

S.No: 1

Exp. Name: ***Design a C program which sorts the strings using array of pointers***

Date: 2023-04-23

**Aim:**

Design a C program that sorts the strings using array of pointers.

**Sample input output**

**Sample input-output -1:**

Enter the number of strings: 2

Enter string 1: Tantra

Enter string 2: Code

Before Sorting

Tantra

Code

After Sorting

Code

Tantra

**Sample input-output -2:**

Enter the number of strings: 3

Enter string 1: India

Enter string 2: USA

Enter string 3: Japan

Before Sorting

India

USA

Japan

After Sorting

India

Japan

USA

**Source Code:**

stringssort.c

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
    char * temp;
    int i,j,diff,n;
    char * strarray[10];
    printf("Enter the number of strings: ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter string %d: ",i+1);
        strarray[i]=(char *)malloc(sizeof(char)*20);
        scanf("%s",strarray[i]);
    }
    printf("Before Sorting\n");
    for(i=0;i<n;i++)
    {
        printf("%s\n",strarray[i]);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-1;j++)
        {
            diff=strcmp(strarray[j],strarray[j+1]);
            if(diff>0)
            {
                temp=strarray[j];
                strarray[j]=strarray[j+1];
                strarray[j+1]=temp;
            }
        }
    }
    printf("After Sorting\n");
    for(i=0;i<n;i++)
    {
        printf("%s\n",strarray[i]);
    }
}

```

### Execution Results - All test cases have succeeded!

Test Case - 1
<b>User Output</b>
Enter the number of strings:
2
Enter string 1:
Tantra
Enter string 2:
Code

Before Sorting

Tantra

Code

After Sorting

Code

Tantra

### Test Case - 2

#### User Output

Enter the number of strings:

3

Enter string 1:

Dhoni

Enter string 2:

Kohli

Enter string 3:

Rohit

Before Sorting

Dhoni

Kohli

Rohit

After Sorting

Dhoni

Kohli

Rohit

**Aim:**

Write a program to search a **key element** with in the given array of elements using [linear search](#) process.

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

Enter value of n :

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

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :  
Enter element for a[1] :  
Enter element for a[2] :

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

Enter element for a[0] : 89  
Enter element for a[1] : 33  
Enter element for a[2] : 56

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

Enter key element :

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

Enter key element : 56

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

The key element 56 is found at the position 2

Similarly if the key element is given as **25** for the above one dimensional array elements then the program should print the output as "**The key element 25 is not found in the array**".

Fill in the missing code so that it produces the desired result.

**Source Code:**

LinearSearch.c

```

#include<stdio.h>
int main()
{
    int a[10],i,j,n,flag=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",&j);
    for(i=0;i<n;i++)
    {
        if(j==a[i])
        {
            flag++;
            break;
        }
    }
    if(flag==1)
    {
        printf("The key element %d is found at the position %d",j,i);
    }
    else
    {
        printf("The key element %d is not found in the array",j);
    }
    printf("\n");
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
Enter value of n :	
4	
Enter element for a[0] :	
1	
Enter element for a[1] :	
22	
Enter element for a[2] :	
33	
Enter element for a[3] :	
44	
Enter key element :	
22	
The key element 22 is found at the position 1	

### Test Case - 2

#### User Output

Enter value of n :

7

Enter element for a[0] :

101

Enter element for a[1] :

102

Enter element for a[2] :

103

Enter element for a[3] :

104

Enter element for a[4] :

105

Enter element for a[5] :

106

Enter element for a[6] :

107

Enter key element :

110

The key element 110 is not found in the array

**Aim:**

Write a program to **search** a key element in the given array of elements using [binary search](#).

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

Enter value of n :

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

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :  
Enter element for a[1] :  
Enter element for a[2] :

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

Enter element for a[0] : 89  
Enter element for a[1] : 33  
Enter element for a[2] : 56

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

Enter key element :

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

Enter key element : 56

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

After sorting the elements in the array are  
Value of a[0] = 33  
Value of a[1] = 56  
Value of a[2] = 89  
The key element 56 is found at the position 1

Similarly if the key element is given as **25** for the above one dimensional array elements then the program should print the output as "**The Key element 25 is not found in the array**".

Fill in the missing code so that it produces the desired result.

**Source Code:**

BinarySearch.c

```

#include<stdio.h>
void main()
{
    int a[5],i,j,temp,k,n,flag=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]);
    }
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[i])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("Enter key element : ");
    scanf("%d",&k);
    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(k==a[i])
        {
            flag++;
            break;
        }
    }
    if(flag==1)
        printf("The key element %d is found at the position %d\n",k,i);
    else
        printf("The Key element %d is not found in the array\n",k);
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1	
<b>User Output</b>	
Enter value of n :	
3	
Enter element for a[0] :	
25	

Enter element for a[1] :
15
Enter element for a[2] :
23
Enter key element :
45
After sorting the elements in the array are
Value of a[0] = 15
Value of a[1] = 23
Value of a[2] = 25
The Key element 45 is not found in the array

<b>Test Case - 2</b>
<b>User Output</b>
Enter value of n :
2
Enter element for a[0] :
80
Enter element for a[1] :
39
Enter key element :
50
After sorting the elements in the array are
Value of a[0] = 39
Value of a[1] = 80
The Key element 50 is not found in the array

**Aim:**

Write a C program to implement **Fibonacci search** technique

**Source Code:**

## FibonacciSearch.c

```
#include<stdio.h>
void main()
{
    int a[20],i,j,n,flag=0;
    printf("Enter the size of an array: ");
    scanf("%d",&n);
    printf("Enter the %d array elements\n",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("Enter the element to be searched: ");
    scanf("%d",&j);
    for(i=0;i<n;i++)
    {
        if(j==a[i])
        {
            flag++;
            break;
        }
    }
    if(flag==1)
        printf("Element found at index: %d.\n",i);
    else
        printf("Element not found.\n");
}
```

**Execution Results - All test cases have succeeded!**

Test Case - 1
<b>User Output</b>
Enter the size of an array:
5
Enter the 5 array elements
3 4 5 6 7
Enter the element to be searched:
3
Element found at index: 0.

**Test Case - 2**

**User Output**

Enter the size of an array:

5

Enter the 5 array elements

3 4 5 6 7

Enter the element to be searched:

4

Element found at index: 1.

S.No: 5

Exp. Name: **Write a C program to Sort the elements using Insertion Sort Technique**

Date: 2023-04-26

**Aim:**

Write a program to **sort** the given elements using [insertion sort technique](#).

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

Enter value of n :

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

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :  
Enter element for a[1] :  
Enter element for a[2] :

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

Enter element for a[0] : 22  
Enter element for a[1] : 33  
Enter element for a[2] : 12

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

Before sorting the elements in the array are  
Value of a[0] = 22  
Value of a[1] = 33  
Value of a[2] = 12  
After sorting the elements in the array are  
Value of a[0] = 12  
Value of a[1] = 22  
Value of a[2] = 33

Fill in the missing code so that it produces the desired result.

**Source Code:**

InsertionSortDemo3.c

```

#include<stdio.h>
void sort(int[],int);
void 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");
    for(i=0;i<n;i++)
    {
        printf("Value of a[%d] = %d\n",i,a[i]);
    }
    sort(a,n);
    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]);
    }
}
void sort(int a[],int n)
{
    int i,j,k;
    for(i=1;i<n;i++)
    {
        k=a[i];
        j=i-1;
        while(j>=0&&a[j]>k)
        {
            a[j+1]=a[j];
            j=j-1;
        }
        a[j+1]=k;
    }
}

```

### Execution Results - All test cases have succeeded!

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

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

Test Case - 2	
<b>User Output</b>	
Enter value of n :	
3	
Enter element for a[0] :	
5	
Enter element for a[1] :	
9	
Enter element for a[2] :	
4	
Before sorting the elements in the array are	
Value of a[0] = 5	
Value of a[1] = 9	
Value of a[2] = 4	
After sorting the elements in the array are	
Value of a[0] = 4	
Value of a[1] = 5	
Value of a[2] = 9	

S.No: 6

Exp. Name: **Write a C program to Sort the elements using Selection Sort - Smallest element method Technique**

Date: 2023-05-01

**Aim:**

Write a program to **sort** the given array elements using **selection sort smallest element** method.

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

Enter value of n :

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

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :  
Enter element for a[1] :  
Enter element for a[2] :

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

Enter element for a[0] : 22  
Enter element for a[1] : 33  
Enter element for a[2] : 12

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

Before sorting the elements in the array are  
Value of a[0] = 22  
Value of a[1] = 33  
Value of a[2] = 12  
After sorting the elements in the array are  
Value of a[0] = 12  
Value of a[1] = 22  
Value of a[2] = 33

Fill in the missing code so that it produces the desired result.

**Source Code:**

SelectionSortDemo6.c

```

#include<stdio.h>
void main()
{
    int a[20],i,j,n,s,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");
    for(i=0;i<n;i++)
    {
        printf("Value of a[%d] = %d\n",i,a[i]);
    }
    for(i=0;i<n;i++)
    {

        for(j=i+1;j<n;j++)
        {
            I=i;
            if(a[j]<a[I])
            {
                I=j;
            }
            s=a[i];
            a[i]=a[I];
            a[I]=s;
        }
    }
    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]);
        printf("\n");
    }
}

```

## 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

S.No: 7

Exp. Name: **Write a C program to sort given elements using shell sort technique.**

Date: 2023-05-01

**Aim:**

Write a program to **sort** (**ascending order**) the given elements using **shell sort** technique.

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

Enter array size :

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

Enter array size : 5

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

Enter 5 elements :

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

Enter 5 elements : 34 67 12 45 22

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

Before sorting the elements are : 34 67 12 45 22  
After sorting the elements are : 12 22 34 45 67

**Note:** Do use the **printf()** function with a **newline** character (**\n**).

**Source Code:**

ShellSort2.c

```

#include<stdio.h>
#include<conio.h>
void sort(int [],int );
void main()
{
    int a[20];
    int n, i;
    printf("Enter array size : ");
    scanf("%d",&n);
    printf("Enter %d elements : ",n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("Before sorting the elements are : ");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    sort(a,n);
    printf("\nAfter sorting the elements are : ");
    for(i=0;i<n;i++)
        printf("%d ",a[i]);
    printf("\n");
}
void sort(int arr[],int n)
{
    int gap, i, j, temp;
    for(gap=n/2;gap>0;gap/=2)
    {
        for(i=gap;i<n;i++)
        {
            temp = arr[i];
            for(j=i;j>=gap && arr[j-gap]>temp;j-=gap)
            {
                arr[j] = arr[j-gap];
            }
            arr[j] = temp;
        }
    }
}

```

### 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	

S.No: 8

Exp. Name: **Write a C program to Sort the elements using Bubble Sort Technique**

Date: 2023-04-26

**Aim:**

Write a program to **sort** the given elements using **bubble sort technique**.

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

Enter value of n :

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

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :  
Enter element for a[1] :  
Enter element for a[2] :

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

Enter element for a[0] : 22  
Enter element for a[1] : 33  
Enter element for a[2] : 12

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

Before sorting the elements in the array are  
Value of a[0] = 22  
Value of a[1] = 33  
Value of a[2] = 12  
After sorting the elements in the array are  
Value of a[0] = 12  
Value of a[1] = 22  
Value of a[2] = 33

Fill in the missing code so that it produces the desired result.

**Source Code:**

BubbleSortDemo3.c

```

#include<stdio.h>
void main()
{
    int a[20],i,j,n,temp;
    printf("Enter value of n : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter element for a[%d] : ",i,a[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=i+1;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]);
    }
}

```

### 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: 9

Exp. Name: **Write a program to sort Ascending order the given elements using quick sort technique.**

Date: 2023-04-26

**Aim:**

Write a program to **sort** (**Ascending order**) the given elements using **quick sort** technique.

**Note: Pick the first element as pivot. You will not be awarded marks if you do not follow this instruction.**

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

Enter array size :

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

Enter array size : 5

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

Enter 5 elements :

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

Enter 5 elements : 34 67 12 45 22

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

Before sorting the elements are : 34 67 12 45 22  
After sorting the elements are : 12 22 34 45 67

**Note:** Do use the **printf()** function with a **newline** character (**\n**).

**Source Code:**

QuickSortMain.c

```

#include<stdio.h>
void sort(int [],int ,int );
void main()
{
    int arr[20],i,n;
    printf("Enter array size : ",n);
    scanf("%d",&n);
    printf("Enter %d elements : ",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Before sorting the elements are : ");
    for(i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    sort(arr,0,n-1);
    printf("\nAfter sorting the elements are : ");
    for(i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\n");
}

void sort(int a[20],int low, int high)
{
    int left,right,pivolt,temp;
    left=low;
    right=high;
    pivolt=a[(low+high)/2];
    do
    {
        while(a[left]<pivolt)
        left++;
        while(a[right]>pivolt)
        right--;
        if(left<=right)
        {
            temp=a[left];
            a[left]=a[right];
            a[right]=temp;
            right--;
            left++;
        }
    while(left<=right);
    if(low<right)
    sort(a,low,right);
    if(left<high)
    sort(a,left,high);
    }
}

```

### **Test Case - 1**

#### **User Output**

Enter array size :

5

Enter 5 elements :

34 67 12 45 22

Before sorting the elements are : 34 67 12 45 22

After sorting the elements are : 12 22 34 45 67

### **Test Case - 2**

#### **User Output**

Enter array size :

8

Enter 8 elements :

77 55 22 44 99 33 11 66

Before sorting the elements are : 77 55 22 44 99 33 11 66

After sorting the elements are : 11 22 33 44 55 66 77 99

### **Test Case - 3**

#### **User Output**

Enter array size :

5

Enter 5 elements :

-32 -45 -67 -46 -14

Before sorting the elements are : -32 -45 -67 -46 -14

After sorting the elements are : -67 -46 -45 -32 -14

S.No: 10

Exp. Name: ***Write a C program to sort the given elements using Heap sort***

Date: 2023-05-08

**Aim:**

Write a program to sort (ascending order) the given elements using heap sort technique.

Note: Do use the printf() function with a newline character (\n).

**Source Code:**

HeapSortMain.c

Page No: 26

ID: 224G1A05B1

2022-2026-CSE-B

Srinivasa Ramanujan Institute of Technology

```

#include <stdio.h>
void main()
{
    int arr[15], i, n;
    printf("Enter array size : ");
    scanf("%d", &n);
    printf("Enter %d elements : ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Before sorting the elements are : ");
    display(arr, n);
    heapsort(arr,n);
    printf("After sorting the elements are : ");
    display(arr, n);
}
int display(int arr[15],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\n");
}

int heapsort(int arr[15],int n)
{
    for(int i=n/2-1;i>=0;i--)
    {
        heapify(arr,n,i);

    }
    for(int i=n-1;i>=0;i--)
    {
        int temp=arr[0];
        arr[0]=arr[i];
        arr[i]=temp;
        heapify(arr,i,0);

    }
    int heapify(int arr[15],int n,int i)
    {
        int largest=i;
        int l=2*i+1;
        int r=2*i+2;
        if(l<n && arr[l]>arr[largest])largest=l;
        if(r<n && arr[r]>arr[largest])largest=r;
        if(largest!=i){int temp=arr[i];arr[i]=arr[largest];
        arr[largest]=temp;
        heapify(arr,n,largest);

    }
}

```

## Execution Results - All test cases have succeeded!

<b>Test Case - 1</b>
<b>User Output</b>
Enter array size :
5
Enter 5 elements :
23 54 22 44 12
Before sorting the elements are : 23 54 22 44 12
After sorting the elements are : 12 22 23 44 54

<b>Test Case - 2</b>
<b>User Output</b>
Enter array size :
6
Enter 6 elements :
12 65 23 98 35 98
Before sorting the elements are : 12 65 23 98 35 98
After sorting the elements are : 12 23 35 65 98 98

<b>Test Case - 3</b>
<b>User Output</b>
Enter array size :
4
Enter 4 elements :
-23 -45 -12 -36
Before sorting the elements are : -23 -45 -12 -36
After sorting the elements are : -45 -36 -23 -12

<b>Test Case - 4</b>
<b>User Output</b>
Enter array size :
6
Enter 6 elements :
1 -3 8 -4 -2 5
Before sorting the elements are : 1 -3 8 -4 -2 5
After sorting the elements are : -4 -3 -2 1 5 8

S.No: 11

Exp. Name: **Write a C program to Sort given elements using Merge sort**

Date: 2023-05-08

**Aim:**

Write a program to **sort** (**Ascending order**) the given elements using **merge sort** technique.

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

Enter array size :

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

Enter array size : 5

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

Enter 5 elements :

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

Enter 5 elements : 34 67 12 45 22

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

Before sorting the elements are : 34 67 12 45 22  
After sorting the elements are : 12 22 34 45 67

**Note:** Do use the **printf()** function with a **newline** character (**\n**).

**Source Code:**

MergeSortMain.c

```

#include <stdio.h>

void main()
{
    int arr[15], i, n;
    printf("Enter array size : ");
    scanf("%d", &n);
    printf("Enter %d elements : ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Before sorting the elements are : ");
    display(arr, n);
    splitAndMerge(arr, 0, n - 1);
    printf("After sorting the elements are : ");
    display(arr, n);
}

void display(int arr[15], int n)
{
    int i;
    for(i=0;i<n;i++)
    printf("%d ",arr[i]);
    printf("\n");
}

void merge(int arr[15], int low, int mid, int high)
{
    int i=low,h=low,j=mid+1,k,temp[15];
    while(h<=mid&&j<=high)
    {
        if(arr[h]<=arr[j]){temp[i]=arr[h];
        h++;}
        else
        {
            temp[i]=arr[j];
            j++;
        }
        i++;
    }
    if(h>mid)
    {
        for(k=j;k<=high;k++)
        {
            temp[i]=arr[k];i++;
        }
    }
    else
    {
        for(k=h;k<=mid;k++)
    }
}

```

```

        i++;
    }

}

for(k=low;k<=high;k++)
{
    arr[k]=temp[k];

}
}

void splitAndMerge(int arr[15],
int low, int high)
{
    if(low<high)
    {
        int mid=(low+high)/2;
        splitAndMerge(arr,low,mid);
        splitAndMerge(arr,mid+1,high);
        merge(arr,low,mid,high);

    }
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size :
5
Enter 5 elements :
34 67 12 45 22
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67

Test Case - 2
User Output
Enter array size :
8
Enter 8 elements :
77 55 22 44 99 33 11 66
Before sorting the elements are : 77 55 22 44 99 33 11 66
After sorting the elements are : 11 22 33 44 55 66 77 99

Test Case - 3
User Output
Enter array size :
5

Enter 5 elements :

-32 -45 -67 -46 -14

Before sorting the elements are : -32 -45 -67 -46 -14

After sorting the elements are : -67 -46 -45 -32 -14

S.No: 12

Exp. Name: **Write a C program to sort given elements using Radix sort**

Date: 2023-05-14

**Aim:**

Write a program to **sort** (**ascending order**) the given elements using **radix sort** technique.

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

Enter array size :

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

Enter array size : 5

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

Enter 5 elements :

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

Enter 5 elements : 34 67 12 45 22

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

Before sorting the elements are : 34 67 12 45 22  
After sorting the elements are : 12 22 34 45 67

**Note:** Do use the **printf()** function with a **newline** character (**\n**).

**Source Code:**

RadixSortMain2.c

```

#include <stdio.h>
#include <conio.h>
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");
}

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;
}

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++)
    }

```

```

        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

S.No: 13

Exp. Name: **C program to performs all operations  
on singly linked list**

Date: 2023-05-13

**Aim:**

Write a program that uses functions to perform the following **operations on singly linked list**

- i) Creation
- ii) Insertion
- iii) Deletion
- iv) Traversal

**Source Code:**

```
singlelinkedlistalloperations.c
```

```

#include<stdio.h>
#include<stdlib.h>
struct node {
    int data;
    struct node *next;
} *head = NULL, *tail = NULL;
void insert();
void Delete();
void display();
void count();
typedef struct node *NODE;
NODE temp, newNode, ptr, ptr2;
int value;
void main()
{
    int option = 0;
    printf("Singly Linked List Example - All Operations\n");
    while(1) {
        printf("Options\n");
        printf("1 : Insert elements into the linked list\n");
        printf("2 : Delete elements from the linked list\n");
        printf("3 : Display the elements in the linked list\n");
        printf("4 : Count the elements in the linked list\n");
        printf("5 : Exit()\n");
        printf("Enter your option : ");
        scanf("%d",&option);
        if(option<=5) {
            switch(option) {
                case 1:
                    insert();
                    break;
                case 2:
                    Delete();
                    break;
                case 3:
                    display();
                    break;
                case 4:
                    count();
                    break;
                case 5:
                    exit(0);
            }
        }
        else {
            printf("Enter options from 1 to 5\n");
            break;
        }
    }
}

void insert() {
    printf("Enter elements for inserting into linked list : ");
    scanf("%d",&value);
}

```

```

if(head == NULL) {
    head = newNode;
    tail = newNode;
}

}
else {
    tail->next = newNode;
    tail = newNode;
}
}
void Delete() {
    int i = 1, j = 1, pos, spot, cnt = 0;
    temp = head, ptr2 = head;
    while(ptr2!=NULL) {
        cnt++;
        ptr2 = ptr2->next;
    }
    printf("Enter position of the element for deleteing the element :
");

    scanf("%d",&spot);
    while(i<=cnt) {
        if(i == spot){
            pos = spot;
            break;
        }
        i++;
    }
    if(pos != spot)
        printf("Invalid Position.\n");
    else
    {
        if(pos == 1)
        {
            head = head->next;
            free(temp);
        }
        else
        {
            while(j<pos)
            {
                ptr = temp;
                temp = temp->next;
                j++;
            }
            if(temp->next == NULL)
            {
                ptr->next = NULL;
                free(temp);
            }
            else
            {
                ptr->next = temp->next;
                free(temp);
            }
        }
    }
}

```

```

void display()
{
    temp = head;
    printf("The elements in the linked list are : ");
    while(temp != NULL)
    {
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
}
void count()
{
    int count = 0;
    temp = head;
    while(temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    printf("No of elements in the linked list
are : %d\n",count);
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1
<b>User Output</b>
Singly Linked List Example - All Operations
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
111
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
222

Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
333
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
444
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
3
The elements in the linked list are : 111 222 333 444
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
2
Enter position of the element for deleteing the element :
2
Deleted successfully
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
3
The elements in the linked list are : 111 333 444
Options

```
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
4
No of elements in the linked list are : 3
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
5
```

### **Test Case - 2**

#### **User Output**

```
Singly Linked List Example - All Operations
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
001
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
010
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
```

```
100
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
1
Enter elements for inserting into linked list :
101
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
3
The elements in the linked list are : 1 10 100 101
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
2
Enter position of the element for deleteing the element :
3
Deleted successfully
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
3
The elements in the linked list are : 1 10 101
Options
1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
4
No of elements in the linked list are : 3
Options
```

1 : Insert elements into the linked list
2 : Delete elements from the linked list
3 : Display the elements in the linked list
4 : Count the elements in the linked list
5 : Exit()
Enter your option :
5

S.No: 14

Exp. Name: **C program which performs all operations on double linked list.**

Date: 2023-05-17

**Aim:**

Write a C program that uses functions to perform the following **operations on double linked list**  
i) Creationii) Insertioniii) Deletioniv) Traversal

**Source Code:**

AllOperationsDLL.c

Page No: 44

ID: 224G1A05B1

2022-2026-CSE-B

Srinivasa Ramanujan Institute of Technology

```

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct dnode{
    struct dnode *prev;
    int data;
    struct dnode *next;
};
struct dnode *start = NULL;
void insert(int);
void remov(int);
void display();
int main(){
    int n, ch;
    do {
        printf("Operations on doubly linked list");
        printf("\n1. Insert \n2.Remove\n3. Display\n0. Exit");
        printf("\nEnter Choice 0-4? : ");
        scanf("%d", &ch);
        switch (ch) {
    case 1:
        printf("Enter number: ");
        scanf("%d", &n);
        insert(n);
        break;
    case 2:
        printf("Enter number to delete: ");
        scanf("%d", &n);
        remov(n);
        break;
    case 3:
        display();
        break;
    }
    } }
    while (ch != 0);
}
void insert(int num)
{
    struct dnode *nptr, *temp = start;
    nptr = malloc(sizeof(struct dnode));
    nptr->data = num;
    nptr->next = NULL;
    nptr->prev = NULL;
    if (start == NULL)
    {
        start = nptr;
    }
    else {
        while (temp->next != NULL) temp = temp->next;
        nptr->prev = temp;
        temp->next = nptr;
    }
}

```

```

struct dnode *temp = start;
while (temp != NULL)
{
    if (temp->data == num) {
        if (temp == start)
        {
            start = start->next;
            start->prev = NULL;

        } else
        {
            if (temp->next == NULL) temp->prev->next = NULL;
            else
            {
                temp->prev->next = temp->next;
                temp->next->prev = temp->prev;

            } free(temp);
        }
        return ;
    }
    temp = temp->next;
}

printf("%d not found.\n", num);

}
void display()
{
    struct dnode *temp = start;
    while (temp != NULL)
    {
        printf("%d\t", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
Operations on doubly linked list	
1.Insert	
2.Remove	
3.Display	
0.Exit	
Enter Choice 0-4?:	

1
Enter number:
15
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
1
Enter number:
16
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
1
Enter number:
17
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
1
Enter number:
18
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
3
15    16    17    18
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
2
Enter number to delete:
19
19 not found
Operations on doubly linked list
1.Insert

2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
3
15      16      17      18
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
2
Enter number to delete:
16
Operations on doubly linked list
1.Insert
2.Remove
3.Display
0.Exit
Enter Choice 0-4?:
0

S.No: 15

Exp. Name: **C program to which performs all operations on Circular linked list.**

Date: 2023-06-11

**Aim:**

Write a program that uses functions to perform the following **operations on Circular linked list**  
i)Creationii)insertioniii)deletioniv) Traversal

**Source Code:**

AlloperationsinCLL.c

Page No: 49

ID: 224G1A05B1

2022-2026-CSE-B

Srinivasa Ramanujan Institute of Technology

```

#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};

void insert();
void deletion();
void find();
void print();
struct node *head = NULL;
int main()
{
    int choice;
    printf("CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT\n");
    while(1)
    {
        printf("1.INSERT ");
        printf("2.DELETE ");
        printf("3.FIND ");
        printf("4.PRINT ");
        printf("5.QUIT\n");
        printf("Enter the choice: ");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:insert();break;
            case 2:deletion();break;
            case 3:find();break;
            case 4:print();break;
            case 5:exit(0);
        }
    }
}

void insert()
{
    int x,n;
    struct node *newnode,*temp = head, *prev;
    newnode = (struct node*)malloc(sizeof(struct node));
    printf("Enter the element to be inserted: ");
    scanf("%d", &x);
    printf("Enter the position of the element: ");
    scanf("%d", &n);
    newnode->data = x;
    newnode->next = NULL;
    if(head == NULL)
    {
        head = newnode;
        newnode->next = newnode;
    }
    else if(n == 1)
    {
        temp = head;
        newnode->next = temp;

```

```

        temp->next = newnode;
        head = newnode;
    }
    else
    {
        for(int i = 1; i < n-1; i++)
        {
            temp = temp->next;
        }
        newnode->next = temp->next;
        temp->next = newnode;
    }
}
void deletion()
{
    struct node *temp = head, *prev, *temp1 = head;
    int key, count = 0;
    printf("Enter the element to be deleted: ");
    scanf("%d", &key);
    if(temp->data == key){ prev = temp -> next;
    while(temp->next != head)
    {
        temp = temp->next;
    }
    temp->next = prev;
    free(head);
    head = prev;
    printf("Element deleted\n");
    }
    else
    {
        while(temp->next != head)
        {
            if(temp->data == key)
            {
                count += 1;
                break;
            }
            prev = temp;
            temp = temp->next;
        }
        if(temp->data == key)
        {
            prev->next = temp->next;
            free(temp);
            printf("Element deleted\n");
        }
        else
        {
            printf("Element does not exist...!\n");
        }
    }
}
void find()
{

```

```

printf("Enter the element to be searched: ");
scanf("%d", &key);
while(temp->next != head)
{
    if(temp->data == key)
    {
        count = 1;
        break;
    }
    temp = temp->next;
}
if (count == 1)
    printf("Element exist...!\n");
else
{
    if(temp->data == key)
        printf("Element exist...!\n");
    else
        printf("Element does not exist...!\n");
}
}

void print()
{
    struct node *temp = head;
    printf("The list element are: ");
    while(temp->next != head)
    {
        printf("%d -> ",temp->data);
        temp = temp->next;
    }
    printf("%d -> ", temp->data) ;
    printf("\n");
}

```

## Execution Results - All test cases have succeeded!

Test Case - 1
<b>User Output</b>
CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT
Enter the choice:
1
Enter the element to be inserted:
12
Enter the position of the element:
1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT
Enter the choice:
1
Enter the element to be inserted:
14

Enter the position of the element:

2

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

1

Enter the element to be inserted:

15

Enter the position of the element:

3

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

4

The list element are: 12 -> 14 -> 15 ->

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

2

Enter the element to be deleted:

14

Element deleted

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

4

The list element are: 12 -> 15 ->

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

3

Enter the element to be searched:

12

Element exist...!

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

5

### Test Case - 2

#### User Output

CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

1

Enter the element to be inserted:

54

Enter the position of the element:

1

1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT

Enter the choice:

2

Enter the element to be deleted: