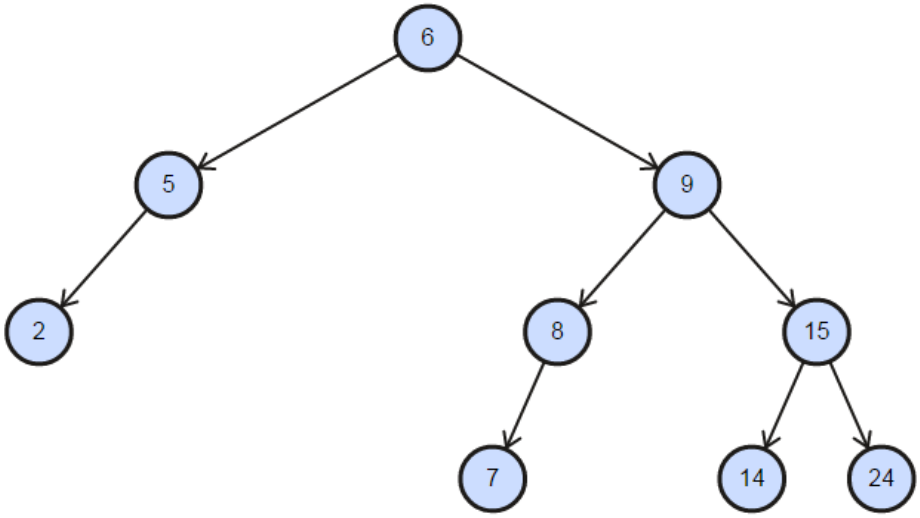
10 Design, Develop and Implement a menu driven Program in C for the following operations on **Binary Search Tree (BST)** of Integers

- **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
- **d.** Exit

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

struct BST
{
        int data;
        struct BST *lchild;
        struct BST *rchild;
};
typedef struct BST * NODE;

NODE get_node()
{
        NODE temp;
        temp = (NODE) malloc(sizeof(struct BST));
        temp->lchild = NULL;
        temp->rchild = NULL;
        return temp;
}

void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root, int key);

void insert(NODE root, NODE newnode)
{
  /*Note: if newnode->data == root->data it will be skipped. No duplicate nodes are allowed */

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

```
void search(NODE root, int key)
{
        NODE cur;
        if(root == NULL)
        {
                printf("\nBST is empty.");
                return;
        }
        cur = root;
        while (cur != NULL)
        {
                if (cur->data == key)
                {
                        printf("\nKey element %d is present in BST", cur->data);
                        return;
                }
                if (key < cur->data)
                        cur = cur->lchild;
                else
                        cur = cur->rchild;
        }
        printf("\nKey element %d is not found in the BST", key);
}

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

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

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

```
}

void main()
{
        int ch, key, val, i, n;
        NODE root = NULL, newnode;
        while(1)
        {
                printf("\n~~~~BST MENU~~~~");
                printf("\n1.Create a BST");
                printf("\n2.Search");
                printf("\n3.BST Traversals: ");
                printf("\n4.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:         printf("\nEnter the number of elements: ");
                                        scanf("%d", &n);
                                        for(i=1;i<=n;i++)
                                        {
                                                newnode = get_node();
                                                printf("\nEnter The value: ");
                                                scanf("%d", &val);
                                                newnode->data = val;
                                                if (root == NULL)
                                                        root = newnode;
                                                else
                                                        insert(root, newnode);
                                        }
                                        break;
                         case 2:        if (root == NULL)
                                                printf("\nTree Is Not Created");
                                        else
                                        {
                                                printf("\nThe Preorder display : ");
                                                preorder(root);
                                                printf("\nThe Inorder display : ");
                                                inorder(root);
                                                printf("\nThe Postorder display : ");
                                                postorder(root);
                                        }
                                        break;
                         case 3:        printf("\nEnter Element to be searched: ");
                                        scanf("%d", &key);
                                        search(root, key);
                                        break;

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



Output:

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: **1**
Enter the number of elements: **12**
Enter The value: **6**
Enter The value: **9**
Enter The value: **5**
Enter The value: **2**
Enter The value: **8**
Enter The value: **15**
Enter The value: **24**
Enter The value: **14**
Enter The value: **7**
Enter The value: **8**
Enter The value: **5**
Enter The value: **2**

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

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

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2
Enter Element to be searched: 66
Key element 66 is not found in the BST

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2
Enter Element to be searched: 14
Key element 14 is present in BST

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

11 Design, Develop and Implement a Program in C for the following operations on **Graph(G)** of Cities

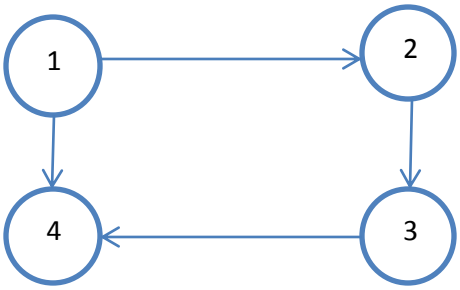
a. Create a Graph of **N** cities using Adjacency Matrix.

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

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

void bfs(int v);
void dfs(int v);

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

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

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

void dfs(int v)
{
        int i;
        visited[v]=1;
        s[++top] = v;
```

```
        for(i=1;i<=n;i++)
        {
                if(a[v][i]  == 1&& visited[i]  == 0 )
                {
                        dfs(i);
                        count++;
                }
        }
}

int main()
{
        int ch, start, i, j;
        creategraph();
        printf("\n\n~~~Menu~~~~");
        printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
        printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
        printf("\n==>3:Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
                case 1:         for(i=1;i<=n;i++)
                                        visited[i] = 0;
                                printf("\nEnter the starting vertex: ");
                                scanf("%d", &start);
                                bfs(start);
                                for(i=1;i<=n;i++)
                                {
                                        if(visited[i] == 0)
                                                printf("\nThe vertex that is not reachable is %d", i);
                                }
                                break;
                case 2:         for(i=1;i<=n;i++)
                                        visited[i] = 0;
                                printf("\n Enter the starting vertex:\t");
                                scanf("%d", &start);
                                dfs(start);
                                printf("\nNodes reachable from starting vertex %d are:\n", start);
                                for(i=1; i<=count; i++)
                                        printf("%d\t", s[i]);
                                break;
                case 3: exit(0);
                default: printf("\nPlease enter valid choice:");
        }
}
```

Output:



Case 1:

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

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

~~~Menu~~~~
==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
Enter your choice: **1**
Enter the starting vertex: 1
Nodes reachable from starting vertex 1 are: 2 4 3

Case 2:

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

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

~~~Menu~~~~
==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
Enter your choice: **1**
Enter the starting vertex: 2
Nodes reachable from starting vertex 2 are: 3 4
The vertex that is not reachable is 1

Case 3:

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

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

~~~Menu~~~~
~~~

==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
Enter your choice: **2**
Enter the starting vertex: 1
Nodes reachable from starting vertex 1 are: 2 3 4

Case 4:
Enter the number of vertices in graph: **4**
Enter the adjacency matrix:
0 1 0 1
0 0 1 0
0 0 0 1
0 0 0 0
~~~Menu~~~~
==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
Enter your choice: **2**
Enter the starting vertex: 2
Nodes reachable from starting vertex 2 are: 3 4
~~~

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

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

int key[20], n, m;
int *ht, hashindex;
int elecount = 0;

void createHashTable()
{
        int i;
        ht = (int*)malloc(m*sizeof(int));
        if(ht == NULL)
                printf("\nUnable to create the hash table");
        else
                for(i=0; i<m; i++)
                        ht[i] = -1;
 }

void insertIntoHashTable(int key)
{
        hashindex = key % m;
        while(ht[hashindex] != -1)
        {
                hashindex = (hashindex+1)%m;
        }
        ht[hashindex] = key;
        elecount++;
 }

void displayHashTable()
{
        int i;
        if(elecount == 0)
        {
                printf("\nHash Table is empty");
                return;
        }
        printf("\nHash Table contents are:\n\n ");
        for(i=0; i<m; i++)
                printf("\nT[%d] --> %d ", i, ht[i]);
}

void main()
{
        int i;
```

```
        printf("\nEnter the number of employee records (N) : ");
        scanf("%d", &n);
        printf("\nEnter the four digit key values (K) of 'N' Employee Records:\n ");
        for(i=0; i<n; i++)
        scanf("%d", &key[i]);
        printf("\nEnter the two digit memory locations (m) for hash table: ");
        scanf("%d", &m);
        createHashTable();
        printf("\nInserting key values of Employee records into hash table….. ");
        for(i=0; i<n; i++)
        {
                if(elecount == m)
                {
                        printf("\nHash table is full. Cannot insert the %d record key value", i+1);
                        break;
                }
                insertIntoHashTable(key[i]);
        }
        displayHashTable();
}
```

Output:

```
Enter the number of employee records (N) : 12
Enter the four digit key values (K) of 'N' Employee Records:
1234
5678
3456
2345
6799
1235
7890
3214
3456
1235
5679
2346
Enter the two digit memory locations (m) for hash table: 15
Inserting key values of Employee records into hash table
Hash Table contents are:
 T[0] --> 7890
 T[1] --> -1
 T[2] --> -1
 T[3] --> -1
 T[4] --> 1234
 T[5] --> 2345
 T[6] --> 3456
 T[7] --> 6799
 T[8] --> 5678
 T[9] --> 1235
 T[10] --> 3214
 T[11] --> 3456
 T[12] --> 1235
 T[13] --> 5679
 T[14] --> 2346
```