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>
int a[20],n,elem,i,pos;
void create();
void display();
void insert();
void delete();
main()
{
        int choice;
        while(1)
        {
                choice=0;       /* to select default option of switch when non-integer */
                printf("\n--------MENU----------\n");
                printf("1. CREATE\n");
                printf("2. DISPLAY\n");
                printf("3. INSERT\n");
                printf("4. DELETE\n");
                printf("5. EXIT\n");
                printf("-----------------------\n");
                printf("ENTER YOUR CHOICE:\t");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: create();
                                break;
                        case 2: display();
                                break;
                        case 3: insert();
                                break;
                        case 4: delete();
                                break;
                        case 5: exit(0);
                        default:printf("Invalid choice:\n");
                                return;
                }
        }
}
```

```
void create()
{
        printf("Enter the size of the array elements:\t");
        scanf("%d",&n);
        printf("Enter the elements for the array:\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
}
void display()
{
        printf("The array elements are:\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
}
void insert()   //inserting an element into an array
{
        printf("Enter the position for the new element:\t");
        scanf("%d",&pos);
        pos--; /*actual position is one more than array index*/
        if(pos>=0 && pos<=n)
        {
                printf("Enter the element to be inserted :\t");
                scanf("%d",&elem);
                for(i=n-1;i>=pos;i--)
                {
                        a[i+1]=a[i];
                }
                a[pos]=elem;
                n=n+1;
        }
        else
        {
                printf("Invalid position\n");
        }
}
void delete()   //deleting an array element
{
        printf("Enter the position of the element to be deleted:\t");
        scanf("%d",&pos);
        pos--;
        if(pos>=0 && pos<n)
        {
                elem=a[pos];
                for(i=pos;i<n-1;i++)
                {
                        a[i]=a[i+1];
                }
                n=n-1;
                printf("The deleted element is =%d",elem);
        }
        else
        {
                printf("Invalid position\n");
        }
}
```

STEPS TO EXECUTE

```
1 gedit array.c
2 gcc array.c -o arry.out
3 ./arry.out
```

OUTPUT:

```
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:    1
Enter the size of the array elements:   5
Enter the elements for the array:
11      22      33      44      55

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:    2
The array elements are:
11      22      33      44      55
```

```
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      3
Enter the position for the new element:         7
Invalid position

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      3
Enter the position for the new element:         0
Invalid position

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      3
Enter the position for the new element:         6
Enter the element to be inserted :      66

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      2
The array elements are:
11      22      33      44      55      66
```

```
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      3
Enter the position for the new element:         3
Enter the element to be inserted :      77

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      2
The array elements are:
11      22      77      33      44      55      66
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      4
Enter the position of the element to be deleted:        8
Invalid position

--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:      4
Enter the position of the element to be deleted:        3
The deleted element is =77
```

```
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
5. EXIT
-----------------------
ENTER YOUR CHOICE:    2
The array elements are:
11      22      33      44      55      66
--------MENU----------
1. CREATE
2. DISPLAY
3. INSERT
4. DELETE
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>
char ans[100], str[100], pat[100], rep[100];
int flag,i,j,k,l,p;
void matchpattern()
{
        while (str[i]!= '\0')
        {
                // Checking for Match
                if ( str[j] == pat[p] )
                {
                        p++;
                        j++;
                        if ( pat[p] == '\0')     /*to check pattern reach end or not  */
                        {
                                flag=1;
                                for(k=0;  rep[k] != '\0'; k++,l++)     //copy replace string in ANS string
                                {
                                        ans[l] = rep[k];
                                }
                                p=0;    // if more occurence of pattern
                                i=j;    /*next character after pattern is matched */
                        }
                }
                else   //mismatch
                {
                        ans[l] = str[i];
                        l++;
                        i++;
                        j = i;
                        p=0;
                }
        }
        ans[l] = '\0';
}
void readstr()
{
        printf("Enter the main string: \n");
        scanf("%[^\n]",str);
        printf("Enter a pattern string: \n");
```

```
	scanf("%*c%[^\n]",pat); /* %*c is to skip enter keypress */
	printf("Enter a replace string: \n");
	scanf("%*c%[^\n]",rep);
}
main()
{
	readstr();
	matchpattern();

	if(flag==0)
	{
		printf("Pattern not found!!!\n");
	}
	else
	{
		printf("The RESULTANT string is:\n%s\n" ,ans);
	}
}
```

Output:

```
Enter the MAIN string:
good morning
Enter a PATTERN string:
morning
Enter a REPLACE string:
evening
The RESULTANT string is:
good evening
……………………………………………………………………………………………………………
Enter the MAIN string:
hai sahyadri
Enter a PATTERN string:
bye
Enter a REPLACE string:
hello
Pattern not found!!!
……………………………………………………………………………………………………………
Enter the MAIN string:
department of cse, department of ise
Enter a PATTERN string:
department
Enter a REPLACE string:
dept.
The RESULTANT string is:
dept. of cse, dept. of ise
```

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<stdlib.h>
#include<stdio.h>
#define MAX 5
int stack[MAX],top=-1,item;
void push() //Inserting element into the stack
{
	if(top==(MAX-1))
	{
		printf("Stack Overflow!!\n");
	}
	else
	{
		printf("Enter the element to be inserted:  ");
		scanf("%d",&item);
		stack[++top]=item;
	}
}
void pop() //deleting an element from the stack
{
	if(top==-1)
	{
		printf("Stack Underflow!!\n");
	}
	else
	{
		item=stack[top--];
		printf("The popped element: %d\n",item);
	}
}
```

```
void display()
{
        int i;
        if(top==-1)
        {
                printf("Stack is Empty\n");
        }
        else
        {
                printf("The stack elements are:\n" );
                for(i=top;i>=0;i--)
                {
                        printf("%d\n",stack[i]);
                }
        }
}
void palindrome()
{
        int j=top,k=0,flag=0;
        if(top == -1)
        {
                printf("Stack is empty.\n");
                return;
        }
        while(k<=top/2)
        {
                if(stack[k++]!=stack[j--])
                {
                        flag=1;
                        break;
                }
        }
        if(flag)
        {
                printf("Stack contents are not a palindrome\n");
        }
        else
        {
                printf("Stack contents are palindrome\n");
        }
}
```

```
main()
{
	int choice;
	while(1)
	{
		choice=0;
		printf("--------STACK OPERATIONS-----------\n");
		printf("1.Push\n");
		printf("2.Pop\n");
		printf("3.Palindrome\n");
		printf("4.Display\n");
		printf("5.Exit\n");
		printf("-----------------------------------\n");
		printf("Enter your choice: ");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1: push();
				break;
			case 2: pop();
				break;
			case 3: palindrome();
				break;
			case 4: display();
				break;
			case 5: exit(0);
			default:printf("Invalid choice:\n");
				return;
		}
	}
}
```

Output

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 1
Enter the element to be inserted:  8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
```

```
4.Display
5.Exit
------------------------------------------------
Enter your choice: 1
Enter the element to be inserted:  9
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------------------------------
Enter your choice: 1
Enter the element to be inserted:  8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------
Enter your choice: 1
Enter the element to be inserted:  9
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------------------------------
Enter your choice: 4
The stack elements are:
9
8
9
8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
------------------------------------------------
Enter your choice: 3
Stack contents are not a palindrome
```

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 1
Enter the element to be inserted:  8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 4
The stack elements are:
8
9
8
9
8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 3
Stack contents are palindrome
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 1
Stack Overflow!!
```

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 2
The popped element: 8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 4
The stack elements are:
9
8
9
8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 3
Stack contents are not a palindrome
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 2
The popped element: 9
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
```

```
Enter your choice: 2
The popped element: 8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 2
The popped element: 9
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 2
The popped element: 8
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 2
Stack Underflow!!
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 4
Stack is Empty.
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
-----------------------------------------------
Enter your choice: 5
```

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>
#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);
        }
}
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 Expn: %s\nPostfix Expn: %s\n", infix, postfix);
}
```

Output

```
Enter the Infix Expression: (1+2)*(4-5)
Given Infix Expn: (1+2)*(4-5)
Postfix Expn: 12+45-*
…………………………………………………………………………………………………………………
Enter the Infix Expression: (a+b)*c/d^e%f
Given Infix Expn: (a+b)*c/d^e%f
Postfix Expn: ab+c*de^/f%
…………………………………………………………………………………………………………………
Enter the Infix Expression: 1^2^3
Given Infix Expn: 1^2^3
Postfix Expn: 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 Expn: sum+123
Postfix Expn: sum123+
…………………………………………………………………………………………………………………
```

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: +, -, *, /, %, ^

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

```
#include<stdio.h>
#include<ctype.h>
#include<math.h>
int stack[50], top=-1;
void push(int elem)
{
	stack[++top]=elem;
}
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]);
}
```

Outputs

```
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
…………………………………………………………………………………………………………………
```

5. Design, Develop and Implement a Program in C for the following Stack Applications
b. Solving Tower of Hanoi problem with n disks

………

```
#include<stdio.h>
void tower(int n, char beg, char aux, char end)
{
	if (n==1)
	{
		printf("Move disk 1 from pole %c to pole %c\n", beg, end);
		return;
	}
	tower(n-1, beg,end,aux);
	printf("Move disk %d from pole %c to pole %c\n", n, beg, end);
	tower(n-1, aux,beg,end);
}
main()
{
	int num;
	printf("Enter the number of disks : ");
	scanf("%d", &num);
	printf("The sequence of moves involved in the Tower of Hanoi are :\n");
	tower(num, 'A', 'B', 'C');
}
```

Outputs

```
Enter the number of disks : 1
The sequence of moves involved in the Tower of Hanoi are :
Move disk 1 from pole A to pole C
```

………

```
Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :
Move disk 1 from pole A to pole C
Move disk 2 from pole A to pole B
Move disk 1 from pole C to pole B
Move disk 3 from pole A to pole C
Move disk 1 from pole B to pole A
Move disk 2 from pole B to pole C
Move disk 1 from pole A to pole C
```

………

6. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate *Overflow* and *Underflow* situations on Circular QUEUE

d. Display the status of Circular QUEUE

e. Exit

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

………

```
#include<stdio.h>
#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 ",queue[i]);
                        i = (i+1)%SIZE;
                }
                printf("%c ",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
```

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<stdlib.h>
int count;
struct node
{
	int sem;
	char name[20],branch[10],usn[20],phno[15];
	struct node *link;
};
typedef struct node *NODE;
NODE first= NULL;
NODE getnode()
{
	NODE x;
	x = (NODE)malloc(sizeof(struct node));
	return x;
}
NODE create()
{
	NODE temp;
	temp = getnode();
	printf("Enter Student details:\n");
	printf("Enter USN:");
	scanf("%s",temp->usn);
	printf("Enter Name:");
	scanf("%s",temp->name);
	printf("Enter Branch:");
	scanf("%s",temp->branch);
	printf("Enter Sem:");
	scanf("%d",&(temp->sem));
	printf("Enter Phone No.:");
	scanf("%s",temp->phno);
	temp->link=NULL;
	count++;
	return temp;
}
```

```
void disp_deleted(NODE temp)
{
	printf("The following Student detail is deleted:\n");
	printf("USN | Name | Branch | Sem | Phone \n");
	printf("----------------------------------------------------------\n");
	printf("%s | %s | %s | %d | %s \n", temp->usn, temp ->name, temp ->branch, temp ->sem, temp -
>phno );
	count--;
}
void insert_front()
{
	NODE temp;
	temp = create();
	temp->link = first; //it is to link present node to the previous(first) node in the list
	first = temp;
}
void delete_front()
{
	NODE temp;
	if(first == NULL)
	{
		printf("List is Empty. Cannot delete.\n");
		return;
	}
	temp = first;
	first = first->link;
	disp_deleted(temp);
	temp->link = NULL;
	free(temp);
	temp=NULL;
}
void insert_rear()
{
	NODE temp,cur;
	temp = create();
	if(first==NULL)
	{
		first = temp;
		return;
	}
	cur = first;
	while(cur->link != NULL)
	{
		cur = cur->link;
	}
	cur->link = temp;
}
```

```
void delete_rear()
{
	NODE cur,prev;
	if(first == NULL)
	{
		printf("List is Empty. Cannot delete.\n");
		return ;
	}
	if(first->link==NULL)
	{
		disp_deleted(first);
		free(first);
		first=NULL;
		return;
	}
	prev = NULL;
	cur = first;
	while(cur->link != NULL)
	{
		prev = cur;
		cur = cur->link;
	}
	disp_deleted(cur);
	prev->link = NULL;
	free(cur);
	cur=NULL;
}
void display()
{
	NODE temp;
	if(first == NULL)
	{
		printf("List is empty. \n");
		return;
	}
	printf("The student details in Singly Linked list from beginning : \n");
	printf("USN | Name | Branch | Sem | Phone \n");
	printf("----------------------------------------------------------\n");
	temp = first;
	while (temp!= NULL)
	{
		printf("%s | %s | %s | %d | %s \n", temp->usn, temp->name,temp->branch,temp->sem,
temp->phno );
		temp = temp->link;
	}
	printf(" No of students = %d\n ", count);
}
```

```
main()
{
        int choice,n,i;
        while(1)
        {
                choice=0;
                printf("-----------------MENU-------------------------------------\n");
                printf("1. Create a SLL of N Students using front insertion\n");
                printf("2. Display from Beginning\n");
                printf("3. Insert at end\n");
                printf("4. Delete at end\n");
                printf("5. Insert at beginning\n");
                printf("6. Delete at beginning\n");
                printf("7. Exit\n");
                printf("--------------------------------------------------------\n");
                printf("Enter choice : ");
                scanf("%d", &choice);
                switch (choice)
                {
                        case 1: printf("Enter no of students : ");
                                scanf("%d", &n);
                                for(i=0;i<n;i++)
                                        insert_front();
                                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 SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 1
Enter no of students : 2
Enter Student details:
Enter USN:4sf16cs001
Enter Name:Ajay
Enter Branch:CSE
Enter Sem:3
Enter Phone No.:9988776655
Enter Student details:
Enter USN:4sf16cs002
Enter Name:Ananya
Enter Branch:CSE
Enter Sem:3
Enter Phone No.:7788556678
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs002 | Ananya | CSE | 3 | 7788556678
4sf16cs001 | Ajay | CSE | 3 | 9988776655
 No of students = 2
 -----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
```

```
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 3
Enter Student details:
Enter USN:4sf16cs003
Enter Name:Chinthan
Enter Branch:CSE
Enter Sem:3
Enter Phone No.:8866779955
----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs002 | Ananya | CSE | 3 | 7788556678
4sf16cs001 | Ajay | CSE | 3 | 9988776655
4sf16cs003 | Chinthan | CSE | 3 | 8866779955
 No of students = 3
 ----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 4
The following Student detail is deleted:
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs003 | Chinthan | CSE | 3 | 8866779955
```

```
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 5
Enter Student details:
Enter USN:4sf16cs006
Enter Name:Pavithra
Enter Branch:CSE
Enter Sem:3
Enter Phone No.:8899007788
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
----------------------------------------------------------
4sf16cs006 | Pavithra | CSE | 3 | 8899007788
4sf16cs002 | Ananya | CSE | 3 | 7788556678
4sf16cs001 | Ajay | CSE | 3 | 9988776655
 No of students = 3
 -----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
```

```
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs006 | Pavithra | CSE | 3 | 8899007788
4sf16cs002 | Ananya | CSE | 3 | 7788556678
4sf16cs001 | Ajay | CSE | 3 | 9988776655
 No of students = 3
 ----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 6
The following Student detail is deleted:
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs006 | Pavithra | CSE | 3 | 8899007788
----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
---------------------------------------------------------
4sf16cs002 | Ananya | CSE | 3 | 7788556678
4sf16cs001 | Ajay | CSE | 3 | 9988776655
 No of students = 2
 ----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
---------------------------------------------------------
```

```
Enter choice : 4
The following Student detail is deleted:
USN | Name | Branch | Sem | Phone
----------------------------------------------------------
4sf16cs001 | Ajay | CSE | 3 | 9988776655
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 4
The following Student detail is deleted:
USN | Name | Branch | Sem | Phone
----------------------------------------------------------
4sf16cs002 | Ananya | CSE | 3 | 7788556678
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 4
List is Empty. Cannot delete.
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 2
List is empty.
```

```
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 3
Enter Student details:
Enter USN:4sf14cs007
Enter Name:Anusha
Enter Branch:CSE
Enter Sem:3
Enter Phone No.:7865555555
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 2
The student details in Singly Linked list from beginning :
USN | Name | Branch | Sem | Phone
----------------------------------------------------------
4sf14cs007 | Anusha | CSE | 3 | 7865555555
 No of students = 1
 -----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
----------------------------------------------------------
Enter choice : 6
The following Student detail is deleted:
USN | Name | Branch | Sem | Phone
----------------------------------------------------------
4sf14cs007 | Anusha | CSE | 3 | 7865555555
```

```
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
-------------------------------------------------------
Enter choice : 6
List is Empty. Cannot delete.
-----------------MENU-------------------------------------
1. Create a SLL of N Students using front insertion
2. Display from Beginning
3. Insert at end
4. Delete at end
5. Insert at beginning
6. Delete at beginning
7. Exit
-------------------------------------------------------
Enter choice : 7
```

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

………

```
#include<stdio.h>
#include<stdlib.h>
int count;
struct node
{
        int ssn;
        float sal;
        char name[20],dept[10],desg[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 Employee Details : \n");
        printf("Enter SSN : ");
        scanf("%d", &(temp->ssn));
        printf("Enter Name : ");
        scanf("%s", temp->name);
        printf("Enter Department : ");
        scanf("%s", temp->dept);
        printf("Enter Designation : ");
        scanf("%s", temp->desg);
        printf("Enter Salary : ");
        scanf("%f", &(temp->sal));
        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 employee detail is deleted:\n");
	printf("SSN | Name | Dept | Designation | Salary | Ph. No \n");
	printf("----------------------------------------------------------\n");
	printf("%d | %s | %s | %s | %.2f | %s \n",temp->ssn,temp->name,temp->dept,temp->desg,temp-
>sal,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;
	if(first == NULL)
	{
		printf("List is empty\n");
	}
	else
	{
		cur = first;

		printf("The employee details in doubly Linked list from beginning : \n");
		printf("SSN | Name | Dept | Designation | Salary | Ph. No \n");
		printf("-----------------------------------------------------------\n");
		while(cur != NULL)
		{
			printf("%d | %s | %s | %s | %.2f | %s \n",cur->ssn,cur->name,cur->dept,cur-
>desg,cur->sal,cur->phno);
			cur = cur->rlink;
		}
		printf("-----------------------------------------------------------\n");
		printf("Number of Nodes = %d\n",count);
	}
}
```

```
main()
{
        int choice,i,n;
        while(1)
        {
                choice=0;
                printf("-----------------MENU--------------------------------------\n");
                printf("1. Create a DLL of N Employees 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 employees:");
                                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 Employees 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 employees:2
Enter Employee Details :
Enter SSN : 1001
Enter Name : Akash
Enter Department : cash
Enter Designation : accountant
Enter Salary : 20000
Enter Phone No : 7777777777
Enter Employee Details :
Enter SSN : 1002
Enter Name : Shreya
Enter Department : office
Enter Designation : clerk
Enter Salary : 15000
Enter Phone No : 8888888888
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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
7 - Exit
-------------------------------------------------------------------
Enter choice : 2
The employee details in doubly Linked list from beginning :
SSN | Name | Dept | Designation | Salary | Ph. No
-------------------------------------------------------------------
1001 | Akash | cash | accountant | 20000.00 | 7777777777
1002 | Shreya | office | clerk | 15000.00 | 8888888888
-------------------------------------------------------------------
Number of Nodes = 2
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 Employee Details :
Enter SSN : 1003
Enter Name : Nithin
Enter Department : sales
Enter Designation : manager
Enter Salary : 50000
Enter Phone No : 5555555555
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 employee details in doubly Linked list from beginning :
SSN | Name | Dept | Designation | Salary | Ph. No
--------------------------------------------------------------------
1001 | Akash | cash | accountant | 20000.00 | 7777777777
1002 | Shreya | office | clerk | 15000.00 | 8888888888
1003 | Nithin | sales | manager | 50000.00 | 5555555555
--------------------------------------------------------------------
Number of Nodes = 3
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 employee detail is deleted:
SSN | Name | Dept | Designation | Salary | Ph. No
--------------------------------------------------------------------
1003 | Nithin | sales | manager | 50000.00 | 5555555555
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 Employee Details :
Enter SSN : 1004
Enter Name : Avanish
Enter Department : sales
Enter Designation : supervisor
Enter Salary : 35000
Enter Phone No : 4444444444
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 employee details in doubly Linked list from beginning :
SSN | Name | Dept | Designation | Salary | Ph. No
--------------------------------------------------------------------
1004 | Avanish | sales | supervisor | 35000.00 | 4444444444
1001 | Akash | cash | accountant | 20000.00 | 7777777777
1002 | Shreya | office | clerk | 15000.00 | 8888888888
--------------------------------------------------------------------
Number of Nodes = 3
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 employee detail is deleted:
SSN | Name | Dept | Designation | Salary | Ph. No
--------------------------------------------------------------------
1004 | Avanish | sales | supervisor | 35000.00 | 4444444444
```

```
-----------------MENU-------------------------------------------
1. Create a DLL of N Employees 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 employee detail is deleted:
SSN | Name | Dept | Designation | Salary | Ph. No
----------------------------------------------------------------
1001 | Akash | cash | accountant | 20000.00 | 7777777777
-----------------MENU-------------------------------------------
1. Create a DLL of N Employees 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 employee detail is deleted:
SSN | Name | Dept | Designation | Salary | Ph. No
----------------------------------------------------------------
1002 | Shreya | office | clerk | 15000.00 | 8888888888
-----------------MENU-------------------------------------------
1. Create a DLL of N Employees 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 Employees 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 Employees 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 Employee Details :
Enter SSN : 1005
Enter Name : Reema
Enter Department : office
Enter Designation : reception
Enter Salary : 15000
Enter Phone No : 5656565656
-----------------MENU------------------------------------------
1. Create a DLL of N Employees 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 employee details in doubly Linked list from beginning :
SSN | Name | Dept | Designation | Salary | Ph. No
------------------------------------------------------------------
1005 | Reema | office | reception | 15000.00 | 5656565656
------------------------------------------------------------------
Number of Nodes = 1
-----------------MENU------------------------------------------
1. Create a DLL of N Employees 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 employee detail is deleted:
SSN | Name | Dept | Designation | Salary | Ph. No
------------------------------------------------------------------
1005 | Reema | office | reception | 15000.00 | 5656565656
-----------------MENU-----------------------------------------
1. Create a DLL of N Employees 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 Employees 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 Employees 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
```

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 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3

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

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

………

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

```
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
```

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

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

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

e. Exit

………

```
# include<stdio.h>
# include<stdlib.h>
int flag, i;
struct node
{
	int data;
	struct node* leftChild, *rightChild;
};
typedef struct node *NODE;
NODE createnode(int item)
{
	NODE temp;
	temp = (NODE)malloc(sizeof(struct node));
	temp->data = item;
	temp->leftChild = NULL;
	temp->rightChild = NULL;
	return temp;
}
void insertBST(NODE root, NODE newNode)
{
	if(newNode->data == root->data)
	{
		printf("Key already exists\n");
		i--;
		return;
	}
	else if (newNode->data < root->data)
	{
		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, int key)
{
        if(!root)
                return -1;
        if(key == root->data)
                return 1;
        if(key < root->data)
                return search(root->leftChild, key);
        else
                return search(root->rightChild,key);
}
void inorder(NODE temp)
{
        if (temp != NULL)
        {
                inorder(temp->leftChild);
                printf("%d\t", temp->data);
                inorder(temp->rightChild);
        }
}
void preorder(NODE temp)
{
        if (temp != NULL)
        {
                printf("%d\t", temp->data);
                preorder(temp->leftChild);
                preorder(temp->rightChild);
        }
}
void postorder(NODE temp)
{
        if (temp != NULL)
        {
                postorder(temp->leftChild);
                postorder(temp->rightChild);
                printf("%d\t", temp->data);
        }
}
```

```
main()
{
	int choice,n,item;
	int key,keyFound = 0;
	NODE root=NULL,newNode;
	while(1)
	{
		choice=0;
		printf("\n-----------------MENU----------------------\n");
		printf("1. Create a BST\n");
		printf("2. Traverse a BST\n");
		printf("3. Search a BST\n");
		printf("4. Exit\n");
		printf("------------------------------------------\n");
		printf("Enter choice : ");
		scanf("%d", &choice);
		switch(choice)
		{
			case 1: root= NULL;
				printf("Enter the number of elements in the BST:");
				scanf("%d",&n);
				for(i=0;i<n;i++)
				{
					printf("Enter the integer:");
					scanf("%d",&item);
					newNode = createnode(item);
					if(root == NULL)
						root = newNode;
					else
						insertBST(root,newNode);
				}
				break;
			case 2: if (root == NULL)
				{
					printf("Tree is empty\n");
					break;
				}
				else
				{
					printf("BST Preorder travsersal\n");
					preorder(root);
					printf("\nBST Inorder travsersal\n");
					inorder(root);
					printf("\nBST Postorder travsersal\n");
					postorder(root);
```

```
                                        break;
                                }
                        case 3: printf("Enter the search key:");
                                scanf("%d",&key);
                                keyFound = search(root,key);
                                if(keyFound == 1)
                                        printf("Element %d is found in the BST",key);
                                else
                                        printf("Element %d is not found in the BST",key);
                                break;
                        case 4: return;
                        default:printf("Wrong choice\n");
                                return;
                }
        }
}
```

Output

```
-----------------MENU----------------------
 1 - Create a BST
 2 - Traverse a BST
 3 - Search a BST
 4 - Exit
-------------------------------------------
Enter choice : 1
Enter the number of elements in the BST:12
Enter the integer:6
Enter the integer:9
Enter the integer:5
Enter the integer:2
Enter the integer:8
Enter the integer:15
Enter the integer:24
Enter the integer:14
Enter the integer:7
Enter the integer:8
Key already exists
Enter the integer:5
Key already exists
Enter the integer:2
Key already exists

Enter the integer:65
Enter the integer:23
Enter the integer:11
```

```
-----------------MENU----------------------
 1 - Create a BST
 2 - Traverse a BST
 3 - Search a BST
 4 - Exit
-------------------------------------------
Enter choice : 2
BST Preorder travsersal
6       5       2       9       8       7       15      14      11      24      23      65
BST Inorder travsersal
2       5       6       7       8       9       11      14      15      23      24      65
BST Postorder travsersal
2       5       7       8       11      14      23      65      24      15      9       6
-----------------MENU----------------------
 1 - Create a BST
 2 - Traverse a BST
 3 - Search a BST
 4 - Exit
-------------------------------------------
Enter choice : 3
Enter the search key:11
Element 11 is found in the BST
-----------------MENU----------------------
 1 - Create a BST
 2 - Traverse a BST
 3 - Search a BST
 4 - Exit
-------------------------------------------
Enter choice : 3
Enter the search key:90
Element 90 is not found in the BST
-----------------MENU----------------------
 1 - Create a BST
 2 - Traverse a BST
 3 - Search a BST
 4 - Exit
-------------------------------------------
Enter 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>
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);
                }
        }
}
```

```
main()
{
        int v;
        printf("Enter the number of vertices: ");
        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:\n");
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                        scanf("%d",&a[i][j]);
        printf("Enter the starting vertex: ");
        scanf("%d",&v);
        if((v<1)||(v>n))
        {
                printf("Invalid vertex\n");
                return;
        }
        printf("Order of BFS Travles\n %d", v);
        bfs(v);
        printf("\nOrder of DFS Travles\n %d", v);
        dfs(v);
        printf("\n");
}
```

<u>Outputs:</u>

```
Enter the number of vertices:9
Enter graph data in matrix form:
0 1 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 1 1
0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
Enter the starting vertex:1
Order of BFS Travles
 1->2->4->3->5->8->9->6->7
Order of DFS Travles
 1->2->3->4->5->6->7->8->9
```

……

```
Enter the number of vertices:5
Enter graph data in matrix form:
0 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
Enter the starting vertex:1
Order of BFS Travles
 1->2->3->4
Order of DFS Travles
 1->2->4->3
```

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

Outputs:

```
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
```