S.No: 1	Exp. Name: ***C Program to implement initialization of array and perform traversal operations in both the directions.***	**Date: 2023-03-23**

ID: B22AI012

Kakatiya Institute of Technology and Science

Aim:

C Program to implement initialization of array and perform traversal operations in both the directions.

Problem description: In this program we should declare an array, read the values from the user and print them in both the directions i.e from index 0 to n-1 and n-1 to 0.

Example:
Input:
Enter how many values you want to read : 5
Enter the value of a[0] : 23
Enter the value of a[1] : 63
Enter the value of a[2] : 45
Enter the value of a[3] : 12
Enter the value of a[4] : 20
Expected Output:
The given array elements in forward direction are: [23 63 45 12 20]
The given array elements in backward direction are: [20 12 45 63 23]

Source Code:

array.c

```
#include<stdio.h>
int main()
{
        int a[100],i,n;
        printf("Enter how many values you want to read : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter the value of a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("The array elements are : ");
        for(i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
        printf("\n");
        printf("Elements in backward direction: ");
        for(i=n-1;i>=0;i--)
        {
                printf("%d ",a[i]);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Enter how many values you want to read :
5
Enter the value of a[0] :
23
Enter the value of a[1] :
63
Enter the value of a[2] :
45
Enter the value of a[3] :
12
Enter the value of a[4] :
20
The array elements are : 23 63 45 12 20
Elements in backward direction: 20 12 45 63 23

Test Case - 2
User Output
Enter how many values you want to read :
5
Enter the value of a[0] :
96
Enter the value of a[1] :
12
Enter the value of a[2] :
45
Enter the value of a[3] :
78
Enter the value of a[4] :
30
The array elements are : 96 12 45 78 30
Elements in backward direction: 30 78 45 12 96

S.No: 2	Exp. Name: ***C Program to implement searching operation on array using Linear Search.***	**Date: 2023-03-23**

Aim:

C Program to implement searching operation on array using **Linear Search.**

Problem description: After reading the array values, and a key element from the user, we need to check whether a given key element is existing in that array or not.

Example:
Input:
Enter the size of the array : 5
Enter element for a[0] : 45
Enter element for a[1] : 67
Enter element for a[2] : 35
Enter element for a[3] : 28
Enter element for a[4] : 16
Enter key element : 28
Expected Output:
The key element 28 is found at the position 3

If key element is not found in the array following message should be displayed (assume key element is 28)
"The key element 28 is not found in the array\n"

Source Code:

LinearSearch.c

```
#include<stdio.h>
int main()
{
        int a[100],i,n,key,found=0;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Enter key element : ");
        scanf("%d",&key);
    for(i=0;i<n;i++)
    {
        if(a[i]==key)
        {
            found=1;
            break;
        }
    }
    if(found==1)
    {
        printf("The key element %d is found at the position %d\n",key,i);
    }
    else
    {
        printf("The key element %d is not found in the array\n",key);
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
5
Enter element for a[0] :
45
Enter element for a[1] :
67
Enter element for a[2] :
35
Enter element for a[3] :
28
Enter element for a[4] :
16
Enter key element :
28
The key element 28 is found at the position 3

Test Case - 2
User Output
Enter value of n :
5
Enter element for a[0] :
2
Enter element for a[1] :
7
Enter element for a[2] :
5
Enter element for a[3] :
1
Enter element for a[4] :
4
Enter key element :
2
The key element 2 is found at the position 0

S.No: 3	Exp. Name: ***C Program to display the count of occurrences of every number in an array.***	**Date: 2023-03-31**

Aim:

C Program to display the count of occurrences of every number in an array.

Problem description: In this program after reading the array values, the user should count the occurrences of every number in the given array and should display them.

Example:

Input:

Enter the size of array: 6
Enter 6 Elements:
5
3
2
3
4
2

Expected Output:

5 occurs 1 times
3 occurs 2 times
2 occurs 2 times
4 occurs 1 times

Source Code:

countArray.c

```c
#include<stdio.h>
int main()
{
        int a[100],f[100];
        int size,i,j,count=1;
        printf("Enter the size of array (n<=10): ");
        scanf("%d",&size);
        printf("Enter %d Elements:",size);
        for(i=0;i<size;i++)
        {
                scanf("%d",&a[i]);
                f[i]=count;
        }
        for(i=0;i<size;i++)
        {
                count=1;
                for(j=i+1;j<size;j++)
                {
                        if(a[i]==a[j])
                        {
                                count++;
                                f[j]=0;
                        }
                }
                if(f[i]!=0)
                {
                        f[i]=count;
                }
        }
        for(i=0;i<size;i++)
        {
                if(f[i]!=0)
                {
                        printf("%d occurs %d times \n",a[i],f[i]);
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the size of array (n<=10):
6
Enter 6 Elements:
5
3
2
3
4
2

5 occurs 1 times
3 occurs 2 times
2 occurs 2 times
4 occurs 1 times

Test Case - 2
User Output
Enter the size of array (n<=10):
4
Enter 4 Elements:
1
2
3
4
1 occurs 1 times
2 occurs 1 times
3 occurs 1 times
4 occurs 1 times

S.No: 4	Exp. Name: ***C Program to implement searching operation on array using Binary Search.***	**Date: 2023-04-06**

Aim:

C Program to implement searching operation on array using Binary Search.

Source Code:

`BinarySearch.c`

```
#include<stdio.h>
int main()
{
        int a[20],i,n,key,flag=0,temp,j;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
        printf("Enter element for a[%d] : ",i);
        scanf("%d",&a[i]);
        }
        printf("Enter key element : ");
        scanf("%d",&key);
        for(i=0;i<n;i++)
        {
                for(j=i;j<n;j++)
                {
                   if(a[i]>a[j])
                   {
                      temp=a[i];
                      a[i]=a[j];
                      a[j]=temp;
                   }
                }
        }
        printf("After sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
        printf("Value of a[%d] = %d\n",i,a[i]);
        }
        for(i=0;i<n;i++)
        {
                if(a[i]==key)
                {
                        flag=1;
                        break;
                }
        }
        if(flag==1)
        {
        printf("The key element %d is found at the position %d\n",key,i);
        }
        else
        {
            printf("The Key element %d is not found in the array\n",key);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :

5
Enter element for a[0] :
4
Enter element for a[1] :
8
Enter element for a[2] :
6
Enter element for a[3] :
2
Enter element for a[4] :
1
Enter key element :
8
After sorting the elements in the array are
Value of a[0] = 1
Value of a[1] = 2
Value of a[2] = 4
Value of a[3] = 6
Value of a[4] = 8
The key element 8 is found at the position 4

Test Case - 2
User Output
Enter value of n :
7
Enter element for a[0] :
56
Enter element for a[1] :
89
Enter element for a[2] :
63
Enter element for a[3] :
215
Enter element for a[4] :
325
Enter element for a[5] :
156
Enter element for a[6] :
256
Enter key element :
458
After sorting the elements in the array are
Value of a[0] = 56
Value of a[1] = 63
Value of a[2] = 89
Value of a[3] = 156
Value of a[4] = 215
Value of a[5] = 256

```
Value of a[6] = 325
The Key element 458 is not found in the array
```

S.No: 5	Exp. Name: ***Write a C program to insert an element in an array.***	Date: 2023-04-01

Aim:

Write a C program to implement insertion operation on the array.

Source Code:

insertElement.c

```
#include<stdio.h>
int main()
{
        int a[20],i,x,n,pos;
        printf("Enter no of elements in array: ");
        scanf("%d",&n);
        printf("Enter %d elements: ",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("Enter the position where you want to insert: ");
        scanf("%d",&pos);
        printf("Enter the value into that position: ");
        scanf("%d",&x);
        for(i=n;i>pos-2;i--)
        {
                a[i]=a[i-1];
        }
        a[pos-1]=x;
        printf("Final array is: ");
        for(i=0;i<=n;i++)
        {
                printf("%d ",a[i]);
        }
        printf("\n");

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter no of elements in array:
3
Enter 3 elements:
1 2 3
Enter the position where you want to insert:
2
Enter the value into that position:
63

Final array is: 1 63 2 3

Test Case - 2
User Output
Enter no of elements in array:
7
Enter 7 elements:
25 96 48 36 22 536 425
Enter the position where you want to insert:
5
Enter the value into that position:
565
Final array is: 25 96 48 36 565 22 536 425

S.No: 6	Exp. Name: ***Write a program to delete an element***	Date: 2023-04-06

Aim:

Write a program to implement deletion operations on array.

Source Code:

deleteElement.c

```
#include<stdio.h>
int main()
{
        int a[20],n,i,pos;
        printf("Enter number of elements in array: ");
        scanf("%d",&n);
        printf("Enter %d elements\n",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("Enter the location where you wish to delete element: ");
        scanf("%d",&pos);
        if(pos<=n)
        {
                for(i=pos-1;i<n;i++)
                {
                        a[i]=a[i+1];
                }
                printf("Resultant array is\n");
                for(i=0;i<n-1;i++)
                {
                        printf("%d\n",a[i]);
                }
        }
        else
        {
                printf("Deletion not possible.\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of elements in array:
5
Enter 5 elements
6
3
2
4

ID: B22AI012

Kakatiya Institute of Technology and Science

5
Enter the location where you wish to delete element:
2
Resultant array is
6
2
4
5

Test Case - 2
User Output
Enter number of elements in array:
3
Enter 3 elements
33
44
55
Enter the location where you wish to delete element:
4
Deletion not possible.

S.No: 7	Exp. Name: ***Write a program to represent and display sparse matrix.***	Date: 2023-04-10

Aim:

Write a program to represent and display sparse matrix.

Source Code:

sparseMatrix.c

```
#include<stdio.h>
#define MAX 20
void read_matrix(int a[10][10],int row,int column);
void print_sparse(int b[MAX][3]);
void create_sparse(int a[10][10],int row,int column,int b[MAX][3]);
int main()
{
        int a[10][10],b[MAX][3],row,column;
    printf("Enter the size of matrix (rows, columns): ");
    scanf("%d %d",&row,&column);
    read_matrix(a,row,column);

    create_sparse(a,row,column,b);
    print_sparse(b);
    return 0;
}
void read_matrix(int a[10][10],int row,int column)
{
        int i,j;
        printf("Enter elements of matrix\n");
        for(i=0;i<row;i++)
        {
                for(j=0;j<column;j++)
                {
                        printf("[%d][%d]: ",i,j);
                        scanf("%d",&a[i][j]);
                }
        }
}
void create_sparse(int a[10][10],int row,int column,int b[MAX][3])
{
        int i,j,k;
        k=1;
        b[0][0]=row;
        b[0][1]=column;
        for(i=0;i<row;i++)
        {
                for(j=0;j<column;j++)
                {
                        if(a[i][j]!=0)
                        {
                                b[k][0]=i;
                                b[k][1]=j;
                                b[k][2]=a[i][j];
                                k++;
                        }
                }
                b[0][2]=k-1;
        }
}
void print_sparse(int b[MAX][3])
{
        int i,column;
        column=b[0][2];
        printf("Sparse form - list of 3 triples\n");
```

```
                printf("%d\t%d\t%d\n",b[i][0],b[i][1],b[i][2]);
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the size of matrix (rows, columns):
2 2
Enter elements of matrix
[0][0]:
0
[0][1]:
0
[1][0]:
1
[1][1]:
2
Sparse form - list of 3 triples
2 2 2
1 0 1
1 1 2

Test Case - 2
User Output
Enter the size of matrix (rows, columns):
3 3
Enter elements of matrix
[0][0]:
0
[0][1]:
0
[0][2]:
3
[1][0]:
0
[1][1]:
4
[1][2]:
0
[2][0]:
0
[2][1]:

```
5
[2][2]:
7
Sparse form - list of 3 triples
3       3       4
0       2       3
1       1       4
2       1       5
2       2       7
```

S.No: 8	Exp. Name: ***Write a program to implement initialization of arrays and traversal operation with DMA***	Date: 2023-04-13

Aim:

Write a program to implement initialization of arrays and traversal operation with DMA

Source Code:

dmaArray.c

```
#include<stdio.h>
int main()
{
        int *a,n,i;
        printf("Enter number of elements:\n");
        scanf("%d",&n);
        a=(int*)malloc(n*sizeof(int));
        printf("Enter %d elements into array:",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("Elements of array are as follow:\n");
        for(i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of elements:
4
Enter 4 elements into array:
1
-3
4
5
Elements of array are as follow:
1 -3 4 5

Test Case - 2
User Output
Enter number of elements:
6
Enter 6 elements into array:

11
22
369
-456
258
145
Elements of array are as follow:
11 22 369 -456 258 145

S.No: 9	Exp. Name: ***Write a program to implement matrix addition and subtraction with DMA.***	Date: 2023-04-12

Aim:

Write a program to implement matrix addition and subtraction with DMA.

Source Code:

matrix.c

```c
#include<stdio.h>
int main( )
{
        int p,q,a[20][20],b[20][20],i,j;
        printf("Enter the order of the matrix 1 & 2: \n");
        scanf("%d %d",&p,&q);
        printf("Enter values into 2D array of rows=%d, cols=%d\n",p,q);
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        scanf("%d",&a[i][j]);
                }
        }
        printf("Enter values into 2D array of rows=%d, cols=%d\n",p,q);
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        scanf("%d",&b[i][j]);
                }
        }
        printf("Matrix-1 is as follow:\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        printf("%d ",a[i][j]);
                }printf("\n");
        }
        printf("Matrix-2 is as follow:\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        printf("%d ",b[i][j]);
                }printf("\n");
        }
        printf("Resultant Sum of 2 Matrices is as follow:\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        printf("%d ",a[i][j]+b[i][j]);
                }printf("\n");
        }
        printf("Resultant difference of 2 Matrices is as follow:\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                {
                        printf("%d ",a[i][j]-b[i][j]);
                }printf("\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the order of the matrix 1 & 2:
2 2
Enter values into 2D array of rows=2, cols=2
3 6
7 8
Enter values into 2D array of rows=2, cols=2
4 5
1 2
Matrix-1 is as follow:
3 6
7 8
Matrix-2 is as follow:
4 5
1 2
Resultant Sum of 2 Matrices is as follow:
7 11
8 10
Resultant difference of 2 Matrices is as follow:
-1 1
6 6

Test Case - 2
User Output
Enter the order of the matrix 1 & 2:
2 1
Enter values into 2D array of rows=2, cols=1
3
6
Enter values into 2D array of rows=2, cols=1
4
5
Matrix-1 is as follow:
3
6
Matrix-2 is as follow:
4
5
Resultant Sum of 2 Matrices is as follow:
7
11
Resultant difference of 2 Matrices is as follow:
-1
1

Test Case - 3
User Output
Enter the order of the matrix 1 & 2:
1 2
Enter values into 2D array of rows=1, cols=2
4 5
Enter values into 2D array of rows=1, cols=2
-4 -5
Matrix-1 is as follow:
4 5
Matrix-2 is as follow:
-4 -5
Resultant Sum of 2 Matrices is as follow:
0 0
Resultant difference of 2 Matrices is as follow:
8 10

S.No: 10	Exp. Name: ***Write a Program to convert Infix expression to Postfix expression.***	Date: 2023-04-20

Aim:

Write a Program to convert Infix expression to Postfix expression.

Constraint: Only alphanumeric characters parenthesis and valid arithmetic operators (+, -, *, % and /) are allowed

Source Code:

Infix2PostfixMain.c

```
#include "Infix2PostfixOperation.c"
int main() {
        char exp[20];
        char *e, x;
        printf("Enter the Infix expression : ");
        scanf("%s",exp);
        e = exp;
        convertInfix(e);
}
```

Infix2PostfixOperation.c

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include<stdbool.h>
#define MAX 20
char stack[MAX];
int top=-1;
char stack[MAX];
int isEmpty()
{
        if(top<0)
        return 1;
        else
        return 0;
}
void push(char x)
{
        if(top==MAX-1)
        {
        printf("Stack is overflow:\n");
        }
        else
        {
                top=top+1;
                stack[top]=x;
        }
}
char pop()
{
        if(top<0)
        {
                printf("Stack is underflow unbalanced parenthesis\n");
                exit(0);
        }
        else
        return stack[top--];
}
int priority(char x)
{
        if(x=='(')
        return 0;
        if((x=='+')||(x=='-'))
        return 1;
        if(x=='*'||x=='/'||x=='%')
        return 2;
}
bool isalphanum(char a)
{
        if((a>=65&&a<=90)||(a>=97&&a<=122))
        return true;
        return false;
}
void convertInfix(char *e)
{
```

```
        char *p=(char*)malloc(sizeof(char)*strlen(e));
        while(*e!='\0')
        {
                if(isalphanum(*e))
                p[k++]=*e;
                else if(*e=='(')
                push(*e);
                else if(*e==')')

                {
                        while(!isEmpty()&&(x=pop())!='(')
                        p[k++]=x;
                }

                else if(*e=='+'||*e=='-'||*e=='*'||*e=='/'||*e=='%')
                {
                        while(priority(stack[top])>=priority(*e))
                        p[k++]=pop();
                        push(*e);
                }
                else
                {
                        printf("Invalid symbols in infix expression. Only alphanumeric
characters  parenthesis and valid arithmetic operators are allowed.\n");
                        exit(0);
                }
            e++;
        }
        while(top!=-1)
        {
                x=pop();
                if(x=='(')
                {
            printf("Invalid infix expression : unbalanced parenthesis.\n");
            exit(0);
                }
            p[k++]=x;
    }
    p[k++]='\0';
    printf("Postfix expression : %s\n",p);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the Infix expression :
a@b
Invalid symbols in infix expression. Only alphanumeric characters parenthesis and valid arithmetic operators are allowed.

Test Case - 2
User Output
Enter the Infix expression :
(a+b)/(c-d)
Postfix expression : ab+cd-/

Test Case - 3
User Output
Enter the Infix expression :
(a+b+c)*((a+b)*(b+c)
Invalid infix expression : unbalanced parenthesis.

S.No: 11	Exp. Name: ***Write a program to implement Stack Operations using arrays.***	Date: 2023-04-12

Aim:

Write a C program to implement Stack Operations using arrays.

a. Push

b. Pop

c. Display

Source Code:

StackOperations.c

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int top = -1;
int stack_arr[MAX];
void push()
{
    int el;
    printf("Enter item to insert: ");
    scanf("%d",&el);
    top=top+1;
    stack_arr[top]=el;
    printf("Item inserted = %d\n",el);
}
int pop()
{
        int el;
        if(top==-1)
        {
                printf("Stack underflow\n");
                return -1;
        }
        el=stack_arr[top];
        top=top-1;
        return el;
}
void display()
{
        int i;
        if(top==-1)
        {
                printf("No more data\n");
                return;
        }
        printf("Stack are\n");
        for(i=top;i>=0;i--)
        printf("%d ",stack_arr[i]);
        printf("\n");
}
main()
{
        int choice,el;
        printf("Enter the choice for stack operation: \n");
        printf("press 1 for push\n");
        printf("press 2 for pop\n");
        printf("press 3 for display\n");
        printf("press 4 for exit\n");
        while(1)
        {
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: push();
                        break;
                        case 2: el=pop();
```

```
                        break;
                        case 3:display();
                        break;
                        case 4:exit(1);
                        default: printf("wrong input");
                }

                printf("Enter the choice for stack operation: ");
        }

}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the choice for stack operation:
press 1 for push
press 2 for pop
press 3 for display
press 4 for exit
2
Stack underflow
Enter the choice for stack operation:
1
Enter item to insert:
4
Item inserted = 4
Enter the choice for stack operation:
2
Item deleted = 4
Enter the choice for stack operation:
2
Stack underflow
Enter the choice for stack operation:
4

Test Case - 2
User Output
Enter the choice for stack operation:
press 1 for push
press 2 for pop
press 3 for display
press 4 for exit
2
Stack underflow
Enter the choice for stack operation:

```
1
Enter item to insert:
36
Item inserted = 36
Enter the choice for stack operation:
1
Enter item to insert:
45
Item inserted = 45
Enter the choice for stack operation:
1
Enter item to insert:
85
Item inserted = 85
Enter the choice for stack operation:
3
Stack are
85 45 36
Enter the choice for stack operation:
2
Item deleted = 85
Enter the choice for stack operation:
2
Item deleted = 45
Enter the choice for stack operation:
3
Stack are
36
Enter the choice for stack operation:
2
Item deleted = 36
Enter the choice for stack operation:
3
No more data
Enter the choice for stack operation:
2
Stack underflow
Enter the choice for stack operation:
3
No more data
Enter the choice for stack operation:
4
```

S.No: 12	Exp. Name: ***Write a C program to find the Multiplication of two matrices using Pointers and DMA***	Date: 2023-04-13

Aim:

Write a program to find the `multiplication` of two matrices using **pointers**.

At the time of execution, the program should print the message on the console as:

```
Enter the size of the first matrix :
```

For example, if the user gives the **input** as:

```
Enter the size of the first matrix : 2 2
```

Next, the program should print the message on the console as:

```
Enter 4 elements :
```

if the user gives the **input** as:

```
Enter 4 elements : 1 2 3 4
```

Next, the program should print the message on the console as:

```
Enter the size of the second matrix :
```

if the user gives the **input** as:

```
Enter the size of the second matrix : 2 2
```

Next, the program should print the message on the console as:

```
Enter 4 elements :
```

if the user gives the **input** as:

```
Enter 4 elements : 5 6 7 8
```

then the program should **print** the result as:

```
The first matrix is
1 2
3 4
The second matrix is
5 6
7 8
The Multiplication Matrix is
19 22
43 50
```

Note: Write the functions **read()**, **display()** and **multiplicationOfTwoMatrices()** in `Program1008a.c`.

Source Code:

`Program1008.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include "Program1008a.c"
void main() {
        int *a, *b, m, n, p, q;
        printf("Enter the size of the first matrix : ");
        scanf("%d%d", &m, &n);
        a = (int *) calloc(m * n, sizeof(int));
        read(a, m, n);
        printf("Enter the size of the second matrix : ");
        scanf("%d%d", &p, &q);
        b = (int *) calloc(p * q, sizeof(int));
        read(b, p, q);
        printf("The first matrix is\n");
        display(a, m, n);
        printf("The second matrix is\n");
        display(b, p, q);
        if (n == p) {
                multiplicationOfTwoMatrices(a, b, m, n, q);
        } else {
                printf("Multiplication is not possible\n");
        }
}
```

Program1008a.c

```
void read(int*p,int m,int n);
void display(int *p,int m,int n);
void multiplicationOfTwoMatrices(int *t,int *s,int m,int n,int p);
void read(int *p,int m,int n)
{
        int i,j;
        printf("Enter %d elements : ",m*n);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d",(p+i*n+j));
                }
        }
}
void display(int *p,int m,int n)
{
        int i,j;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d ",*(p+i*n+j));
                }printf("\n");
        }
}
void multiplicationOfTwoMatrices(int *t,int*s,int m,int n,int p)
{
        int i,j,k,sum,*z;
        z=(int*)malloc(m*p*sizeof(int));
        for(i=0;i<m;i++)
        {
                for(j=0;j<p;j++)
                {
                        sum=0;
                        for(k=0;k<n;k++)
                        {
                                sum=sum+(*(t+i*n+k))*(*(s+k*p+j));
                        }
                *(z+i*p+j)=sum;
                }
        }
        printf("The Multiplication Matrix is\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<p;j++)
                {
                        printf("%d ",*(z+i*p+j));
                }
                printf("\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the size of the first matrix :
2 3
Enter 6 elements :
1 2 3 4 5 6
Enter the size of the second matrix :
2 2
Enter 4 elements :
1 2 3 4
The first matrix is
1 2 3
4 5 6
The second matrix is
1 2
3 4
Multiplication is not possible

Test Case - 2
User Output
Enter the size of the first matrix :
2 2
Enter 4 elements :
1 2 3 4
Enter the size of the second matrix :
2 2
Enter 4 elements :
3 4 5 6
The first matrix is
1 2
3 4
The second matrix is
3 4
5 6
The Multiplication Matrix is
13 16
29 36

S.No: 13	Exp. Name: ***C program to evaluate given postfix expression.***	**Date: 2023-04-24**

Aim:
C Program to evaluate given postfix expression.

Problem Description: Postfix notation is one of the few ways to represent algebraic expressions. It is used in computers because it is faster than other types of notations (such as infix notation) as parentheses are not required to represent them. As the name suggests, in the postfix expression operators follow their operands. Therefore, the process of postfix evaluation is quite different than solving the infix notation (normal notation used in daily use).

Input - A string consisting of numeric values along with operator symbols (*, /, +, -, etc) is given as the input.
Output - The result obtained by solving the given input is required to be printed.

Example 1:-
Input - 23*45+*
Output - 54
Source Code:

postfix.c

```
#include<stdio.h>
#include<ctype.h>
int stack[20];
int top=-1;
void push(int x)
{
        stack[++top]=x;
}
int pop()
{
        return stack[top--];
}
int main()
{
        char exp[20];
        char *e;
        int n1,n2,n3,num;
        printf("Enter the expression : ");
        scanf("%s",exp);
        e=exp;
        while(*e!='\0')
        {
                if(isdigit(*e))
                {
                        num=*e-48;
                        push(num);
                }
                else
                {
                        n1=pop();
                        n2=pop();
                        switch(*e)
                        {
                                case '+':
                                {
                                        n3=n1+n2;
                                        break;
                                }
                                case '-':
                                {
                                        n3=n2-n1;
                                        break;
                                }
                                case '*':
                                {
                                        n3=n1*n2;
                                        break;
                                }
                                case '/':
                                {
                                        n3=n2/n1;
                                        break;
                                }
                        }
                        push(n3);
```

```
        }
        printf("The result of expression %s = %d\n\n",exp,pop());
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the expression :
245+*
The result of expression 245+* = 18

Test Case - 2
User Output
Enter the expression :
231*+9-
The result of expression 231*+9- = -4

Test Case - 3
User Output
Enter the expression :
523**
The result of expression 523** = 30

S.No: 14	Exp. Name: ***C Program to define a recursive function to solve the Tower of Hanoi puzzle***	**Date: 2023-04-20**

Aim:

C Program to define a recursive function to solve the Tower of Hanoi puzzle

Problem Description: Tower of Hanoi is a mathematical puzzle where we have three rods (**A, B**, and**C**) and**N**disks. Initially, all the disks are stacked in decreasing value of diameter i.e., the smallest disk is placed on the top and they are on rod**A**. The objective of the puzzle is to move the entire stack to another rod (here considered**C**), obeying the following simple rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.

Example-1:
Input:
Enter number of disks : 3
Output:
Move disk - 1 from pole A to C
Move disk - 2 from pole A to B
Move disk - 1 from pole C to B
Move disk - 3 from pole A to C
Move disk - 1 from pole B to A
Move disk - 2 from pole B to C
Move disk - 1 from pole A to C

Source Code:

Hanoi.c

```
#include<stdio.h>
void towerofHanoi(int n,char from_rod,char to_rod,char aux_rod)
{
        if(n==1)
        {
                printf("Move disk - 1 from pole %c to %c",from_rod,to_rod);
                return;
        }
        towerofHanoi(n-1,from_rod,aux_rod,to_rod);
        printf("\nMove disk - %d from pole %c to %c\n",n,from_rod,to_rod);
        towerofHanoi(n-1,aux_rod,to_rod,from_rod);
}
int main()
{
        int n;
        printf("Enter number of disks : ");
        scanf("%d",&n);
        towerofHanoi(n,'A','C','B');
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of disks :
2
Move disk - 1 from pole A to B
Move disk - 2 from pole A to C
Move disk - 1 from pole B to C

Test Case - 2
User Output
Enter number of disks :
3
Move disk - 1 from pole A to C
Move disk - 2 from pole A to B
Move disk - 1 from pole C to B
Move disk - 3 from pole A to C
Move disk - 1 from pole B to A
Move disk - 2 from pole B to C
Move disk - 1 from pole A to C

S.No: 15	Exp. Name: ***C Program to display the Fibonacci series with the help of recursive function***	**Date: 2023-04-20**

Aim:

C Program to display the Fibonacci series with the help of a recursive function

Problem Description: The Fibonacci numbers are the numbers in the following integer sequence. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

By definition, the first two numbers are 0 and 1. And each subsequent number in the series is equal to the sum of the previous two numbers.

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation

Fn = Fn-1 + Fn-2 with seed values F0 = 0 and F1 = 1.

Example-1:

Input:

Enter the Total terms: 8

Output:

The Fibonacci series of 8 terms are : 0 1 1 2 3 5 8 13

Example-2:

Input:

Enter the Total terms: 6

Output:

The Fibonacci series of 8 terms are : 0 1 1 2 3 5

Source Code:

fibonacci.c

```
#include<stdio.h>
int fibonacci(int);
int main()
{
        int n,m=0,i;
        printf("Enter the Total terms : ");
        scanf("%d",&n);
        printf("The Fibonacci series of %d terms are :",n);
        for(i=1;i<=n;i++)
        {
                printf(" %d",fibonacci(m));
                m++;
        }printf(" ");
        return 0;
}
int fibonacci(int n)
{
        if(n==0||n==1)
        return n;
        else
        return(fibonacci(n-1)+fibonacci(n-2));
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the Total terms :
3
The Fibonacci series of 3 terms are : 0 1 1

Test Case - 2
User Output
Enter the Total terms :
5
The Fibonacci series of 5 terms are : 0 1 1 2 3

Test Case - 3
User Output
Enter the Total terms :
6
The Fibonacci series of 6 terms are : 0 1 1 2 3 5

S.No: 16	Exp. Name: ***C Program to implement Multi Stack***	Date: 2023-05-03

Aim:

Write a **C** Program to implement Multi Stack.

1. Push
2. Pop
3. Display

Source Code:

`MultiStack.c`

```
#include<stdio.h>
#include<malloc.h>
#define MAX 10
int stack[MAX],topA=-1,topB=MAX;
void pushA(int val)
{
        if(topA==topB-1)
        {
                printf("Overflow\n");
        }
        else
        {
                topA+=1;
                stack[topA]=val;
        }
}
int popA()
{
        int val;
        if(topA==-1)
        {
                printf("Stack 1 is empty\n");
                val=-999;
        }
        else
        {
                val=stack[topA];
                topA--;
        }
        return val;
}
void display_stackA()
{
        int i;
        if(topA==-1)
        {
                printf("\n");
        }
        else
        {
                for(i=topA;i>=0;i--)
                printf("%d ",stack[i]);
                printf("\n");
        }
}
void pushB(int val)
{
        if(topB-1 == topA)
        {
                printf("Overflow\n");
        }
        else
        {
                topB-=1;
                stack[topB]=val;
```

```
  int popB()
   {
                int val;
                if(topB==MAX)
                {
                        printf("Stack 2 is empty\n");
                        val = -999;
                }
                else
                {
                        val=stack[topB];
                        topB++;
                }
                return val;
        }
void display_stackB()
{
        int i;
        if(topB==MAX)
        {
                printf("\n");
        }
        else
        {
                for(i=topB;i<MAX;i++)
                printf("%d ",stack[i]);
                printf("\n");
        }
}
int main()
{
        int option,val,sno;
        while(1)
        {

                printf("1. Push\n2. Pop\n3. Display\n4. Quit\n");
                printf("Enter your choice: ");
                scanf("%d",&option);
                switch(option)
                {
                        case 1:
                        {
                        printf("Enter stack number (1 or 2): ");
                        scanf("%d",&sno);
                        printf("Enter data to be pushed: ");
                        scanf("%d",&val);
                        if(sno==1)
                        pushA(val);
                        else
                        pushB(val);
                        break;
                        }
                        case 2:
                        {
                        printf("Enter stack number (1 or 2): ");
```

```
                        {
                                val=popA();
                                if(val !=-999)
                                printf("Popped element: %d\n",val);
                        }
                        else
                        {
                                val=popB();
                                if(val!=-999)
                                printf("Popped element: %d\n",val);
                        }
                        break;
                    }
                        case 3:
                        {
                          printf("Stack 1: ");
                          display_stackA();
                          printf("Stack 2: ");
                          display_stackB();
                          break;
                        }
                        case 4:
                        exit(0);
                    }
            }
    }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
1
Enter stack number (1 or 2):
1
Enter data to be pushed:
10
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
1
Enter stack number (1 or 2):
2
Enter data to be pushed:

```
20
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
1
Enter stack number (1 or 2):
1
Enter data to be pushed:
16
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
1
Enter stack number (1 or 2):
2
Enter data to be pushed:
45
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
3
Stack 1: 16 10
Stack 2: 45 20
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
2
Enter stack number (1 or 2):
1
Popped element: 16
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
2
Enter stack number (1 or 2):
2
Popped element: 45
1. Push
2. Pop
3. Display
```

4. Quit
Enter your choice:
3
Stack 1: 10
Stack 2: 20
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
4

Test Case - 2
User Output
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
2
Enter stack number (1 or 2):
1
Stack 1 is empty
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
1
Enter stack number (1 or 2):
2
Enter data to be pushed:
10
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
2
Enter stack number (1 or 2):
2
Popped element: 10
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
2
Enter stack number (1 or 2):

```
2
Stack 2 is empty
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
3
Stack 1:
Stack 2:
1. Push
2. Pop
3. Display
4. Quit
Enter your choice:
4
```

S.No: 17	Exp. Name: ***C Program to implement queue operations using arrays***	Date: 2023-04-27

Aim:

Write a C Program to implement queue operations using arrays.

- Enqueue
- Dequeue
- Display
- Size
- Check if the queue is **Empty** or not.

Source Code:

QueueUsingArray.c

```
#include <conio.h>
#include <stdio.h>
#include "QueueOperations.c"
int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

QueueOperations.c

```
#define MAX 5
void enqueue(int);
int dequeue();
void display();
void isEmpty();
void size();
int Queue[MAX];
int rear=-1,front=-1,cnt=0;
void enqueue(int x)
{
        if(rear==MAX-1)
        {
                printf("Queue is not empty");
        }
        else if(front==-1)
        {
        front=0;
        rear=0;
        Queue[rear]=x;
        cnt++;
        printf("Successfully inserted.\n");
        }
        else
        {
                rear++;
                Queue[rear]=x;
                cnt++;
                printf("Successfully inserted.\n");
        }
}
int dequeue()
{
        int x;
        if(front==-1)
        {
                printf("Queue is underflow.\n");
        }
        else
        {
                x=Queue[front];
                if(front==rear)
                {
                        front=-1;
                        rear=-1;
                }
                else
                front++;
                printf("Deleted element = %d\n",x);
                cnt--;
        }
}
void display()
{
        int i;
        if(front==-1)
```

```
        }
        else{
        printf("Elements in the queue : ");
        for(i=front;i<=rear;i++)
        {
                printf("%d ",Queue[i]);
        }
        printf("\n");
        }
}
void isEmpty()
{
        if(front==-1||rear==-1)
        {
                printf("Queue is empty.\n");
        }
        else
        {
        printf("Queue is not empty.\n");
        }
}
void size()
{
        printf("Queue size : %d\n",cnt);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
23
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 23 56
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.

1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 23
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 56
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
14
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :

78
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
53
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 14 78 53
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 3
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
1
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
2
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :

```
4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6
```

S.No: 18	Exp. Name: ***C Program to implement circular queue operations using arrays***	Date: 2023-04-27

Aim:

Write a C program to implement circular queues using arrays.

- Enqueue
- Dequeue
- Display
- Size
- Check if the queue is **Empty** or not.

Source Code:

CQueueUsingArray.c

```
#include <stdio.h>
#include <stdlib.h>
#include "CQueueOperations.c"

int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

CQueueOperations.c

```
#include<stdio.h>
#include<stdlib.h>
int a[5],i,j,front=-1,rear=-1,max=5;
void enqueue(int x)
{
        if((front==0&&rear==max-1)||(rear+1==front))
        {
                printf("Circular queue is overflow.\n");
        }
                else if(rear==-1 && front==-1)
                {
                        rear=front=0;
                        a[rear]=x;
                        printf("Successfully inserted.\n");
                }
                else
                {
                        rear++;
                        a[rear]=x;
                        printf("Successfully inserted.\n");
                }

}
void dequeue()
{
        if(front==-1&&rear==-1)
        {
                printf("Circular queue is underflow.\n");
        }
        else if(front==max-1)
        {
                printf("Deleted element = %d\n",a[front]);
                front=0;
        }
        else if(front==rear)
        {
                printf("Deleted element = %d\n",a[front]);
                rear = front=-1;
        }
        else
        {
                printf("Deleted element = %d\n",a[front++]);
        }
}
void display()
{
        if(front==-1 && rear==-1)
        {
                printf("Circular queue is empty.\n");
        }
                else if(front<=rear)
                {
                        printf("Elements in the circular queue : ");
                        for(i=front;i<=rear;i++)
                        printf("%d ",a[i]);
```

```
                else
                {
                        printf("Elements in the circular queue : ");
                        for(i=front;i<max;i++)
                        printf("%d",a[i]);
                        printf("\n");
                }
}
void isEmpty()
{
        if(front==-1&&rear==-1)
        {
        printf("Circular queue is empty.\n");
        }
        else
        {
        printf("Circular queue is not empty.\n");
        }
}
void size()
{
        if(front==-1&&rear==-1)
        {
        printf("Circular queue size : 0\n");
        }
        else if(front<=rear)
        {
                for(i=front,j=0;i<=rear;i++)
                j++;
                printf("Circular queue size : %d\n",j);
        }
        else
        {
        j=0;
        for(i=front;i<max;i++)
        j++;
        printf("Circular queue size: %d\n",j);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Circular queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Circular queue is empty.

```
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
11
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
12
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 11 12
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 11
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 12
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Circular queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6
```

Test Case - 2

User Output
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
34
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
55
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
26
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
77
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
38
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
59
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 34 55 26 77 38
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 5
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 34

1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 55
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 26
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 77 38
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6

Test Case - 3
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Circular queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Circular queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
12
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :

```
34
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 12 34 56
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
38
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
25
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 12 34 56 38 25
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Circular queue is overflow.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 12
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
2
Deleted element = 34
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 56 38 25
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
4
Circular queue is not empty.
```

1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
5
Circular queue size : 3
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
1
Enter element :
11
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
3
Elements in the circular queue : 56 38 25 11
1.Enqueue 2.Dequeue 3.Display 4.Is empty 5.Size 6.Exit
Enter your option :
6

S.No: 19	Exp. Name: ***C Program to Implementation of double-ended queue using arrays.***	Date: 2023-05-04

Aim:

Write a C Program to Implementation of double-ended queue using arrays.

- Inject
- Eject
- Display

Source Code:

DequeueArrayMain1.c

```
#include <stdio.h>
#include <conio.h>
#include "DequeueArrayInjectEject.c"
int main() {
  int op, x;
  while (1) {
    printf("1.Inject 2.Eject 3.Display 4.Exit\n");
    printf("Enter your option : ");
    scanf("%d", & op);
    switch (op) {
    case 1:
      printf("Enter element : ");
      scanf("%d", & x);
      inject(x);
      break;
    case 2:
      eject();
      break;
    case 3:
      display();
      break;
    case 4:
      exit(0);
    }
  }
}
```

DequeueArrayInjectEject.c

```
#define MAX 10
int dequeue[MAX];
int rear = -1,front = -1;
void inject(int x)
{
        if(rear==MAX-1)
        printf("Double ended queue is overflow\n");
        else
        {
                rear = rear+1;
                dequeue[rear]=x;
                if(front==-1)
                front=0;
                printf("Successfully inserted at rear side.\n");
        }
}
void eject()
{
        if(rear==-1)
        printf("Double ended queue is underflow.\n");
        else
        {
        printf("Deleted element from the rear side = %d\n",dequeue[rear]);
        if(front==rear)
        front=rear=-1;
        else
        rear=rear-1;
        }
}
void display()
{
        int i;
        if(front==-1 && rear==-1)
        printf("Double ended queue is empty.\n");
        else
        {
                printf("Elements in the double ended queue : ");
                for(i=front;i<=rear;i++)
                        printf("%d ",dequeue[i]);
                printf("\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
1
Enter element :
13

```
Successfully inserted at rear side.
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
1
Enter element :
14
Successfully inserted at rear side.
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
1
Enter element :
15
Successfully inserted at rear side.
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
3
Elements in the double ended queue : 13 14 15
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
2
Deleted element from the rear side = 15
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
3
Elements in the double ended queue : 13 14
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
1
Enter element :
26
Successfully inserted at rear side.
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
1
Enter element :
67
Successfully inserted at rear side.
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
3
Elements in the double ended queue : 13 14 26 67
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
2
Deleted element from the rear side = 67
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
2
Deleted element from the rear side = 26
1.Inject 2.Eject 3.Display 4.Exit
```

Enter your option :
2
Deleted element from the rear side = 14
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
3
Elements in the double ended queue : 13
1.Inject 2.Eject 3.Display 4.Exit
Enter your option :
4

S.No: 20	Exp. Name: ***C Program to implement Priority Queues.***	Date: 2023-05-18

Aim:

Write a C Program to implement **Priority Queues**.

- Enqueue
- Dequeue
- Display
- Size
- Check if the queue is **Empty** or not.

Source Code:

`PriorityQueueMain.c`

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
        int data;
        int priority;
        struct Node*next;
};
struct Node*front=NULL;
void enqueue(int data,int priority)
{
        struct Node*temp=(struct Node*)malloc(sizeof(struct Node));
        temp->data=data;
        temp->priority=priority;
        temp->next=NULL;
        if(front==NULL || priority<front->priority)
        {
                temp->next=front;
                front=temp;
        }
        else
        {
                struct Node*current=front;
                while(current->next!=NULL&¤t->next->priority<=priority)
                {
                        current=current->next;
                }
                temp->next=current->next;
                current->next=temp;
        }
}
void dequeue()
{
        if(front==NULL)
        {
                printf("Priority queue is underflow.\n");
                return;
        }
        struct Node*temp=front;
        printf("Deleted value = %d\n",temp->data);
        front=front->next;
        free(temp);
}
void display()
{
        if(front==NULL)
        {
          printf("Priority queue is empty.\n");
          return;
        }

    struct Node*current=front;
    printf("Elements in the priority queue : ");
    while(current!=NULL)
      {
```

```
        }
    printf("\n");
}
int size()
{
        int count=0;
        struct Node*current=front;
        while(current!=NULL)
        {
                count++;
                current=current->next;
        }
        return count;
}
int isEmpty()
{
        return front==NULL;
}
int main()
{
        int choice,data,priority;
        while(1)
        {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                        printf("Enter element : ");
                        scanf("%d",&data);
                        printf("Enter priority : ");
                        scanf("%d",&priority);
                        enqueue(data,priority);
                        break;
                        case 2:
                        dequeue();
                        break;
                        case 3:
                        display();
                        break;
                        case 4:
                        if(isEmpty())
                        {
                                printf("Priority queue is empty.\n");
                        }
                        else
                        {
                                printf("Priority queue is not empty.\n");
                        }
                        break;
                        case 5:
                        printf("Priority queue size : %d\n",size());
                        break;
                        case 6:
```

```
                }
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Priority queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Priority queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Priority queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Priority queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
34
Enter priority :
5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
78
Enter priority :
2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
56
Enter priority :
3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit

Enter your option :
1
Enter element :
89
Enter priority :
1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the priority queue : 89 (1) 78 (2) 56 (3) 34 (5)
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 89
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Priority queue size : 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 78
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 56
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the priority queue : 34 (5)
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
4
Enter priority :
4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
5

```
Enter priority :
4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the priority queue : 4 (4) 5 (4)
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
3
Enter priority :
2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the priority queue : 3 (2) 4 (4) 5 (4)
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6
```

S.No: 21	Exp. Name: ***C Program to implement a Single Linked list with a header and implement its operationsx***	Date: 2023-05-31

Aim:

C Program to implement a Single Linked list with a header and implement its operations

Operations on Single Linked List:

The following operations are performed on a Single Linked List :

- Insertion
- Deletion
- Reverse
- Traverse

Following are the statements to be displayed for Empty conditions

=>When List is Empty and if you have selected Deletion option, the following message should be displayed **"Single Linked List is empty so deletion is not possible"**

=>When List is Empty and if you have selected Traverse option, the following message should be displayed **"Single Linked List is empty"**

Source Code:

`Alloperations.c`

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *next;
};
typedef struct node *NODE;
NODE createNode()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        temp -> next =NULL;
        return temp;
}
NODE insertAtBegin(NODE first,int x)
{
        NODE temp;
        temp=createNode();
        temp->data = x;
        temp->next =first;
        first =temp;
        return first;
}
NODE deleteAtBegin(NODE first)
{
        NODE temp = first;
        first =first->next;
        printf("The deleted element from SLL : %d\n",temp->data);
        free(temp);
        return first;
}
void reverselist(NODE *first)
{
        struct node *prevNode,*curNode;
        if(*first!=NULL)
        {
                prevNode = *first;
                curNode=(*first)->next;
                *first = (*first)->next;
                prevNode->next=NULL;
                while(*first !=NULL)
                {
                        *first = (*first)->next;
                        curNode->next = prevNode;
                        prevNode = curNode;
                        curNode = *first;
                }
                *first = prevNode;
        }
}
void traverseList(NODE first)
{
        NODE temp = first;
        while(temp != NULL)
```

```
                temp = temp -> next;
        }
        printf("NULL\n");
}
void main()
{
        NODE first = NULL;
        int x,op;
        while(1)
        {
                printf("1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse
the List 5.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op)
                {
                        case 1: printf("Enter an element : ");
                        scanf("%d",&x);
                        first = insertAtBegin(first,x);
                        break;
                        case 2: if (first == NULL)
                        {
                                printf("Single Linked List is empty so deletion is not
possible\n");

                        }
                        else
                        {
                                first = deleteAtBegin(first);
                        }
                        break;
                        case 3:reverselist(&first);
                        printf("Single Linked List is reversed\n");
                        break;
                        case 4: if(first == NULL)
                        {
                                printf("Single Linked List is empty\n");
                        }
                        else
                        {
                                printf("The elements in SLL are : ");
                                traverseList(first);
                        }
                        break;
                        case 5: exit(0);
                }

        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
12
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
13
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
14
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
15
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 15 14 13 12 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
2
The deleted element from SLL : 15
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 14 13 12 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
3
Single Linked List is reversed
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 12 13 14 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
5

Test Case - 2
User Output
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit

```
Enter your option :
1
Enter an element :
101
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
102
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
103
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 103 102 101 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
2
The deleted element from SLL : 103
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 102 101 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
2
The deleted element from SLL : 102
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
3
Single Linked List is reversed
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 101 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
1
Enter an element :
102
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
4
The elements in SLL are : 102 101 NULL
1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit
Enter your option :
```

S.No: 22	Exp. Name: ***C program to Search the Position of a given element in SLL***	Date: 2023-05-31

Aim:

Write a C program to Insert an element at the beginning and **Search** the Position of a given element in a Single Linked List.

Source Code:

SingleLL.c

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

#include "SearchPositionOfEle.c"

void main() {
        NODE first = NULL;
        int x, pos, op;
        while(1) {
                printf("1.Insert At Begin 2.Search an element Position 3.Traverse the List
4.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1: printf("Enter an element : ");
                                        scanf("%d", &x);
                                        first = insertAtBegin(first, x);
                                        break;
                        case 2: printf("Enter search element : ");
                                        scanf("%d", &x);
                                        pos = searchPosOfEle(first, x);
                                        if (pos == 0) {
                                                printf("The given element %d is not found in
the given SLL\n", x);
                                        } else {
                                                printf("The given element %d is found at
position : %d\n", x, pos);
                                        }
                                        break;
                        case 3: if (first == NULL) {
                                                printf("Single Linked List is empty\n");
                                        } else {
                                                printf("The elements in SLL are : ");
                                                traverseList(first);
                                        }
                                        break;
                        case 4: exit(0);
                }
        }
}
```

SearchPositionOfEle.c

```
struct node
{
        int data;
        struct node*next;

};
typedef struct node*NODE;
NODE createNode()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        temp->next = NULL;
        return temp;
}
NODE insertAtBegin(NODE first,int x)
{
        NODE temp;
        temp = createNode();
        temp->data = x;
        temp->next =first;
        first = temp;
        return first;
}
int searchPosOfEle(NODE first,int key)
{
        NODE currentNode =first;
        int count = 0;
        if(currentNode == NULL)
        {
                return count;
        }
        while(currentNode != NULL && currentNode ->data!=key)
        {
                if(currentNode ->next ==NULL)
                {
                        return 0;
                }
                count++;
                currentNode =currentNode -> next;
        }
        return (count +1);
}
void traverseList(NODE first)
{
        NODE temp=first;
        while(temp!=NULL)
        {
        printf("%d --> ",temp->data);
        temp =temp->next;
    }
    printf("NULL\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
10
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
20
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
30
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
2
Enter search element :
10
The given element 10 is found at position : 3
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 30 --> 20 --> 10 --> NULL
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
2
Enter search element :
20
The given element 20 is found at position : 2
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
4

Test Case - 2
User Output
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
33
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :

```
1
Enter an element :
66
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
88
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 88 --> 66 --> 33 --> NULL
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
2
Enter search element :
99
The given element 99 is not found in the given SLL
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
2
Enter search element :
88
The given element 88 is found at position : 1
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 88 --> 66 --> 33 --> NULL
1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit
Enter your option :
4
```

S.No: 23	Exp. Name: ***C Program to concatenate the given two linked lists***	Date: 2023-05-31

Aim:

Write a **C** Program to **concatenate** the given two Single linked lists

Example:

Input and Output:-

```
Enter list-1 elements :
Enter element : 3
Enter element : 6
Enter element : 7
Enter element : 8
Enter element : -1
Enter list-2 elements :
Enter element : 1
Enter element : 4
Enter element : 5
Enter element : -1
Elements after merging is : 1 3 4 5 6 7 8
```

Source Code:

SingleLL.c

```
#include<stdio.h>
#include <stdlib.h>
#include "concatenateSLL.c"
void main() {
        NODE l1, l2, l3;
        l1 = l2 = l3 = NULL;
        printf("Enter list-1 elements :\n");
        l1 = createAndAddNodes(l1);
        sort(l1);
        printf("Enter list-2 elements :\n");
        l2 = createAndAddNodes(l2);
        sort(l2);
        l3 = merge(l1, l2);
        printf("Elements after merging is : ");
        print(l3);
}
```

concatenateSLL.c

```
struct node
{
        int data;
        struct node *next;
};
typedef struct node *NODE;
NODE createAndAddNodes(NODE first)
{
        NODE temp,q;
        int x;
        printf("Enter element : ");
        scanf("%d",&x);
        while(x!=-1)
        {
                temp=(NODE)malloc(sizeof(struct node));
                temp->data=x;
                temp->next=NULL;
        if(first == NULL)
        {
                first=temp;
        }
        else
        {
                q->next=temp;
        }
        q=temp;
        printf("Enter element : ");
        scanf("%d",&x);
   }
   return first;
}
NODE merge(NODE t1,NODE t2)
{
        if(t1==NULL)
        return t2;
        if(t2==NULL)
        return t1;
        if(t1->data<t2->data)
        {
                t1->next=merge(t1->next,t2);
                return t1;
        }
        else
        {
                t2->next=merge(t1,t2->next);
                return t2;
        }
}
NODE sort(NODE first)
{
        NODE t1,t2;
        int x;
        for(t1=first;t1->next!=NULL;t1=t1->next)
        {
                for(t2=t1->next;t2!=NULL;t2=t2->next)
```

```
                        {
                                x=t1->data;
                                t1->data=t2->data;
                                t2->data=x;
                        }
                }
        }
        return first;
}
void print(NODE node)
{
        while(node!=NULL)
        {
                printf("%d ",node->data);
                node=node->next;
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter list-1 elements :
Enter element :
3
Enter element :
6
Enter element :
7
Enter element :
8
Enter element :
-1
Enter list-2 elements :
Enter element :
1
Enter element :
4
Enter element :
5
Enter element :
-1
Elements after merging is : 1 3 4 5 6 7 8

Test Case - 2
User Output
Enter list-1 elements :

```
Enter element :
65
Enter element :
75
Enter element :
85
Enter element :
95
Enter element :
-1
Enter list-2 elements :
Enter element :
55
Enter element :
45
Enter element :
95
Enter element :
-1
Elements after merging is : 45 55 65 75 85 95 95
```

S.No: 24	Exp. Name: ***C Program to create circular linked list with header and implement its operations***	**Date: 2023-05-31**

Aim:

Write a **C** Program to create a **Circular linked list** with a header and implement its operations

- Insertion
- Deletion
- Search
- Traversal

Source Code:

`allOperationsInCLL.c`

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *next;
};
typedef struct node *NODE;
NODE createNodeInCLL()
{
        NODE temp;
        temp=(NODE)malloc(sizeof(struct node));
        temp->next=NULL;
        return temp;
}
NODE insertAtPositionInCLL(NODE first,int pos,int x)
{
        NODE temp,lastNode=first;
        int i;
        for(i=1;i<(pos-1);i++)
        {
                if(lastNode ->next == first)
                {
                        printf("No such position in CLL so insertion is not possible\n");
                        return first;
                }
                lastNode =lastNode->next;
        }
        temp=createNodeInCLL();
        temp->data=x;
        if(pos==1)
        {
                if(first == NULL)
                {
                        first =temp;
                        temp->next=first;
                }
                else
                {
                        while(lastNode->next!=first)
                        {
                                lastNode=lastNode->next;
                        }
                temp->next=first;
                first=temp;
                lastNode->next=first;
                }
        }
        else
        {
                temp->next=lastNode->next;
                lastNode->next=temp;
        }
        return first;
}
```

```
        NODE prev=first,lastNode=first;
        int i;
        if(pos==1)
        {
                if(prev->next==first)
                {
                        first=NULL;
                }
                else
                {
                        while(lastNode->next!=first)
                        {
                                lastNode=lastNode->next;
                        }
                        first=prev->next;
                        lastNode->next=first;
                }
        }
        else
        {
                for(i=1;i<pos;i++)
                {
                        if(prev->next==first)
                        {
                                printf("No such position in CLL so deletion is not
possible\n");
                                return first;
                        }
                        lastNode=prev;
                        prev=prev->next;
                }
                lastNode->next=prev->next;
        }
        printf("The deleted element from CLL : %d\n",prev->data);
        free(prev);
        return first;
}
int searchPosOfEleInCLL(NODE first,int key)
{
        NODE currentNode=first,q=first;
        int count=0;
        if(currentNode == NULL)
        {
                return count;
        }
        else{
                do{
                        count++;
                        q=currentNode;
                        if(currentNode->next==first && currentNode->data!=key)
                        {
                                return 0;
                        }
                        currentNode=currentNode->next;
                }while(q->next!=first && q->data!=key);
```

```
}
void traversalListInCLL(NODE first)
{
        NODE temp=first;
        do
        {
                printf("%d --> ",temp->data);
                temp=temp->next;
        }
        while(temp!=first);
        printf("\n");
}
int main()
{
        NODE first=NULL;
        int x,pos,op;
        while(1)
        {
                printf("1. Insert At specified position\n2. Traverse the List\n3. Search\n4.
Delete\n5. Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op)
                {
                case 1 : printf("Enter a position : ");
                scanf("%d",&pos);
                if(pos<=0)
                {
                        printf("No such position in CLL so insertion is not possible\n");
                }
                else
                {
                        printf("Enter an element : ");
                        scanf("%d",&x);
                        first=insertAtPositionInCLL(first,pos,x);
                }
                break;
                case 2: if(first==NULL)
                {
                        printf("Circular Linked List is empty\n");
                }
                else
                {
                        printf("The elements in CLL are : ");
                        traversalListInCLL(first);
                }
                break;
                case 3: printf("Enter search element : ");
                scanf("%d",&x);
                pos=searchPosOfEleInCLL(first,x);
                if(pos==0)
                {
                        printf("The given element %d is not found in the given CLL\n",x);
                }
                else
```

```
                }
                break;
                case 4:
                if(first == NULL)
                {
                        printf("Circular Linked List is empty so deletion is not
possible\n");
                }
                else
                {
                        printf("Enter position : ");
                        scanf("%d",&pos);
                        first=deleteAtPositionInCLL(first,pos);
                }
                break;
                case 5:exit(0);
                }
        }

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
12
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
13
1. Insert At specified position
2. Traverse the List

```
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
2
Enter an element :
14
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
3
Enter an element :
15
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 13 --> 14 --> 15 --> 12 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
4
Enter position :
2
The deleted element from CLL : 14
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
3
Enter search element :
12
The given element 12 is found at position : 3
1. Insert At specified position
2. Traverse the List
```

3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 13 --> 15 --> 12 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
5

Test Case - 2
User Output
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
1
Enter an element :
5
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
8
Enter an element :
2
No such position in CLL so insertion is not possible
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
2
Enter an element :

1
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
3
Enter an element :
5
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
4
Enter position :
9
No such position in CLL so deletion is not possible
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
4
Enter position :
2
The deleted element from CLL : 1
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
5

Test Case - 3
User Output
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1

```
Enter a position :
1
Enter an element :
1
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
2
Enter an element :
2
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
3
Enter an element :
3
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
4
Enter an element :
4
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
5
Enter an element :
5
1. Insert At specified position
2. Traverse the List
```

```
3. Search
4. Delete
5. Exit
Enter your option :
1
Enter a position :
6
Enter an element :
6
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
2
The elements in CLL are : 1 --> 2 --> 3 --> 4 --> 5 --> 6 -->
1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit
Enter your option :
5
```

S.No: 25	Exp. Name: ***C Program to create double linked list with header and implement its operations***	**Date: 2023-06-07**

Aim:

Write a **C** program that uses functions to perform the following **Operations on double linked list.**

- Creation
- Insertion
- Deletion
- Search
- Traversal

Source Code:

`AllOperationsDLL.c`

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node*prev;
        struct node*next;
}*head=NULL;
void insertBegin()
{
        struct node*ptr;
        ptr=(struct node*)malloc(sizeof(struct node));
        if(ptr==NULL)
        {
                printf("The Double Linked List is overflow\n");
        }
        else
        {
                printf("Enter an element: ");
                scanf("%d",&ptr->data);
                if(head==NULL)
                {
                        ptr->next=NULL;
                        ptr->prev=NULL;
                        head=ptr;
                }
                else
                {
                        ptr->prev=NULL;
                        ptr->next=head;
                        head->prev=ptr;
                        head=ptr;
                }
        }
}
void beginDelete()
{
        struct node*temp;
        if(head==NULL)
        {
                printf("Double Linked List is empty so deletion is not possible\n");
        }
        else if(head->next==NULL)
        {
                printf("The deleted element from DLL : %d\n",head->data);
                head=NULL;
                free(head);
        }
        else
        {
                temp=head;
                printf("The deleted element from DLL : %d\n",temp->data);
                head=head->next;
                head->prev=NULL;
                free(temp);
```

```
void search()
{
        int pos=0,ele,flag=0;
        struct node*temp;
        printf("Enter search element: ");
        scanf("%d",&ele);
        temp=head;
        while(temp!=NULL)
        {
                pos++;
                if(temp->data==ele)
                {
                        printf("The given element %d is found at position : %d\n",ele,pos);
                        flag=1;
                        break;
                }
                else
                {
                        flag=0;
                }
                temp=temp->next;
        }
        if(flag==0)
        {
                printf("The given element %d is not found in the given DLL\n",ele);
        }
}
void traverse()
{
        struct node*temp;
        if(head==NULL)
        {
                printf("Double Linked List is empty\n");
        }
        else
        {
                temp=head;
                printf("The elements in DLL are: ");
                while(temp!=NULL)
                {
                        printf("%d <--> ",temp->data);
                        temp=temp->next;
                }
                printf("NULL\n");
        }
}
int main()
{
        int op;
        while(1)
        {
                printf("1.Insert At Begin\n2.Delete at Begin\n3.Search an element
Position\n4.Traverse the List\n5.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
```

```
                        case 1 : insertBegin();
                        break;
                        case 2 : beginDelete();
                        break;
                        case 3: search();
                        break;
                        case 4: traverse();
                        break;
                        case 5:exit(0);
                }
        }
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
15
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
The deleted element from DLL : 15
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
12
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :

1
Enter an element:
16
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
17
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 17 <--> 16 <--> 12 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
3
Enter search element:
16
The given element 16 is found at position : 2
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 17 <--> 16 <--> 12 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
5

Test Case - 2
User Output
1.Insert At Begin

```
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
Double Linked List is empty so deletion is not possible
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
3
Enter search element:
0
The given element 0 is not found in the given DLL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
Double Linked List is empty
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
15
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
151
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
```

```
Enter an element:
1551
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
1
Enter an element:
15551
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
2
The deleted element from DLL : 15551
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 1551 <--> 151 <--> 15 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
3
Enter search element:
15
The given element 15 is found at position : 3
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
Enter your option :
4
The elements in DLL are: 1551 <--> 151 <--> 15 <--> NULL
1.Insert At Begin
2.Delete at Begin
3.Search an element Position
4.Traverse the List
5.Exit
```

```
Enter your option :
5
```

S.No: 26	Exp. Name: ***C Program to implement Doubly Circular Linked List Operations.***	Date: 2023-06-07

Aim:

Write a C Program to implement Doubly Circular Linked List Operations.

- Insertion
- Deletion
- Search
- Traverse the List

Source Code:

`AllOperationsDCLL.c`

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *prev;
        struct node *next;
};
typedef struct node *NODE;
NODE createNodeInDCLL()
{
        NODE temp;
        temp=(NODE)malloc(sizeof(struct node));
        temp->prev=NULL;
        temp->next=NULL;
        return temp;
}
NODE insertAtBeginInDCLL(NODE first,int x)
{
        NODE temp,lastNode;
        temp = createNodeInDCLL();
        temp->data=x;
        if(first == NULL)
        {
                temp->next=temp;
                temp->prev=temp;
        }
        else
        {
                lastNode=first->prev;
                temp->prev=lastNode;
                temp->next=first;
                lastNode->next=temp;
                first->prev=temp;
        }
        first=temp;
        return first;
}
NODE deleteAtBeginInDCLL(NODE first)
{
        NODE temp=first,lastNode=first;
        if(temp->next == first)
        {
                first=NULL;
        }
        else
        {
                lastNode=first->prev;
                first=first->next;
                first->prev=lastNode;
                lastNode->next=first;
        }
        printf("The deleted element from DCLL : %d\n",temp->data);
        free(temp);
        return first;
```

```
{
        NODE lastNode = first;
        do
        {
                printf("%d ",lastNode->data);
                lastNode=lastNode->next;
        }while(lastNode!=first);
        printf("\n");
}
int searchPosOfEleInDCLL(NODE first,int key)
{
        NODE currentNode=first;
        int count=0;
        if(currentNode==NULL)
        {
                return count;
        }
        while(currentNode!=NULL && currentNode->data!=key)
        {
                if(currentNode->next==first)
                {
                        return 0;
                }
                count++;
                currentNode=currentNode->next;
        }
        return(count+1);
}
void main()
{
        NODE first = NULL;
        int x,pos,op;
        while(1)
        {
                printf("1.Insert At Begin 2.Delete at Begin 3.Search an element Position
4.Traverse the List 5.Exit\n");
                printf("Enter your option: ");
                scanf("%d",&op);
                switch(op)
                {
                        case 1:printf("Enter an element: ");
                        scanf("%d",&x);
                        first = insertAtBeginInDCLL(first,x);
                        break;
                        case 2: if(first == NULL)
                        {
                                printf("Double Linked List is empty so deletion is not
possible\n");
                        }
                        else
                        {
                                first=deleteAtBeginInDCLL(first);
                        }
                        break;
                        case 3:
```

```
                        pos=searchPosOfEleInDCLL(first,x);
                        if(pos == 0)
                        {
                        printf("The given element %d is not found in the given DCLL\n",x);
                        }
                        else
                        {
                                printf("The given element %d is found at position:
%d\n",x,pos);
                        }
                        break;
                        case 4:
                        if(first==NULL)
                        {
                                printf("Doubly Circular Linked List is empty\n");
                        }
                        else
                        {
                                printf("The elements in DCLL are: ");
                                traverselistInDCLL(first);
                        }
                        break;
                        case 5: exit(0);
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
18
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
19
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
21

1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
400
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
4
The elements in DCLL are: 400 21 19 18
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
2
The deleted element from DCLL : 400
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
4
The elements in DCLL are: 21 19 18
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
3
Enter search element:
19
The given element 19 is found at position: 2
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
5

Test Case - 2
User Output
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
100
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
1
Enter an element:
200
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit

Enter your option:
1
Enter an element:
300
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
4
The elements in DCLL are: 300 200 100
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
3
Enter search element:
400
The given element 400 is not found in the given DCLL
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
3
Enter search element:
300
The given element 300 is found at position: 1
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
2
The deleted element from DCLL : 300
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
4
The elements in DCLL are: 200 100
1.Insert At Begin 2.Delete at Begin 3.Search an element Position 4.Traverse the List 5.Exit
Enter your option:
5

S.No: 27	Exp. Name: ***C Program to implement stack operations using Linked Lists.***	Date: 2023-05-31

Aim:

Write a C Program to implement **stack** operations using Linked Lists.

- Push
- Pop
- Peek
- Display
- Check whether the Stack is **Empty** or not

Source Code:

StackUsingLL.c

```
#include <stdio.h>
#include <stdlib.h>
#include "StackOperationsLL.c"

int main() {
        int op, x;
        while(1) {
                printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d", &x);
                                push(x);
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                peek();
                                break;
                        case 6:
                                exit(0);
                }
        }
}
```

StackOperationsLL.c

```
struct node
{
        int data;
        struct node*next;
};
struct node *top;
void push(int x)
{
        struct node *ptr;
        ptr=(struct node*)malloc(sizeof(struct node));
        ptr->data=x;
        ptr->next=top;
        top=ptr;
        printf("Successfully pushed.\n");
}
void pop()
{
        if(top==NULL)
        {
                printf("Stack is underflow.\n");
        }
        else
        {
                struct node *ptr;
                ptr=top;
                printf("Popped value = %d\n",ptr->data);
                top=top->next;
                free(ptr);
        }
}
void display()
{
        struct node *ptr;
        ptr=top;
        if(ptr==NULL)
        {
                printf("Stack is empty.\n");
        }
        else
        {
                printf("Elements of the stack are : ");
                while(ptr!=NULL)
                {
                        printf("%d ",ptr->data);
                        ptr=ptr->next;
                }
                printf("\n");
        }
}
void isEmpty()
{
        if(top==NULL)
        {
                printf("Stack is empty.\n");
        }
```

```
                printf("Stack is not empty.\n");
        }
}
void peek()
{
        if(top==NULL)
        {
                printf("Stack is underflow.\n");
        }else
        {
                printf("Peek value = %d\n",top->data);
        }
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
33
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
22
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
55
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
66
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 66 55 22 33
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :

2
Popped value = 66
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 55
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 22 33
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 22
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is not empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
23
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :

```
1
Enter element :
24
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 24 23
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 24
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 24
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 23
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6
```

S.No: 28	Exp. Name: ***C Program to implement queue operations using Linked Lists***	Date: 2023-05-31

Aim:

Write a C Program to implement queue operations using Linked Lists

- Enqueue
- Dequeue
- Display
- Size
- Check if the queue is **Empty** or not.

Source Code:

QueueUsingLL.c

```
#include <conio.h>
#include <stdio.h>
#include "QueueOperationsLL.c"
int main() {
        int op, x;
        while(1) {
                printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op) {
                        case 1:
                                printf("Enter element : ");
                                scanf("%d",&x);
                                enqueue(x);
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                isEmpty();
                                break;
                        case 5:
                                size();
                                break;
                        case 6: exit(0);
                }
        }
}
```

QueueOperationsLL.c

```
struct queue
{
        int data;
        struct queue *next;
};
typedef struct queue *Q;
Q front=NULL,rear=NULL;
void enqueue(int element)
{
     Q temp=NULL;
     temp=(Q)malloc(sizeof(struct queue));
     if(temp==NULL)
     {
        printf("Queue is overflow.\n");
     }
     else
     {
        temp->data=element;
        temp->next=NULL;
        if(front==NULL)
        {
                front=temp;
        }
        else
        {
                rear->next=temp;
        }
        rear=temp;
        printf("Successfully inserted.\n");
     }
}
void dequeue()
{
        Q temp=NULL;
        if(front==NULL)
        {
                printf("Queue is underflow.\n");
        }
        else
        {
                temp=front;
                if(front==rear)
                {
                        front=rear=NULL;
                }
                else
                {
                        front=front->next;
                }
                printf("Deleted value = %d\n",temp->data);
                free(temp);
        }
}
void display()
{
```

```
                printf("Queue is empty.\n");
        }
        else
        {
                Q temp=front;
                printf("Elements in the queue : ");
                while(temp!=NULL)
                {
                        printf("%d ",temp->data);
                        temp=temp->next;
                }
                printf("\n");
        }
}
void size()
{
                int count=0;
                Q temp=front;
                while(temp!=NULL)
                {
                        temp=temp->next;
                        count=count+1;
                }
                printf("Queue size : %d\n",count);
}
void isEmpty()
{
        if(front==NULL)
        {
                printf("Queue is empty.\n");
        }
        else
        {
                printf("Queue is not empty.\n");
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :

```
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
44
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
55
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
66
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
67
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 44 55 66 67
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 44
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 55
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.
```

1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 2
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
23
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
234
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
45
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
456
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 23
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 234 45 456
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 234
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 45 456
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :

4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

Test Case - 3
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Queue is underflow.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
25
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
1
Enter element :
27
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 25 27
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
4
Queue is not empty.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 2

1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 25
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
2
Deleted value = 27
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 0
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

S.No: 29	Exp. Name: ***C Program to create create an XOR Linked List and implement its operations***	Date: 2023-06-07

Aim:

Write a **C** Program to create an **XOR Linked List and** implement its operations

- Insertion
- Deletion
- Traversal

Source Code:

`XORLinkedList.c`

```
#include<stdio.h>
#include<stdlib.h>
#include<stdint.h>
typedef struct Node
{
        int data;
        struct Node*npx;
}Node;
Node*XOR(Node*a,Node*b)
{
        return(Node*)((uintptr_t)(a)^(uintptr_t)(b));
}
Node*insert(Node* head,int data)
{
        Node*new_node=(Node*)malloc(sizeof(Node));
        new_node->data=data;
        new_node->npx=XOR(head,NULL);
        if(head!=NULL)
        {
                Node*next=XOR(head->npx,NULL);
                head->npx=XOR(new_node,next);
        }
        head=new_node;
        return head;
}
void printList(Node*head)
{
        Node*curr=head;
        Node*prev=NULL;
        Node*next;
            printf("List contents: ");
                while(curr!=NULL)
                {
                        printf("%d ",curr->data);
                        next=XOR(prev,curr->npx);
                        prev=curr;
                        curr=next;
                }
                printf("\n");
}
int main()
{
        Node*head=NULL;
        int choice,data;
        do
        {
                printf("1. Insert\n2. Print List\n3. Exit\n");
                printf("Enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                        printf("Enter data to insert: ");
                        scanf("%d",&data);
                        head=insert(head,data);
```

```
                        printList(head);
                        break;
                        case 3:
                        printf("Exiting...\n");
                        break;
                        default:
                        printf("Invalid choice.\n");
                        break;
                }
        }while(choice!=3);
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
77
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
50
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
90
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
758
1. Insert
2. Print List
3. Exit

Enter your choice:
1
Enter data to insert:
789
1. Insert
2. Print List
3. Exit
Enter your choice:
2
List contents: 789 758 90 50 77
1. Insert
2. Print List
3. Exit
Enter your choice:
3
Exiting...

Test Case - 2
User Output
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
4
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
5
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
7
1. Insert
2. Print List
3. Exit
Enter your choice:
1
Enter data to insert:
53
1. Insert

```
2. Print List
3. Exit
Enter your choice:
2
List contents: 53 7 5 4
1. Insert
2. Print List
3. Exit
Enter your choice:
3
Exiting...
```

S.No: 30	Exp. Name: ***C Program to implement Bubble Sort***	Date: 2023-06-07

Aim:

C Program to implement Bubble Sort

Example:

Input and Output:

```
Enter value of n : 3
Enter element for a[0] : 34
Enter element for a[1] : 25
Enter element for a[2] : 28
Before sorting the elements in the array are
Value of a[0] = 34
Value of a[1] = 25
Value of a[2] = 28
After sorting the elements in the array are
Value of a[0] = 25
Value of a[1] = 28
Value of a[2] = 3
```

Source Code:

BubbleSort.c

```c
#include<stdio.h>
int main()
{
        int a[20],i,j,temp,n;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Before sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-i-1;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
        printf("After sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
3
Enter element for a[0] :
34
Enter element for a[1] :
25
Enter element for a[2] :
28
Before sorting the elements in the array are
Value of a[0] = 34
Value of a[1] = 25
Value of a[2] = 28

After sorting the elements in the array are
Value of a[0] = 25
Value of a[1] = 28
Value of a[2] = 34

Test Case - 2
User Output
Enter value of n :
5
Enter element for a[0] :
1
Enter element for a[1] :
6
Enter element for a[2] :
3
Enter element for a[3] :
8
Enter element for a[4] :
4
Before sorting the elements in the array are
Value of a[0] = 1
Value of a[1] = 6
Value of a[2] = 3
Value of a[3] = 8
Value of a[4] = 4
After sorting the elements in the array are
Value of a[0] = 1
Value of a[1] = 3
Value of a[2] = 4
Value of a[3] = 6
Value of a[4] = 8

S.No: 31	Exp. Name: ***C program to Sort the elements using Selection Sort - Largest element***	**Date: 2023-06-07**

Aim:

C program to Sort the elements using Selection Sort - Largest element method Technique

Example:

Input and Output:

```
Enter value of n : 5
Enter element for a[0] : 99
Enter element for a[1] : 44
Enter element for a[2] : 77
Enter element for a[3] : 22
Enter element for a[4] : 55
Before sorting the elements in the array are
Value of a[0] = 99
Value of a[1] = 44
Value of a[2] = 77
Value of a[3] = 22
Value of a[4] = 55
After sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 44
Value of a[2] = 55
Value of a[3] = 77
Value of a[4] = 99
```

Source Code:

SelectionSort.c

```c
#include<stdio.h>
void main()
{
        int a[20],i,n,j,large,index;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Before sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
        for(i=n-1;i>0;i--)
        {
                large=a[0];
                index=0;
                for(j=1;j<=i;j++)
                {
                        if(a[j]>large)
                        {
                                large=a[j];
                                index=j;
                        }
                }
                a[index]=a[i];
                a[i]=large;
        }
        printf("After sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
4
Enter element for a[0] :
67
Enter element for a[1] :
34
Enter element for a[2] :
16
Enter element for a[3] :

49
Before sorting the elements in the array are
Value of a[0] = 67
Value of a[1] = 34
Value of a[2] = 16
Value of a[3] = 49
After sorting the elements in the array are
Value of a[0] = 16
Value of a[1] = 34
Value of a[2] = 49
Value of a[3] = 67

Test Case - 2
User Output
Enter value of n :
5
Enter element for a[0] :
99
Enter element for a[1] :
44
Enter element for a[2] :
77
Enter element for a[3] :
22
Enter element for a[4] :
55
Before sorting the elements in the array are
Value of a[0] = 99
Value of a[1] = 44
Value of a[2] = 77
Value of a[3] = 22
Value of a[4] = 55
After sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 44
Value of a[2] = 55
Value of a[3] = 77
Value of a[4] = 99

S.No: 32	Exp. Name: ***C program to Sort the elements using Selection Sort - Smallest element***	**Date: 2023-06-07**

Aim:

C program to Sort the elements using Selection Sort - Smallest element method Technique

Example:

Input and Output:-

```
Enter value of n : 4
Enter element for a[0] : 78
Enter element for a[1] : 43
Enter element for a[2] : 99
Enter element for a[3] : 27
Before sorting the elements in the array are
Value of a[0] = 78
Value of a[1] = 43
Value of a[2] = 99
Value of a[3] = 27
After sorting the elements in the array are
Value of a[0] = 27
Value of a[1] = 43
Value of a[2] = 78
Value of a[3] = 99
```

Source Code:

SelectionSort.c

```c
#include<stdio.h>
void main()
{
        int a[20],i,n,j,small,index;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Before sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
        for(i=0;i<n;i++)
        {
                small=a[i];
                index=i;
                for(j=i+1;j<n;j++)
                {
                        if(a[j]<small)
                        {
                                small=a[j];
                                index=j;
                        }
                }
                a[index]=a[i];
                a[i]=small;
        }
        printf("After sorting the elements in the array are\n");
        for(i=0;i<n;i++)
        {
                printf("Value of a[%d] = %d\n",i,a[i]);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
4
Enter element for a[0] :
78
Enter element for a[1] :
43
Enter element for a[2] :
99
Enter element for a[3] :

27
Before sorting the elements in the array are
Value of a[0] = 78
Value of a[1] = 43
Value of a[2] = 99
Value of a[3] = 27
After sorting the elements in the array are
Value of a[0] = 27
Value of a[1] = 43
Value of a[2] = 78
Value of a[3] = 99

Test Case - 2
User Output
Enter value of n :
6
Enter element for a[0] :
45
Enter element for a[1] :
23
Enter element for a[2] :
7
Enter element for a[3] :
85
Enter element for a[4] :
15
Enter element for a[5] :
9
Before sorting the elements in the array are
Value of a[0] = 45
Value of a[1] = 23
Value of a[2] = 7
Value of a[3] = 85
Value of a[4] = 15
Value of a[5] = 9
After sorting the elements in the array are
Value of a[0] = 7
Value of a[1] = 9
Value of a[2] = 15
Value of a[3] = 23
Value of a[4] = 45
Value of a[5] = 85

S.No: 33	Exp. Name: ***C Program to implement Insertion Sort***	Date: 2023-06-07

Aim:

C Program to implement Insertion Sort for the given elements.

Example:

Input and Output:-

```
Enter value of n : 5
Enter element for a[0] : 6
Enter element for a[1] : 9
Enter element for a[2] : 2
Enter element for a[3] : 5
Enter element for a[4] : 1
Before sorting the elements in the array are
Value of a[0] = 6
Value of a[1] = 9
Value of a[2] = 2
Value of a[3] = 5
Value of a[4] = 1
After sorting the elements in the array are
Value of a[0] = 1
Value of a[1] = 2
Value of a[2] = 5
Value of a[3] = 6
Value of a[4] = 9
```

Source Code:

InsertionSort.c

```c
#include<stdio.h>
void insert(int a[20],int n)
{
        int i,j,temp;
        for(i=1;i<n;i++)
        {
                temp=a[i];
                j=i-1;
                while(j>=0 && temp<=a[j])
                {
                        a[j+1]=a[j];
                        j=j-1;
                }
                a[j+1]=temp;
        }
}
void print(int a[20],int n)
{
        int i;
        for(i=0;i<n;i++)
        printf("Value of a[%d] = %d\n",i,a[i]);
}
int main()
{
        int a[20],n,i;
        printf("Enter value of n : ");
        scanf("%d",&n);
        for(i==0;i<n;i++)
        {
                printf("Enter element for a[%d] : ",i);
                scanf("%d",&a[i]);
        }
        printf("Before sorting the elements in the array are\n");
        print(a,n);
        insert(a,n);
        printf("After sorting the elements in the array are\n");
        print(a,n);

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
5
Enter element for a[0] :
6
Enter element for a[1] :
9
Enter element for a[2] :

2
Enter element for a[3] :
5
Enter element for a[4] :
1
Before sorting the elements in the array are
Value of a[0] = 6
Value of a[1] = 9
Value of a[2] = 2
Value of a[3] = 5
Value of a[4] = 1
After sorting the elements in the array are
Value of a[0] = 1
Value of a[1] = 2
Value of a[2] = 5
Value of a[3] = 6
Value of a[4] = 9

Test Case - 2
User Output
Enter value of n :
3
Enter element for a[0] :
5
Enter element for a[1] :
9
Enter element for a[2] :
2
Before sorting the elements in the array are
Value of a[0] = 5
Value of a[1] = 9
Value of a[2] = 2
After sorting the elements in the array are
Value of a[0] = 2
Value of a[1] = 5
Value of a[2] = 9

S.No: 34	Exp. Name: ***C Program to implement shell sort***	Date: 2023-06-08

Aim:

C Program to implement shell sort.

Example:

Input and Output:

Enter array size : 5
Enter 5 elements : 12 32 43 56 78
Before sorting the elements are : 12 32 43 56 78
After sorting the elements are : 12 32 43 56 78

Source Code:

ShellSortMain2.c

```
#include <stdio.h>
#include <conio.h>
#include "ShellSort2.c"
int main()  {
        int size;
        int *arr, i;
        printf("Enter array size : ");
        scanf("%d",&size);
        arr = (int*) malloc(size * sizeof(int));
        printf("Enter %d elements : ",size);
        for (i = 0; i < size; i++) {
                scanf("%d", &arr[i]);
        }
        printf("Before sorting the elements are : ");
        printArray(arr,size);

        shellSort(arr,size);

        printf("After sorting the elements are : ");
        printArray(arr,size);
        return 0;
}
```

ShellSort2.c

```
int shellSort(int arr[],int n)
{
  for(int gap=n/2;gap>0;gap/=2)
  {
        for(int i=gap;i<n;i+=1)
        {
                int temp=arr[i];
                int j;
                for(j=i;j>=gap&&arr[j-gap]>temp;j-=gap)
                arr[j]=arr[j-gap];
                arr[j]=temp;
        }
 }
        return 0;
}
void printArray(int arr[],int n)
{
        for(int i=0;i<n;i++)
        printf("%d ",arr[i]);
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size :
5
Enter 5 elements :
12 32 43 56 78
Before sorting the elements are : 12 32 43 56 78
After sorting the elements are : 12 32 43 56 78

Test Case - 2
User Output
Enter array size :
7
Enter 7 elements :
12 34 25 65 36 26 65
Before sorting the elements are : 12 34 25 65 36 26 65
After sorting the elements are : 12 25 26 34 36 65 65

S.No: 35	Exp. Name: ***C Program to implement Radix sort.***	Date: 2023-06-08

Aim:

C Program to implement Radix sort.

Example:

Input and Output:

Enter the size of the array: 5
Enter 5 elements : 23 43 54 12 65
Before sorting the elements are : 23 43 54 12 65
After sorting the elements are : 12 23 43 54 65

Source Code:

RadixSortMain2.c

```
#include <stdio.h>
#include <conio.h>
#include "RadixSort2.c"
int main()  {
        int size;
        int *arr, i;
        printf("Enter array size : ");
    scanf("%d",&size);
        arr = (int*) malloc(size * sizeof(int));
        printf("Enter %d elements : ",size);
    for (i = 0; i < size; i++) {
                scanf("%d", &arr[i]);
    }
    printf("Before sorting the elements are : ");
        printArray(arr,size);
        RadixSort(arr,size);
        printf("After sorting the elements are : ");
        printArray(arr,size);
        return 0;
}
```

RadixSort2.c

```
int largest(int a[],int n)
{
        int large =a[0],i;
        for(i=1;i<n;i++)
        {
                if(large<a[i])
                large=a[i];
        }
        return large;
}
void printArray(int arr[],int n)
{
        for(int i=0;i<n;i++)
        printf("%d ",arr[i]);
        printf("\n");
}
void RadixSort(int a[],int n)
{
        int bucket[10][10],bucket_count[10];
        int i,j,k,remainder,NOP=0,divisor=1,large,pass;
        large=largest(a,n);
        while(large>0)
        {
                NOP++;
                large/=10;
        }
        for(pass=0;pass<NOP;pass++)
        {
                for(i=0;i<10;i++)
                {
                        bucket_count[i]=0;
                }
                for(i=0;i<n;i++)
                {
                        remainder=(a[i]/divisor)%10;
                        bucket[remainder][bucket_count[remainder]]=a[i];
                        bucket_count[remainder]+=1;
                }
                i=0;
                for(k=0;k<10;k++)
                {
                        for(j=0;j<bucket_count[k];j++)
                        {
                                a[i]=bucket[k][j];
                                i++;
                        }
                }
                divisor*=10;
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Enter array size :
5
Enter 5 elements :
23 43 54 12 65
Before sorting the elements are : 23 43 54 12 65
After sorting the elements are : 12 23 43 54 65

Test Case - 2
User Output
Enter array size :
7
Enter 7 elements :
23 54 136 85 24 65 76
Before sorting the elements are : 23 54 136 85 24 65 76
After sorting the elements are : 23 24 54 65 76 85 136