EXPERIMENT DETAILS

1. Design and Implement a program in C on Frequency Histogram, that builds a frequency array for data values in the range 1 to n and then prints their histogram. The program should,

a. Read, Store and Print the data in an array.

b. Analyze the data in the array, one element at a time. Add 1 to the corresponding element in a frequency array based on the data value.

c. Print a histogram using asterisks for each occurrence of an element.

……

PROGRAM:

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

int a[100],i,j,n;
int frequency[10]={0};
void getdata();
void printdata();
void makefrequency();
void makeHistogram();

void main()
{
   getdata();
   printdata();
   makefrequency();
   makeHistogram();
}
void getdata()
{
   printf("enter the how many number of array elements\n");
   scanf("%d",&n);
   if(n ==0)
   {
      printf( "Array is Empty\n");
      exit(0);
   }
   printf("Enter %d elements between the range 0-9\n",n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
```

```
    }
}
void printdata()
{
    printf("The array elements are \n");
    for(int i = 0; i < n; i++)
        printf("%d\t",a[i]);
}
void makefrequency()
{
    for(i=0;i<10;i++)
                for(j=0;j<n;j++)
                        if(a[j]==i)
                                frequency[i]++;
}
void  makeHistogram()
{
    printf("\nFrequency histogram is:\n");
    printf("Histo---Frequency\t");
        for(i=0;i<10;i++)
        {
                printf("\n[%d]",i);
                for(j=0;j<frequency[i];j++)
                        printf("\t*");
        }
        printf("\n");
}
```

STEPS TO EXECUTE

```
1 gedit freqencyhisto.c
2 gcc freqencyhisto.c
3 ./a.out
```

OUTPUT:

```
enter the how number of array elements
20
Enter 20 elements between the range 0-9
0 1 2 1 0 3 4 5 3 4 7 8 2 6 3 6 9 8 4 1
The array elements are
0       1       2       1       0       3       4       5       3       4       7       8       2
6       3       6       9       8       4       1
```

```
Frequency histogram is:
Histo---Frequency
[0]    *      *
[1]    *      *      *
[2]    *      *
[3]    *      *      *
[4]    *      *      *
[5]    *
[6]    *      *
[7]    *
[8]    *      *
[9]    *
```

2. Design and Implement a program in C for the following Stack Applications,
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

………

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
#include<math.h>
int stack[50], top=-1;
void push(int elem)
{
        stack[++top]=elem;
}
void main()
{
        char postfix[50],ch;
        int i=0,op1,op2;
        printf("Enter a Suffix expression with single digit operands and operators:");
        scanf("%s",postfix);
        while((ch=postfix[i++])!='\0')
        {
                if(isalpha(ch))
                {
                        printf("Invalid expression\n");
                        return;
                }
                else if(isdigit(ch))
                        push(ch-48);
```

```
				else
				{
						op2=stack[top--];
						if(top<=-1)
						{
								printf("Invalid Expression\n");
								return;
						}
						op1=stack[top--];
						switch(ch)
						{
								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 '^': push(pow(op1,op2));
										break;
								default:  printf("Invalid operator\n");
										return;
						}
				}
		}
		if(top!=0)
				printf("invalid expression\n");
		else
				printf("Result = %d\n",stack[top]);
}
```

<u>OUTPUT</u>

```
Enter a Suffix expression with single digit operands and operators:123+*
Result = 5
.......................................................................................................
Enter a Suffix expression with single digit operands and operators:22^32*%
Result = 4
.......................................................................................................
```

```
Enter a Suffix expression with single digit operands and operators:45+3#
Invalid operator
```

……………………………………………………………………………………………………

```
Enter a Suffix expression with single digit operands and operators:1+a
Invalid Expression
```

……………………………………………………………………………………………………

```
Enter a Suffix expression with single digit operands and operators:+12
Invalid Expression
```

……………………………………………………………………………………………………

```
Enter a Suffix expression with single digit operands and operators:1+2
Invalid Expression
```

……………………………………………………………………………………………………

```
Enter a Suffix expression with single digit operands and operators:123+
Invalid expression
```

……………………………………………………………………………………………………

```
Enter a Suffix expression with single digit operands and operators:45*+3
Invalid Expression
```

…………………………………………………………………………………………………….

```
Enter a Suffix expression with single digit operands and operators:ab/
Invalid expression
```

……………………………………………………………………………………………………

b. Conversion of Arithmetic Expressions

……………………………………………………………………………………………………

<u>PROGRAM:</u>

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#define SIZE 20
char stack[SIZE];
int top = -1;
void push(char elem)
{
	stack[++top] = elem;
}
char pop()
```

```
{
        return (stack[top--]);
}
int precedence(char elem) /* Decides the precedence */
{
        switch (elem)
        {
                case'#': return 0;
                case'(': return 1;
                case'+':
                case'-': return 2;
                case'*':
                case'/':
                case'%': return 3;
                case'^': return 4;
                default: printf("Not a Valid Expression\n");
                         exit(0);
        }
}
void main()
{
        char infix[20], postfix[20], ch, elem;
        int i = 0, k = 0, pr;
        printf("Enter the Infix Expression: ");
        scanf("%s", infix);
        push('#');   /* Initial element of stack. It is a handler */
        while ((ch = infix[i++]) != '\0')
        {
                if (ch == '(')          /* Verifying left parenthesis */
                {
                        push(ch);
                }
                else if (isalnum(ch))   /* Verifying operand */
                {
                        postfix[k++] = ch;
                }
                else if (ch == ')')     /* Verifying right parenthesis */
                {
                        while (stack[top] != '(')
                        {
                                postfix[k++] = pop();
                                if(stack[top] == '#')
                                {
                                        printf("Not a Valid Expression\n");
                                        exit(0);
                                }
                        }
                        elem = pop();  /* Removing left parenthesis */
                }
                else                    /* Verifying operators */
                {
                        pr=precedence(ch);
                        if(ch=='^')
                        {
                                pr++;  /* If ^ operator appears more than once evaluation takes
            place from right to left */
                        }
                        while (precedence(stack[top]) >= pr)
                        {
                                postfix[k++] = pop();
                        }
                        push(ch);         /* Push the operator to stack */
                }
        }
        while (stack[top] != '#') /* Pop from stack till empty */
        {
                postfix[k++] = pop();
             }
        postfix[k] = '\0'; /* Make postfix as valid string */
        printf("Given Infix Expression: %s\nPostfix Expression: %s\n", infix, postfix);
}
```

<u>OUTPUT:</u>

```
Enter the Infix Expression: (1+2)*(4-5)
Given Infix Expression: (1+2)*(4-5)
Postfix Expression: 12+45-*
.........................................................................................................
Enter the Infix Expression: (a+b)*c/d^e%f
Given Infix Expression: (a+b)*c/d^e%f
Postfix Expression: ab+c*de^/f%
.........................................................................................................
Enter the Infix Expression: 1^2^3
```

```
Given Infix Expression: 1^2^3
Postfix Expression: 123^^
..............................................................................................................
Enter the Infix Expression: 1:2
Not a Valid Expression
..............................................................................................................
Enter the Infix Expression: sum$+add
Not a Valid Expression
..............................................................................................................
Enter the Infix Expression: sum+123
Given Infix Expression: sum+123
Postfix Expression: sum123+
..............................................................................................................
```

3. Design and Implement a program in C for the following operations on QUEUES,

a. Categorize the numbers (Range 1 to 100) without losing the original ordering as mentioned below:

Group 1: Less than 10

Group 2: Between 10 and 19

Group 3: Between 20 and 29

Group 4: Between 30 and 99

b. Sort the categorized data using any sorting algorithm */

…………………………………………………………………………………………………….

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 20
typedef struct
{
        int front, rear;
        int item[MAXSIZE];
} queue;
queue q1, q2, q3, q4;
int a[10], no=0, no2=0;
void insert(queue *q, int n)
{
        if(q->rear==MAXSIZE)
        {
                printf("Queue is full.");
        }
        else
        {
                q->item[++q->rear]=n;
        }
}
int delete(queue *q)
{
        if(q->rear<q->front)
        printf("Queue is empty.");
    else
        return q->item[q->front++];
}
//Display from Queue
void displayQ(queue *q)
{
    int i;
    no++;
    if(q->rear<q->front)
    {
        printf("Group %d: Queue is empty.",no);
    }
    else
    {

        printf("Group %d: Contents of queue %d are\n ", no,no);
        for(i=q->front; i<=q->rear; i++)
        {
            printf("%d\t", q->item[i]);
        }
    }
    printf("\n");
}
// Display the temp array
void displaySorted(int a[])
{
    no2++;
    printf("Group %d: Contents after sorting\n ", no2);
    for(int i=0; a[i]!='\0';i++)
    {
        printf("%d\t", a[i]);
        a[i]='\0';
    }
    printf("\n");
}
// Insertion Sort Function
void insertionSort()
{
    for (int i = 1; a[i]!='\0'; i++)
    {
        int element = a[i];
        int j = i - 1;
        while (j >= 0 && a[j] > element)
        {
            a[j + 1] = a[j];
            j = j - 1;
        }
        a[j + 1] = element;
    }
}
void sortqueue(queue *q)
{
    for(int i=0;q->item[i]!='\0';i++)
    {
        a[i]=delete(q);
    }
    insertionSort(a);
    displaySorted(a);
}
void main()
{
    q1.front=0,q2.front=0,q3.front=0,q4.front=0;
    q1.rear=-1,q2.rear=-1,q3.rear=-1,q4.rear=-1;
    int i,n,a[50];
    printf("Enter how many Elements:\n");
    scanf("%d",&n);
    printf("Enter %d Elements:\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    // insert with grouping
    for(i=0;i<n;i++)
    {
        if(a[i]>=0 && a[i]<10)
        {
            insert(&q1, a[i]);
        }
        else if (a[i]>=10 && a[i]<20)
        {
            insert(&q2, a[i]);
        }
        else if (a[i]>=20 && a[i]<30)
        {
                insert(&q3, a[i]);
        }
        else if (a[i]>=30)
        {
                insert(&q4, a[i]);
        }
}
printf("Categorised data into different group:\n");
displayQ(&q1);
displayQ(&q2);
displayQ(&q3);
displayQ(&q4);
printf("\nSorted data:\n");
sortqueue(&q1);
sortqueue(&q2);
sortqueue(&q3);
sortqueue(&q4);
}
```

OUTPUT:

```
Enter how many Elements:
20
Enter 20 Elements:
34 21 56 23 11 10 9 3 2 78 15 87 20 35 1 8 27 95 33 60

Categorised data into different group:
Group 1: Contents of queue 1 are
 9     3       2       1       8
Group 2: Contents of queue 2 are
 11    10      15
Group 3: Contents of queue 3 are
 21    23      20      27
Group 4: Contents of queue 4 are
 34    56      78      87      35      95      33      60

Sorted data:
Group 1: Contents after sorting
```

```
 1      2         3         8         9
Group 2: Contents after sorting
 10    11        15
Group 3: Contents after sorting
 20    21        23        27
Group 4: Contents after sorting
 33    34        35        56        60        78        87        95
```

4. Design and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters,
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

…………………………………………………………………………………………………….

PROGRAM:

```
#include<stdio.h>
#define SIZE 4
int  rear=-1, front=-1;
char queue[SIZE];
char item;
void insert()
{
        if(front==((rear + 1) % SIZE))
                printf("Queue is full.\n");
        else
        {
                rear = (rear + 1) % SIZE;
                printf("Enter ITEM: ");
                scanf("%*c%c", &item);
                queue[rear] = item;
                printf("Item inserted: %c\n", item);
                if(front == -1)         /* first element insertion into queue used for deletion */
                        front++;
        }
}
void del()
{
        if(front == -1)
                printf("Queue is empty.\n");
        else
        {
                item = queue[front];
                printf("ITEM deleted: %c\n", item);
                if(front == rear)
                {
                        front = rear = -1;
                }
                else
                        front = (front + 1) % SIZE;
        }
}
void display()
{
        int i,j;
        if(front == -1)
                printf("Queue is empty.\n");
        else
        {       printf("Elements of queue are\n");
                i = front;
                while(i != rear)
                {
                        printf("%c\t",queue[i]);
                        i = (i+1)%SIZE;
                }
                printf("%c\t",queue[i]);
                printf("\n");
        }
}
main()
{
        int choice;
        while(1)
        {
                choice=0;       /* to select default option of switch when non-integer */
                printf("\nCircular Queue Operations:\n");
                printf("1.Insert \n2.Delete \n3.Display \n4.Exit \n");
                printf("Enter your choice: ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1: insert();
                                break;
                        case 2: del();
                                break;
                        case 3: display();
                                break;
                        case 4: return;
                        default:printf("Invalid choice.\n");
```

```
                                        return;
                        }
            }
}
```

OUTPUT:

```
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: a
Item inserted: a
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: b
Item inserted: b

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: c
Item inserted: c

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: d
Item inserted: d
```

```
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Queue is full.

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Elements of queue are
a b c d

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
ITEM deleted: a

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
ITEM deleted: b

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Elements of queue are
c d
```

```
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: e
Item inserted: e

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Elements of queue are
c d e
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
ITEM deleted: c

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
ITEM deleted: d

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
ITEM deleted: e
```

```
Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2
Queue is empty.

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Queue is empty.

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1
Enter ITEM: f
Item inserted: f

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
Elements of queue are
f

Circular Queue Operations:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 4
```

5. Design and Implement a menu driven program in C for the following operations on Doubly Linked List (DLL) of Student Data with the fields: USN, Name, Dept, Marks, PhNo

a. Create a DLL of N Students 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. Display the total and average marks for each student

…………………………………………………………………………………………………

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int count;
struct node
{
        float marks1,marks2,marks3;
        char usn[15],name[20],dept[20],phno[15];
        struct node *llink;
        struct node *rlink;
};

typedef struct node *NODE;
NODE first=NULL;
NODE getnode()
{
        NODE x;
        x = (NODE)malloc(sizeof(struct node));
        return x;
}
NODE create_node()
{
        NODE temp;
        temp = getnode();
        printf("Enter Students Details : \n");
        printf("Enter usn : ");
        scanf("%s", temp->usn);
        printf("Enter Name : ");
        scanf("%s", temp->name);
        printf("Enter Dept : ");
        scanf("%s", temp->dept);
        printf("Enter marks1 : ");
        scanf("%f", &(temp->marks1));
        printf("Enter marks2 : ");
        scanf("%f", &(temp->marks2));
        printf("Enter marks3 : ");
        scanf("%f", &(temp->marks3));
        printf("Enter Phone No : ");
        scanf("%s", temp->phno);
        temp->llink = NULL;
        temp->rlink = NULL;
        count++;
        return temp;
}
void disp_deleted(NODE temp)
{
        printf("The following Student detail is deleted:\n");
        printf("USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No \n");
        printf("---------------------------------------------------------------------------------\n");
        printf("%s | %s | %s | %.2f | %.2f | %.2f | %s \n",temp->usn,temp->name,temp->dept,
        temp->marks1,temp->marks2,temp->marks3,temp->phno);
        count--;
}
void insert_front()
{
        NODE temp;
        temp = create_node();
        if(first == NULL)
        {
                first = temp;
        }
        else
        {
                temp->rlink = first;
                first->llink = temp;
                first = temp;
        }
}
void delete_front()
{
        NODE temp;

        if(first == NULL)
        {
                printf("List is Empty\n");

        }
        else if(first->rlink == NULL)
        {
                disp_deleted(first);
                free(first);
                first = NULL;
        }
        else
        {
                temp = first;
                disp_deleted(temp);
                first = first->rlink;
                first->llink = NULL;
                temp->rlink = NULL;
                free(temp);
                temp = NULL;
        }
}
void insert_rear()
{
        NODE temp,cur;
        temp = create_node();
        if(first == NULL)
        {
                first = temp;
        }
        else
        {
                cur = first;
                while(cur->rlink !=NULL)
                {
                        cur = cur->rlink;
                }

                cur->rlink = temp;
                temp->llink = cur;
        }
}
```

```
void delete_rear()
{
        NODE cur;
        if(first == NULL)
        {
                printf("List is empty\n");
        }
        else if(first->rlink == NULL)
        {
                disp_deleted(first);
                free(first);
                first = NULL;
        }
        else
        {
                cur = first;
                while(cur->rlink != NULL)
                {
                        cur = cur->rlink;
                }
                disp_deleted(cur);
                cur->llink->rlink = NULL;
                cur->llink = NULL;
                free(cur);
                cur = NULL;
        }
}
void display()
{
        NODE cur;
        float total, average;
        if(first == NULL)
        {
                printf("List is empty\n");
        }
        else
        {
                cur = first;
                printf("The student details in doubly Linked list from beginning : \n");
                printf("USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average |
                Ph. No \n");
```

```
		printf("-----------------------------------------------------------------------------------\n");
		while(cur != NULL)
		{
			total=cur->marks1 + cur->marks2 + cur->marks3;
			average=total/3;
			printf("%s | %s | %s | %.2f | %.2f | %.2f | %.2f | %.2f | %s \n",cur->usn,
			cur->name, cur->dept,cur->marks1,cur->marks2,cur->marks3,total,
			average, cur->phno);
			cur = cur->rlink;
		}
		printf("-----------------------------------------------------------------------------------\n");
		printf("Number of Nodes = %d\n",count);
	}
}
void main()
{
	int choice,i,n;
	while(1)
	{
		choice=0;
		printf("-----------------------------------------MENU---------------------------------\n");
		printf("1. Create a DLL of N student by using End Insertion\n");
		printf("2. Display Status and Count of nodes\n");
		printf("3. Insertion at rear\n");
		printf("4. Deletion at rear\n");
		printf("5. Insertion at front\n");
		printf("6. Delete at front\n");
		printf("7. Exit\n");
		printf("-----------------------------------------------------------------------------------\n");
		printf("Enter choice : ");
		scanf("%d", &choice);
		switch (choice)
		{
			case 1: printf("Enter number of student:");
				scanf("%d",&n);
				for(i=0;i<n;i++)
					insert_rear();
				break;
			case 2: display(); break;
			case 3: insert_rear(); break;
			case 4: delete_rear(); break;
```

```
                        case 5: insert_front(); break;
                        case 6: delete_front(); break;
                        case 7: return;
                    default: printf("Invalid choice\n"); return;

                }
        }
}
```

OUTPUT:

```
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
------------------------------------------------------------------------------
Enter choice : 1
Enter number of student:2
Enter Students Details :
Enter usn : 4SF21CS001
Enter Name : Ananya
Enter Dept : CS
Enter marks1 : 56
Enter marks2 : 78
Enter marks3 : 55
Enter Phone No : 4568756432
Enter Students Details :
Enter usn : 4SF21CS002
Enter Name : Suma
Enter Dept : CS
Enter marks1 : 78
Enter marks2 : 90
Enter marks3 : 56
Enter Phone No : 3456789432
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
```

```
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 189.00 | 63.00 | 4568756432
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 224.00 | 74.67 | 3456789432
--------------------------------------------------------------------------------
Number of Nodes = 2
----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 3
Enter Students Details :
Enter usn : 4SFIS21CS004
Enter Name : Ajay
Enter Dept : IS
Enter marks1 : 90
Enter marks2 : 78
Enter marks3 : 87
Enter Phone No : 4567894567
----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
```

```
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 189.00 | 63.00 | 4568756432
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 224.00 | 74.67 | 3456789432
4SFIS21CS004 | Ajay | IS | 90.00 | 78.00 | 87.00 | 255.00 | 85.00 | 4567894567
--------------------------------------------------------------------------------
Number of Nodes = 3
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 4
The following Student detail is deleted:
USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No
--------------------------------------------------------------------------------
4SFIS21CS004 | Ajay | IS | 90.00 | 78.00 | 87.00 | 4567894567
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 189.00 | 63.00 | 4568756432
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 224.00 | 74.67 | 3456789432
--------------------------------------------------------------------------------
Number of Nodes = 2
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
```

```
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 5
Enter Students Details :
Enter usn : 4SF21AI001
Enter Name : Navya
Enter Dept : AI
Enter marks1 : 56
Enter marks2 : 67
Enter marks3 : 78
Enter Phone No : 44522134567
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21AI001 | Navya | AI | 56.00 | 67.00 | 78.00 | 201.00 | 67.00 | 44522134567
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 189.00 | 63.00 | 4568756432
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 224.00 | 74.67 | 3456789432
--------------------------------------------------------------------------------
Number of Nodes = 3
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
```

```
7. Exit
--------------------------------------------------------------------------------
Enter choice : 6
The following Student detail is deleted:
USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No
--------------------------------------------------------------------------------
4SF21AI001 | Navya | AI | 56.00 | 67.00 | 78.00 | 44522134567
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 189.00 | 63.00 | 4568756432
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 224.00 | 74.67 | 3456789432
--------------------------------------------------------------------------------
Number of Nodes = 2
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 4
The following Student detail is deleted:
USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No
--------------------------------------------------------------------------------
4SF21CS002 | Suma | CS | 78.00 | 90.00 | 56.00 | 3456789432
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
```

```
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 4
The following Student detail is deleted:
USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No
--------------------------------------------------------------------------------
4SF21CS001 | Ananya | CS | 56.00 | 78.00 | 55.00 | 4568756432
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 4
List is empty
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
List is empty
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
```

```
--------------------------------------------------------------------------------
Enter choice : 5
Enter Students Details :
Enter usn : 4SF21CS007
Enter Name : Shashank
Enter Dept : CS
Enter marks1 : 89
Enter marks2 : 78
Enter marks3 : 98
Enter Phone No : 4567845678
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
The student details in doubly Linked list from beginning :
USN | Name | Dept | Marks1 |Marks2 | Marks3 | Total Marks | Average | Ph. No
--------------------------------------------------------------------------------
4SF21CS007 | Shashank | CS | 89.00 | 78.00 | 98.00 | 265.00 | 88.33 | 4567845678
--------------------------------------------------------------------------------
Number of Nodes = 1
----------------------------------------MENU--------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 6
The following Student detail is deleted:
USN | Name | Dept | Marks1 | Marks2 | Marks3 | Ph. No
--------------------------------------------------------------------------------
4SF21CS007 | Shashank | CS | 89.00 | 78.00 | 98.00 | 4567845678
----------------------------------------MENU--------------------------------------
```

```
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 6
List is Empty
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 2
List is empty
-----------------------------------------MENU---------------------------------------
1. Create a DLL of N student by using End Insertion
2. Display Status and Count of nodes
3. Insertion at rear
4. Deletion at rear
5. Insertion at front
6. Delete at front
7. Exit
--------------------------------------------------------------------------------
Enter choice : 7
```

6. Design 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).

……………………………………………………………………………………………………

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct node
{
        int coef;
        int expox;
        int expoy;
        int expoz;
        struct node *link;
};
typedef struct node *NODE;
NODE createnode(int coef, int ex, int ey, int ez)
{
        NODE temp;
        temp=(NODE)malloc(sizeof(struct node));
        temp->coef = coef;
        temp->expox = ex;
        temp->expoy = ey;
        temp->expoz = ez;
        temp->link = NULL;
        return temp;
}
NODE createheadnode()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        temp->coef = 0;
        temp->expox = -1;
        temp->expoy = -1;
        temp->expoz = -1;
        temp->link = temp;
        return temp;
}
```

```
void insert_rear(int coef, int ex, int ey, int ez, NODE head)
{
        NODE temp,cur;
        temp = createnode(coef,ex,ey,ez);
        if(head->link == head)
        {
                head->link = temp;
        }
        else
        {
                cur = head;
                while(cur->link != head)
                {
                        cur = cur->link;
                }
                cur->link = temp;
        }
        temp->link = head;
        head->coef = (head->coef) + 1;   //increment node count in header node
}
void createpoly(NODE poly)
{
        int i,n;
        int coef,ex,ey,ez;
        printf("Enter the number of terms in the polynomial:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter the coefficient: ");
                scanf("%d",&coef);
                printf("Enter the exponent of (x,y,z): ");
                scanf("%d%d%d",&ex,&ey,&ez);
                insert_rear(coef,ex,ey,ez,poly);
        }
}
void display(NODE head)
{
        NODE cur;
        if(head->link == head)
        {
                printf("List is empty\n");
```

```
	}

	else
	{
		cur = head->link;
		while(cur!= head)
		{
			if(cur->coef < 0)
				printf("%dx^%dy^%dz^%d ",cur->coef,cur->expox,cur->expoy,
				cur->expoz);
			else
				printf("+%dx^%dy^%dz^%d ",cur->coef,cur->expox,cur->expoy,
				cur->expoz);
			cur = cur->link;
		}
		printf("\nNumber of terms = %d\n",head->coef);
	}
}
double evaluate(int x, int y, int z,NODE head)
{
	double result = 0;
	NODE cur;
	if(head->link == head)
	{
		printf("List is empty\n");
	}
	else
	{
		cur = head->link;
		while(cur != head)
		{
			result += cur->coef * pow(x,cur->expox) *  pow(y,cur->expoy) *
					pow(z,cur->expoz);
			cur = cur->link;
		}
	}
	return result;
}
NODE polyadd(NODE a, NODE b)
{
	NODE c,starta,startb;
```

```
int sum = 0;
starta = a;
startb = b;
a =a->link;
b = b->link;
c = createheadnode();
while((a != starta) && (b != startb))
  {
        if((a->expox == b->expox) && (a->expoy == b->expoy) && (a->expoz ==
                b-> expoz))
        {
                sum = a->coef + b->coef;
                insert_rear(sum,a->expox,a->expoy,a->expoz,c);
                a = a->link;
                b = b->link;
        }
        else if(a->expox > b->expox)
        {
                insert_rear(a->coef,a->expox,a->expoy,a->expoz,c);
                a = a->link;
        }
        else if((a->expox == b->expox) && (a->expoy > b->expoy))
        {
                insert_rear(a->coef,a->expox,a->expoy,a->expoz,c);
                a = a->link;
        }
        else if((a->expox == b->expox) && (a->expoy == b->expoy) && (a->expoz >
                b->expoz))
        {
                insert_rear(a->coef,a->expox,a->expoy,a->expoz,c);
                a = a->link;
        }
        else
        {
                insert_rear(b->coef,b->expox,b->expoy,b->expoz,c);
                b = b->link;
        }

}
/* attach the remaining terms in the polynomial to end of resultant polynomial */
while(a != starta )
```

```
        {
                insert_rear(a->coef,a->expox,a->expoy,a->expoz,c);
                a = a->link;
        }
        while(b != startb )
        {
                insert_rear(b->coef,b->expox,b->expoy,b->expoz,c);
                b = b->link;
        }
        return c;
}
void main()
{
        int x,y,z;
        double eval=0;
        NODE poly1= NULL;
        NODE poly2 = NULL;
        NODE polysum = NULL;

        printf("*****Evaluation of a Polynomial****\n");
        poly1 = createheadnode();
        createpoly(poly1);
        printf("Polynomial is:\n");
        display(poly1);
        printf("Enter the values for x,y,z:");
        scanf("%d%d%d",&x,&y,&z);
        eval = evaluate(x,y,z,poly1);
        printf("Polynomial Evaluation value = %.2f \n",eval);

        printf("\n*****Adition of two polynomials*****\n");
        poly1 = NULL;
        poly1 = createheadnode();
        createpoly(poly1);
        poly2 = createheadnode();
        createpoly(poly2);
        printf("Polynomial-1 is:\n");
        display(poly1);
        printf("Polynomial-2 is:\n");
        display(poly2);
        polysum = polyadd(poly1,poly2);
        printf("Polynomial sum is:\n");
        display(polysum);
}
```

OUTPUT:

```
*****Evaluation of a Polynomial****
Enter the number of terms in the polynomial:5
Enter the coefficient: 6
Enter the exponent of (x,y,z): 2 2 1
Enter the coefficient: -4
Enter the exponent of (x,y,z): 0 1 5
Enter the coefficient: 3
Enter the exponent of (x,y,z): 3 1 1
Enter the coefficient: 2
Enter the exponent of (x,y,z): 1 5 1
Enter the coefficient: -2
Enter the exponent of (x,y,z): 1 1 1
Polynomial is:
+6x^2y^2z^1 -4x^0y^1z^5 +3x^3y^1z^1 +2x^1y^5z^1 -2x^1y^1z^1
Number of terms = 5
Enter the values for x,y,z:2 2 1
Polynomial Evaluation value = 256.00

*****Adition of two polynomials*****
Enter the number of terms in the polynomial:3
Enter the coefficient: 8
Enter the exponent of (x,y,z): 4 3 3
Enter the coefficient: 7
Enter the exponent of (x,y,z): 3 3 3
Enter the coefficient: 4
Enter the exponent of (x,y,z): 2 2 1
Enter the number of terms in the polynomial:5
Enter the coefficient: 7
Enter the exponent of (x,y,z): 4 4 3
Enter the coefficient: 16
Enter the exponent of (x,y,z): 4 3 3
Enter the coefficient: 7
Enter the exponent of (x,y,z): 3 3 3
Enter the coefficient: 3
Enter the exponent of (x,y,z): 3 2 1
Enter the coefficient: -7
Enter the exponent of (x,y,z): 2 2 1
```

```
Polynomial-1 is:
+8x^4y^3z^3 +7x^3y^3z^3 +4x^2y^2z^1
Number of terms = 3
Polynomial-2 is:
+7x^4y^4z^3 +16x^4y^3z^3 +7x^3y^3z^3 +3x^3y^2z^1 -7x^2y^2z^1
Number of terms = 5
Polynomial sum is:
+7x^4y^4z^3 +24x^4y^3z^3 +14x^3y^3z^3 +3x^3y^2z^1 -3x^2y^2z^1
Number of terms = 5
```

7. Design and Implement a program in C that reads a list of names and telephone numbers to inserts them into a Binary Search Tree for the following operations,
   a. Search the list for a specified name.
   b. Insert a new name.
   c. Delete an existing name.
   d. Traverse the phone list using Inorder, Preorder and Postorder.

……

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
        char name[25],phno[15];
        struct node* leftChild, *rightChild;
};
typedef struct node *NODE;
int delflag;
NODE createnode()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        printf("Enter the name:");
        scanf("%s",temp->name);
        printf("Enter the phone number:");
        scanf("%s",temp->phno);
        temp->leftChild = NULL;
        temp->rightChild = NULL;
        return temp;
}
void insertBST(NODE root, NODE newNode)
{
        if(strcmp(newNode->name,root->name)==0)
        {
                printf("Key already exists\n");
                return;
        }
        else if (strcmp(newNode->name,root->name)<0)
        {
                if (root->leftChild == NULL)
                        root->leftChild = newNode;
                else
                        insertBST(root->leftChild, newNode);
        }
        else
        {
                if (root->rightChild == NULL)
                        root->rightChild = newNode;
                else
                        insertBST(root->rightChild, newNode);
        }
}
int search(NODE root, char keyname[])
{
        if(!root)
                return -1;
        if(strcmp(keyname,root->name)==0)
                return 1;
        else if(strcmp(keyname,root->name)<0)
                return search(root->leftChild, keyname);
        else
                return search(root->rightChild,keyname);
}
NODE getRightMin(NODE root)
{
   NODE temp = root;
   while(temp->leftChild != NULL)
   {
        temp = temp->leftChild;
   }
   return temp;
}
NODE deleteBST(NODE root, char keyname[])
{
        if(!root)
        {
                delflag=-1;
                return NULL;
        }
        if(strcmp(keyname,root->name)<0)
                root->leftChild=deleteBST(root->leftChild, keyname);
        else if(strcmp(keyname,root->name)>0)
```

```
            root->rightChild=deleteBST(root->rightChild,keyname);
        else
        {
            if(root->leftChild==NULL && root->rightChild==NULL)
            {
                free(root);
                return NULL;
            }
            else if(root->leftChild == NULL)
            {
                NODE temp = root->rightChild;
                free(root);
                return temp;
            }
            else if(root->rightChild == NULL)
            {
                NODE temp = root->leftChild;
                free(root);
                return temp;
            }
            else
            {
                NODE rightMin = getRightMin(root->rightChild);
                strcpy(root->name,rightMin->name);
                strcpy(root->phno,rightMin->phno);
                root->rightChild = deleteBST(root->rightChild,rightMin->name);
            }

        }
        return root;
}
void inorder(NODE temp)
{
        if (temp != NULL)
        {
            inorder(temp->leftChild);
            printf("|%s|%s|\t", temp->name,temp->phno);
            inorder(temp->rightChild);
        }
}
void preorder(NODE temp)
```

```
{
        if (temp != NULL)
        {
                printf("|%s|%s|\t", temp->name,temp->phno);
                preorder(temp->leftChild);
                preorder(temp->rightChild);
        }
}
void postorder(NODE temp)
{
        if (temp != NULL)
        {
                postorder(temp->leftChild);
                postorder(temp->rightChild);
                printf("|%s|%s|\t", temp->name,temp->phno);
        }
}
void main()
{
        int choice,n,i,keyFound = 0;
        char keyname[25];
        NODE root=NULL,newNode;
        printf("--------------------Creating a BST------------------\n");
        printf("Enter the number of records in the BST:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                newNode = createnode();
                if(root == NULL)
                        root = newNode;
                else
                        insertBST(root,newNode);
        }
        while(1)
        {
                choice=0;
                printf("\n-----------------MENU----------------------\n");
                printf("1. Search a list for a specified name\n");
                printf("2. Insert a new name\n");
                printf("3. Deleting existing name\n");
                printf("4. Traverse the phone list\n");
                printf("5. Exit\n");
                printf("-------------------------------------------\n");
                printf("Enter choice : ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1: printf("Enter the name to be searched:");
                                        scanf("%s",keyname);
                                        keyFound = search(root,keyname);
                                        if(keyFound == 1)
                                                printf("Name: %s is found in the BST",keyname);
                                        else
                                                printf("Name: %s is not found in the BST",
                                                keyname);
                                        break;
                        case 2: newNode = createnode();
                                if(root == NULL)
                                        root = newNode;
                                else
                                        insertBST(root,newNode);
                                break;
                        case 3: if(root == NULL)
                                {
                                        printf("Tree is empty\n");
                                }
                                else
                                {
                                                delflag=0;
                                                printf("Enter the name to be deleted:");
                                                scanf("%s",keyname);
                                                root=deleteBST(root,keyname);
                                                if(delflag==-1)
                                                        printf("Name: %s is not found in the
                                                        BST\n",keyname);
                                                else
                                                        printf("Name: %s is deleted from the
                                                        BST\n",keyname);
                                        }
                                        break;
                        case 4: if (root == NULL)
                                        {
```

```
                                                printf("Tree is empty\n");
                                        }
                                        else
                                        {
                                                printf("BST Preorder traversal\n");
                                                preorder(root);
                                                printf("\nBST Inorder traversal\n");
                                                inorder(root);
                                                printf("\nBST Postorder traversal\n");
                                                postorder(root);
                                        }
                                        break;
                        case 5: return;
                        default:printf("Wrong choice\n");
                                        return;
                }
        }
}
```

OUTPUT:

```
--------------------Creating a BST------------------
Enter the number of records in the BST:5
Enter the name:Shreya
Enter the phone number:5678945678
Enter the name:Adya
Enter the phone number:5799543217
Enter the name:Sumana
Enter the phone number:5890654321
Enter the name:Poorna
Enter the phone number:6906543268
Enter the name:Vaishnavi
Enter the phone number:5673423456

-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
------------------------------------------
Enter choice : 4
```

```
BST Preorder traversal
|Shreya|5678945678|         |Adya|5799543217|    |Poorna|6906543268|  |Sumana|5890654321|
|Vaishnavi|5673423456|

BST Inorder traversal
|Adya|5799543217|  |Poorna|6906543268|  |Shreya|5678945678|  |Sumana|5890654321|
|Vaishnavi|5673423456|

BST Postorder traversal
|Poorna|6906543268|         |Adya|5799543217|    |Vaishnavi|5673423456|
|Sumana|5890654321|         |Shreya|5678945678|
-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 1
Enter the name to be searched:Vaishnavi
Name: Vaishnavi is found in the BST
-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 1
Enter the name to be searched:Ananya
Name: Ananya is not found in the BST
-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 2
Enter the name:Chaithra
Enter the phone number:4567321000

-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 4

BST Preorder traversal
|Shreya|5678945678|         |Adya|5799543217|    |Poorna|6906543268|
|Chaithra|4567321000|
|Sumana|5890654321|        |Vaishnavi|5673423456|

BST Inorder traversal
|Adya|5799543217|  |Chaithra|4567321000|         |Poorna|6906543268|  |Shreya|5678945678|
|Sumana|5890654321|        |Vaishnavi|5673423456|

BST Postorder traversal
|Chaithra|4567321000|        |Poorna|6906543268|  |Adya|5799543217| |Vaishnavi|5673423456|
|Sumana|5890654321|        |Shreya|5678945678|
-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 3
Enter the name to be deleted:Poorna
Name: Poorna is deleted from the BST

-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
```

```
Enter choice : 4

BST Preorder traversal
|Shreya|5678945678|         |Adya|5799543217|    |Chaithra|4567321000|
|Sumana|5890654321|
|Vaishnavi|5673423456|

BST Inorder traversal
|Adya|5799543217|  |Chaithra|4567321000|         |Shreya|5678945678|  |Sumana|5890654321|
|Vaishnavi|5673423456|

BST Postorder traversal
|Chaithra|4567321000|   |Adya|5799543217|   |Vaishnavi|5673423456|   |Sumana|5890654321|
|Shreya|5678945678|
-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 3
Enter the name to be deleted:Anu
Name: Anu is not found in the BST

-----------------MENU----------------------
1. Search a list for a specified name
2. Insert a new name
3. Deleting existing name
4. Traverse the phone list
5. Exit
-------------------------------------------
Enter choice : 5
```

8. A company has seven top officers working for it. They are each fluent in atleast one language according to the following sample table:

Officer	**Hindi**	**Malayalam**	**Kannada**	**Telugu**
01	**-**	**-**	**Y**	**-**
02	**-**	**-**	**Y**	**Y**
03	**-**	**-**	**-**	**Y**
04	**-**	**Y**	**-**	**Y**
05	**Y**	**Y**	**-**	**-**
06	**Y**	**-**	**Y**	**-**
07	**-**	**Y**	**-**	**-**

Design and Implement a program in C for the following operations on Graphs (G),

a. Create a graph using adjacency matrix indicating people who can communicate directly with each other.

b. Print all the officers which are reachable from a given officer as a starting node in a digraph.

Example: An officer wants to send a message to each other officer: A message comes to an officer, he reads it and transmits it to another officer possibly after translation to someone who has not read it.

……

PROGRAM:

```
#include<stdio.h>
int a[20][20],q[20],visited[20],reach[20],n,i,j,f=0,r=-1,count=0;
void bfs(int v)
{
        int u;
        q[++r] = v;
        visited[v] = 1;
        while(f <= r)
        {
                u = q[f++];
                for(i=1;i<=n;i++)
                {
                        if(a[u][i] && !visited[i])
                        {
                                q[++r]=i;
                                visited[i] = 1;
                                printf("->%d",i);
                        }
                }

        }
}
void dfs(int v)
{
        int i;
        reach[v]=1;
        for(i=1;i<=n;i++)
        {
                if(a[v][i] && !reach[i])
                {
                        printf("->%d", i);
                        dfs(i);
                }
        }
}
void main()
{
        int v;
        printf("Enter the number of officers: ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                q[i]=0;
                reach[i]=0;
                visited[i]=0;
        }
        printf("Enter graph data in matrix form indicating people who can communicate directly
with each other:\n");
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                        scanf("%d",&a[i][j]);
        printf("Enter the officer as a starting node: ");
        scanf("%d",&v);
        if((v<1)||(v>n))
        {
                printf("Invalid node\n");
                return;
```

```
	}
	printf("All the officers which are reachable from a given officer as a starting node using
BFS algorithm\n %d", v);
	bfs(v);
	printf("\nAll the officers which are reachable from a given officer as a starting node
using DFS algorithm\n %d", v);
	dfs(v);
	printf("\n");
}
```

OUTPUT:

```
Enter the number of officers: 7
Enter graph data in matrix form indicating people who can communicate directly with each
other:

0 1 0 0 0 1 0
1 0 1 1 0 1 0
0 1 0 1 0 0 0
0 1 1 0 1 0 1
0 0 0 1 0 1 1
1 1 0 0 1 0 0
0 0 0 1 1 0 0
Enter the officer as a starting node: 1
All the officers which are reachable from a given officer as a starting node using BFS
algorithm
 1->2->6->3->4->5->7
All the officers which are reachable from a given officer as a starting node using DFS
algorithm
 1->2->3->4->5->6->7
```

…………………………………………………………………………………………………

```
Enter the number of officers: 7
Enter graph data in matrix form indicating people who can communicate directly with each
other:
0 1 0 0 0 1 0
1 0 1 1 0 1 0
0 1 0 1 0 0 0
0 1 1 0 1 0 1
0 0 0 1 0 1 1
1 1 0 0 1 0 0
0 0 0 1 1 0 0
Enter the officer as a starting node: 6
```

```
All the officers which are reachable from a given officer as a starting node using BFS
algorithm
 6->1->2->5->3->4->7
All the officers which are reachable from a given officer as a starting node using DFS
algorithm
 6->1->2->3->4->5->7
```

9. Design and Implement a program in C that uses Hash Function H:K->L as H(K)=K mod m(reminder method) and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

……………………………………………………………………………………………………

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int key,n,m,*ht,hi,elec,flag;
void createht()
{
        int i;
        ht = (int*)malloc(m*sizeof(int));
        if(m==0)
        {
                printf("Unable to create the hash table\n");
                exit(0);
        }
        else
                for(i=0; i<m; i++)
                  ht[i] = -1;
}
void insertht(int key)
{
        hi = key % m;
        while(ht[hi] != -1)
        {
                hi = (hi+1)%m;
                flag = 1;
        }
        if(flag)
        {
          printf("Collision Detected and avoided by Linear Probing!\n");
                flag = 0;
        }
        ht[hi] = key;
        elec++;
}
void displayht()
{
        int i;
        if(elec == 0)
```

```
	{
		printf("Hash Table is empty\n");
		return;
	}
	printf("Hash Table contents are:\n");
	for(i=0; i<m; i++)
		printf("[%d] --> %d\n", i, ht[i]);
}
void main()
{
	int i;
	printf("Enter the number of employee  records: ");
	scanf("%d", &n);
	printf("Enter the two digit memory locations: ");
	scanf("%d", &m);
	createht();
	printf("Enter four digit key values of Employee records\n");
	for(i=0; i<n; i++)
	{
		scanf("%d", &key);
		if(elec == m)
		{
			printf("Hash table is full.\n");
			break;
		}
		insertht(key);
	}
	displayht();
}
```

OUTPUT:

```
Enter the number of employee records: 5
Enter the two digit memory locations: 10
Enter four digit key values of Employee records
1234
1456
1784
Collision Detected and avoided by Linear Probing!
1890
1536
Collision Detected and avoided by Linear Probing!
```

```
Hash Table contents are:
[0] --> 1890
[1] --> -1
[2] --> -1
[3] --> -1
[4] --> 1234
[5] --> 1784
[6] --> 1456
[7] --> 1536
[8] --> -1
[9] --> -1
..................................................................................................
Enter the number of employee records: 5
Enter the two digit memory locations: 00
Unable to create the hash table
..................................................................................................
Enter the number of employee records: 5
Enter the two digit memory locations: 4
Enter four digit key values of Employee records
1456
1321
1676
Collision Detected and avoided by Linear Probing!
7845
Collision Detected and avoided by Linear Probing!
8952
Hash table is full.
Hash Table contents are:
[0] --> 1456
[1] --> 1321
[2] --> 1676
[3] --> 7845
..................................................................................................
Enter the number of employee records: 8
Enter the two digit memory locations: 20
Enter four digit key values of Employee records
1890
1678
1111
3458
Collision Detected and avoided by Linear Probing!
```

```
1342
1876
1456
Collision Detected and avoided by Linear Probing!
1278
Collision Detected and avoided by Linear Probing!
Hash Table contents are:
[0] --> 1278
[1] --> -1
[2] --> 1342
[3] --> -1
[4] --> -1
[5] --> -1
[6] --> -1
[7] --> -1
[8] --> -1
[9] --> -1
[10] --> 1890
[11] --> 1111
[12] --> -1
[13] --> -1
[14] --> -1
[15] --> -1
[16] --> 1876
[17] --> 1456
[18] --> 1678
[19] --> 3458
```

VIVA QUESTIONS AND ANSWERS

1) What is data structure?

Data structures refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with data structure, we not only focus on one piece of data, but rather different set of data and how they can relate to one another in an organized manner.

2) Differentiate file structure from storage structure.

Basically, the key difference is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is search starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link of data storage.

5) How do you reference all the elements in a one-dimension array?

To do this, an indexed loop is used, such that the counter runs from 0 to the array size minus one. In this manner, we are able to reference all the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures applied?

Data structures is important in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?

LIFO is short for Last In First Out, and refers to how data is accessed, stored and retrieved. Using this scheme, data that was stored last , should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?

A queue is a data structures that can simulates a list or stream of data. In this structure, new elements are inserted at one end and existing elements are removed from the other end.

9) What are binary trees?

A binary tree is one type of data structure that has two nodes, a left node and a right node. In programming, binary trees are actually an extension of the linked list structures.

10) Which data structures is applied when dealing with a recursive function?

Recursion, which is basically a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree.
A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?
Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using a single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?
It actually depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?
Aside from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?
FIFO is short for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?
An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?
Merge sort takes a divide-and-conquer approach to sorting data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continuous until you have one single sorted list.

19) Differentiate NULL and VOID.

Null is actually a value, whereas Void is a data type identifier. A variable that is given a Null value simply indicates an empty value. Void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?
A linked list is a very ideal data structure because it can be modified easily. This means that modifying a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?
Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular refers to the topmost data being accessed.

22) What is a linear search?
A linear search refers to the way a target key is being searched in a sequential data structure. Using this method, each element in the list is checked and compared against the target key, and is repeated until found or if the end of the list has been reached.

23) How does variable declaration affect memory allocation?
The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?
Basically, the heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?
A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is the difference between the HEAP and the STACK?
HEAP is used to store dynamically allocated memory (malloc). STACK stores static data
(int, const).)

27) Where in memory are HEAP and the STACK located relative to the executing program?
The STACK and HEAP are stored "below" the executing program. The HEAP "grows" toward the program executable while the STACK grows away from it.

28) Describe the data structures of a double-linked list.
A double-linked list structure contains one pointer to the previous record in the list and a pointer to the next record in the list plus the record data.

29) How do you insert a record between two nodes in double-linked list?
Previous R; Data R; Next R; To insert a record (B) between two others (A and C): Previous.B = A; Next.B = C; Next.A = B; Previous.C = B;)

30) In which data structure, elements can be added or removed at either end, but not in the middle?
Dequeue.

31) Which one is faster? A binary search of an orderd set of elements in an array or a sequential search of the elements.
binary search

32) What is a balanced tree?
A binary tree is balanced if the depth of two subtrees of every node never differ by more than one.

33) Which data structure is needed to convert infix notations to post fix notations?
Stack.

34) What is data structure or how would you define data structure?
Data Structure is defined as the way in which data is organized in the memory location..
Data Structure = Organized Data + Allowed Operations.

35) Which data structures we can implement using link list?
queue and stack.

36) List different types of data structures?
Link list, queue, stack, trees, files, graphs.

37) Define a linear and non linear data structure.
Linear data structure: In linear data structure all the data are stored linearly or contiguously in the memory. A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached. Ex: Arrays, Stack, Oueue, Linked Lists

Non-Linear data structure: Every data item is attached to several other data items in a way that is specific for reflecting relationships. The data items are not arranged in a sequential structure. Ex: Trees, Graphs.