S.No: 1	Exp. Name: ***C program to insert and delete the element of one dimensional array.***	Date: 2023-06-11

ID: 22K61A05B4

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

Aim:

Write C program to insert and delete the element of one dimensional array

Source Code:

array.c

```
#include<stdio.h>
void main()
{
   int n,i,pos,ele,posi,a[10];
   printf("Enter the size of the array: ");
   scanf("%d",&n);
   printf("Enter the elements of the array:\n");
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   printf("Enter the position where you want to insert an element: ");
   scanf("%d",&pos);
   printf("Enter the value to insert: ");
   scanf("%d",&ele);
   for(i=n-1;i>=pos;i--)
   {
           a[i+1]=a[i];
   }
   a[pos]=ele;
   printf("Element inserted successfully!\nEnter the position of the element you want to
delete: ");
   scanf("%d",&posi);
   for(i=posi;i<n;i++)
   {
        a[i]=a[i+1];
   }
   printf("Element deleted successfully!\nUpdated array:\n");
   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 the size of the array:
4
Enter the elements of the array:
1 5 2 3

Enter the position where you want to insert an element:
2
Enter the value to insert:
11
Element inserted successfully!
Enter the position of the element you want to delete:
4
Element deleted successfully!
Updated array:
1 5 11 2

Test Case - 2
User Output
Enter the size of the array:
7
Enter the elements of the array:
11 22 33 44 55 66 77
Enter the position where you want to insert an element:
1
Enter the value to insert:
88
Element inserted successfully!
Enter the position of the element you want to delete:
7
Element deleted successfully!
Updated array:
11 88 22 33 44 55 66

S.No: 2	Exp. Name: ***C program to create dynamic memory allocation using malloc( )***	**Date: 2023-06-09**

Aim:

Write a C program to create dynamic memory allocation using malloc( )

Source Code:

malloc.c

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

int main() {
int *p,n,i,sum=0;
float avg;
printf("Enter the number of integers: ");
scanf("%d",&n);
p=(int*)malloc(n*sizeof(int));
if(p==0)
{
        printf("Sorry!Unable to allocate memory");
        exit(0);
}
printf("Enter %d integers:\n",n);
for(i=0;i<n;i++)
{
scanf("%d",p+i) ;
sum+=*(p+i);
}
avg=(float)sum/n;
printf("The sum of the integers is %d\n",sum);
printf("The average of the integers is %.2f",avg);
printf("\n");
free(p);
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of integers:
3
Enter 3 integers:
1 5 3
The sum of the integers is 9
The average of the integers is 3.00

Test Case - 2

User Output
Enter the number of integers:
5
Enter 5 integers:
1 2 3 4 5
The sum of the integers is 15
The average of the integers is 3.00

S.No: 3	Exp. Name: ***C program to create dynamic memory allocation using calloc( )***	Date: 2023-06-08

ID: 22K61A05B4

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

Aim:

Write a C program to create dynamic memory allocation using calloc( )

Source Code:

calloc.c

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

int main() {
 int *p,n,a[10],sum=0,i,j=1;
 printf("Enter the number of elements: ");
 scanf("%d",&n);
 p=(int*)calloc(n,sizeof(int));
 for(i=0;i<n;i++)
 {
        printf("Enter element %d: ",j);
        scanf("%d",p+i);
        sum=sum+*(p+i);
        j++;
 }
 printf("The sum of the array is %d.\n",sum);

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of elements:
5
Enter element 1:
1
Enter element 2:
2
Enter element 3:
3
Enter element 4:
4
Enter element 5:
5
The sum of the array is 15.

Test Case - 2
User Output

```
Enter the number of elements:
4
Enter element 1:
11
Enter element 2:
22
Enter element 3:
33
Enter element 4:
44
The sum of the array is 110.
```

S.No: 4	Exp. Name: ***Write a C program to Search an element using Linear Search process***	Date: 2023-04-01

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**".

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

Source Code:

Program509.c

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

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

Test Case - 3
User Output
Enter value of n :
4
Enter element for a[0] :
452
Enter element for a[1] :
356
Enter element for a[2] :
754
Enter element for a[3] :
127
Enter key element :
127
The key element 127 is found at the position 3

Test Case - 4
User Output
Enter value of n :
3
Enter element for a[0] :
5
Enter element for a[1] :
7
Enter element for a[2] :
3
Enter key element :

4
The key element 4 is not found in the array

Test Case - 5
User Output
Enter value of n :
3
Enter element for a[0] :
11
Enter element for a[1] :
45
Enter element for a[2] :
37
Enter key element :
25
The key element 25 is not found in the array

S.No: 5	Exp. Name: ***Write a Program to Search an element using Linear Search and Recursion***	**Date: 2023-04-13**

ID: 22K61A05B4

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

Aim:

Write a program to `search` the given element from a list of elements with `linear search` technique using **recursion**.

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 : 6
```

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

```
Enter 5 elements :
```

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

```
Enter 5 elements : 12 54 32 9 26
```

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

```
Enter a key element :
```

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

```
Enter a key element : 9
```

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

```
The key element 9 is found at position : 3
```

Similarly, if the key element is given as `18` for the above example then the program should print the output as:

```
The key element 18 is not found
```

Note: Write the functions **read()** and **linearSearch()** in `Program911a.c`

Source Code:

`Program911.c`

```c
#include <stdio.h>
#include "Program911a.c"
void main() {
        int a[20], n, pos, key;
        printf("Enter n value : ");
        scanf("%d", &n);
        read(a, n);
        printf("Enter a key element : ");
        scanf("%d", &key);
        pos = linearSearch(a, 0, n - 1, key);
        if (pos == -1) {
                printf("The key element %d is not found\n", key);
        } else {
                printf("The key element %d is found at position : %d\n", key, pos);
        }
}
```

Program911a.c

```c
int read(int a[],int n)
{
        int i;
        printf("Enter %d elements : ",n);
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}
int linearSearch(int a[],int i,int n,int pos)
{
        if(i==n+1)
        return -1;
        else if(a[i]==pos)
        return i;
        else
        return linearSearch(a,i+1,n,pos);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n value :
4
Enter 4 elements :
10 20 15 12
Enter a key element :
15
The key element 15 is found at position : 2

Test Case - 2

User Output
Enter n value :
6
Enter 6 elements :
2 6 4 1 3 7
Enter a key element :
5
The key element 5 is not found

Test Case - 3
User Output
Enter n value :
5
Enter 5 elements :
11 44 33 55 22
Enter a key element :
11
The key element 11 is found at position : 0

Test Case - 4
User Output
Enter n value :
5
Enter 5 elements :
99 65 78 34 27
Enter a key element :
26
The key element 26 is not found

S.No: 6	Exp. Name: ***Write a C program to Search an element using Binary Search process***	Date: 2023-04-14

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**".

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

Source Code:

Program510.c

```c
#include<stdio.h>
void main()
{
        int a[10],n,temp,key;
        int lb,ub,mid,pos=-1,i,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-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",i);
        for(i=0;i<n;i++)
        {
                printf("\nValue of a[%d] = %d",i,a[i]);
        }
        lb=0;
        ub=n-1;
        while(lb<=ub)
        {
                mid=(lb+ub)/2;
                if(a[mid]==key)
                {
                        pos=mid;
                        break;
                }
                else
                {
                        if(a[mid]<key)
                        lb=mid+1;
                        else
                        ub=mid-1;
                }
        }
        if(pos!=-1)
        printf("\nThe key element %d is found at the position %d",key,pos);
        else
        printf("\nThe Key element %d is not found in the array",key);
        printf("\n");
}
```

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: 7	Exp. Name: ***Write a Program to Search an element using Binary Search and Recursion***	**Date: 2023-04-14**

Aim:

Write a program to `search` the given element from a list of elements with `binary search` technique using **recursion**.

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 : 5
```

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

```
Enter 5 elements :
```

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

```
Enter 5 elements : 33 55 22 44 11
```

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

```
After sorting the elements are : 11 22 33 44 55
```

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

```
Enter key element :
```

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

```
Enter key element : 11
```

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

```
The given key element 11 is found at position : 0
```

Similarly, if the key element is given as `18` for the above example then the program should print the output as:

```
The given key element 18 is not found
```

Note: Write the functions **read()**, **bubbleSort()**, **display()** and **binarySearch()** in `Program912a.c`

Source Code:

`Program912.c`

```c
#include <stdio.h>
#include "Program912a.c"
void main() {
        int a[20], n, key, flag;
        printf("Enter value of n : ");
        scanf("%d", &n);
        read(a, n);
        bubbleSort(a, n);
        printf("After sorting the elements are : ");
        display(a, n);
        printf("Enter key element : ");
        scanf("%d", &key);
        flag = binarySearch(a, 0, n - 1, key);
        if (flag == -1) {
                printf("The given key element %d is not found\n", key);
        } else {
                printf("The given key element %d is found at position : %d\n", key, flag);
        }
}
```

Program912a.c

```
void read(int a[],int n)
{
        int i;
        printf("Enter %d elements : ",n);
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}
void bubbleSort(int a[],int n)
{
        int i,j,temp;
        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;
                        }
                }
        }
}
void display(int a[],int n)
{
        int i;
        for(i=0;i<n;i++)
        printf("%d ",a[i]);
        printf("\n");
}
int binarySearch(int a[],int l,int h,int key)
{
        int mid,flag;
        if(l>h)
        return -1;
        else
        {
                mid=(l+h)/2;
                if(a[mid]==key)
                {
                        flag=mid;
                        return flag;
                }
                else
                {
                        if(a[mid]<key)
                        binarySearch(a,mid+1,h,key);
                        else
                        binarySearch(a,l,mid-1,key);
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :
5
Enter 5 elements :
33 55 22 44 11
After sorting the elements are : 11 22 33 44 55
Enter key element :
11
The given key element 11 is found at position : 0

Test Case - 2
User Output
Enter value of n :
4
Enter 4 elements :
23 67 45 18
After sorting the elements are : 18 23 45 67
Enter key element :
24
The given key element 24 is not found

Test Case - 3
User Output
Enter value of n :
6
Enter 6 elements :
10 20 18 9 11 15
After sorting the elements are : 9 10 11 15 18 20
Enter key element :
18
The given key element 18 is found at position : 4

S.No: 8	Exp. Name: ***Write a C program to Sort the given elements in Ascending order using Bubble Sort***	**Date: 2023-04-14**

Aim:

Write a program to `sort` (`Ascending order`) 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
```

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

Source Code:

Program504.c

```c
#include<stdio.h>
int main()
{
        int n,a[20],i,j,temp;
        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] = ",i);
                printf("%d",a[i]);
                {
                        printf("\n");
                }
        }
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-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] = ",i);
                printf("%d",a[i]);
                {
                        printf("\n");
                }
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter value of n :

5
Enter element for a[0] :
2
Enter element for a[1] :
7
Enter element for a[2] :
6
Enter element for a[3] :
4
Enter element for a[4] :
1
Before sorting the elements in the array are
Value of a[0] = 2
Value of a[1] = 7
Value of a[2] = 6
Value of a[3] = 4
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] = 4
Value of a[3] = 6
Value of a[4] = 7

Test Case - 2
User Output
Enter value of n :
4
Enter element for a[0] :
28
Enter element for a[1] :
34
Enter element for a[2] :
26
Enter element for a[3] :
29
Before sorting the elements in the array are
Value of a[0] = 28
Value of a[1] = 34
Value of a[2] = 26
Value of a[3] = 29
After sorting the elements in the array are
Value of a[0] = 26
Value of a[1] = 28
Value of a[2] = 29
Value of a[3] = 34

Test Case - 3
User Output
Enter value of n :
8
Enter element for a[0] :
7
Enter element for a[1] :
3
Enter element for a[2] :
9
Enter element for a[3] :
2
Enter element for a[4] :
5
Enter element for a[5] :
4
Enter element for a[6] :
6
Enter element for a[7] :
1
Before sorting the elements in the array are
Value of a[0] = 7
Value of a[1] = 3
Value of a[2] = 9
Value of a[3] = 2
Value of a[4] = 5
Value of a[5] = 4
Value of a[6] = 6
Value of a[7] = 1
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] = 4
Value of a[4] = 5
Value of a[5] = 6
Value of a[6] = 7
Value of a[7] = 9

Test Case - 4
User Output
Enter value of n :
4
Enter element for a[0] :
-23
Enter element for a[1] :
-14

Enter element for a[2] :
-56
Enter element for a[3] :
-35
Before sorting the elements in the array are
Value of a[0] = -23
Value of a[1] = -14
Value of a[2] = -56
Value of a[3] = -35
After sorting the elements in the array are
Value of a[0] = -56
Value of a[1] = -35
Value of a[2] = -23
Value of a[3] = -14

Test Case - 5
User Output
Enter value of n :
5
Enter element for a[0] :
28
Enter element for a[1] :
45
Enter element for a[2] :
-1
Enter element for a[3] :
-5
Enter element for a[4] :
2
Before sorting the elements in the array are
Value of a[0] = 28
Value of a[1] = 45
Value of a[2] = -1
Value of a[3] = -5
Value of a[4] = 2
After sorting the elements in the array are
Value of a[0] = -5
Value of a[1] = -1
Value of a[2] = 2
Value of a[3] = 28
Value of a[4] = 45

S.No: 9	Exp. Name: ***Write a C program to Sort given elements using Quick sort***	Date: 2023-04-27

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>
#include "QuickSortFunctions.c"
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);
        quickSort(arr, 0, n - 1);
        printf("After sorting the elements are : ");
        display(arr, n);
}
```

QuickSortFunctions.c

```
void display(int arr[15],int n){
        printf("");
        for(int i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
        printf("\n");
}
int partition(int arr[15],int lb,int ub){
        int pivot=arr[ub];
        int i=lb-1;
        for(int j=lb;j<ub;j++)
        {
                if(arr[j]<=pivot)
                {
                        i++;
                        swap(&arr[i],&arr[j]);
                }
        }
        swap(&arr[i+1],&arr[ub]);
        return i+1;
}
void quickSort(int arr[15],int low, int high)
{
        if(low<high)
        {
                int pi=partition(arr,low,high);
                quickSort(arr,low,pi-1);
                quickSort(arr,pi+1,high);
        }
}
void swap(int *a,int *b)
{
        int temp=*a;
        *a=*b;
        *b=temp;
}
```

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: 10	Exp. Name: ***Write a C program to Sort given elements using Insertion sort***	Date: 2023-04-15

Aim:

Write a program to `sort` (`Ascending order`) 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
```

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

Source Code:

Program505.c

```c
#include<stdio.h>
int main()
{
        int a[20],n,i,j,temp;
        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=1;i<n;i++)
        {
                j=i;
                while(j>0 && a[j-1]>a[j])
                {
                        temp=a[j];
                        a[j]=a[j-1];
                        a[j-1]=temp;
                        j--;
                }
        }
        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 :
5
Enter element for a[0] :
7
Enter element for a[1] :
33
Enter element for a[2] :
12
Enter element for a[3] :
56
Enter element for a[4] :
9
Before sorting the elements in the array are

```
Value of a[0] = 7
Value of a[1] = 33
Value of a[2] = 12
Value of a[3] = 56
Value of a[4] = 9
After sorting the elements in the array are
Value of a[0] = 7
Value of a[1] = 9
Value of a[2] = 12
Value of a[3] = 33
Value of a[4] = 56
```

S.No: 11	Exp. Name: ***Write a C program to Sort given elements using Merge sort***	Date: 2023-04-15

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>
#include "MergeSortFunctions.c"
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);
}
```

MergeSortFunctions.c

```
void display(int arr[15], int n) {
        for(int i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
        printf("\n");
        }
void merge(int arr[15], int low, int mid, int high) {
        int b[15],i,j,k;
        i=low;
        j=mid+1;
        k=low;
        while(i<=mid&&j<=high)
        {
                if(arr[i]<=arr[j])
                {
                        b[k]=arr[i];
                        i++;
                        k++;
                }
                else
                {
                        b[k]=arr[j];
                        j++;
                        k++;
                }
        }
        while(i<=mid)
        {
                b[k]=arr[i];
                k++;
                i++;
        }
        while(j<=high)
        {
                b[k]=arr[j];
                k++;
                j++;
        }
        for(i=low;i<=high;i++)
        {
                arr[i]=b[i];
        }
}
void splitAndMerge(int arr[15], int low,int high) {
        if(low<high)
        {
                int m = (low+high)/2;
                splitAndMerge(arr,low,m);
                splitAndMerge(arr,m+1,high);
                merge(arr,low,m,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 Create a Singly Linked List***	Date: 2023-05-13

Aim:

In the below `singly linked list` program we have two files, one file contains the **main program** and the other file contains the **functions**, to be implemented by the **user**.

Here the user has to implement the code for two functions `addNodes()` and `traverseList()`.

The `addNodes()` function creates a new list and adds elements to the list until delimiter `-1` is occurred.

Fill in the missing code in the below functions `addNodes(NODE first, int x)` and `traverseList(NODE first)` in the file `CreateAndAddNodes.c`.

Source Code:

SingleLL1.c

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

#include "CreateAndAddNodes.c"

void main() {
        NODE first = NULL;
        int x;
        printf("Enter elements up to -1 : ");
        scanf("%d", &x);
        while (x != -1) {
                first = addNodes(first, x);
                scanf("%d", &x);
        }
        if (first == NULL) {
                printf("Single Linked List is empty\n");
        } else {
                printf("The elements in SLL are : ");
                traverseList(first);
        }
}
```

CreateAndAddNodes.c

```
struct node {
        int data;
        struct node *next;
};
typedef struct node *NODE;
NODE first = NULL;
NODE createNode() {
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        temp -> next = NULL;
        return temp;

}

NODE addNodes(NODE first, int x)
{
        NODE temp;
        temp = createNode();
        temp ->data = x;
        if(first == NULL)
        {
                first = temp;
        }
        else
        {
                NODE lastNode = first;
                while(lastNode->next!=NULL)
                {
                        lastNode=lastNode->next;
                }
                lastNode->next=temp;
        }
return first;
}

void traverseList(NODE first)
{
        if(first==NULL)
        {
                printf("List is Empty\n");
        }
        else
        {
                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
Enter elements up to -1 :
9 18 27 36 45 -1
The elements in SLL are : 9 --> 18 --> 27 --> 36 --> 45 --> NULL

Test Case - 2
User Output
Enter elements up to -1 :
12 14 19 23 -1
The elements in SLL are : 12 --> 14 --> 19 --> 23 --> NULL

S.No: 13	Exp. Name: ***Write a C program to Insert an element at Begin and Count number of Nodes in Singly Linked List***	Date: 2023-04-29

Aim:

Fill in the missing code in the below functions `insertAtBegin(NODE first, int x)` and `count(NODE first)` in the file `InsAtBeginAndCount.c`.

The `insertAtBegin(NODE first, int x)` function inserts a new node at the beginning of the singly linked list.

The algorithm for `insertAtBegin(NODE first, int x)` is as follows:

```
Step-1: Allocate memory to the node temp.
Step-2: Store an integer value into data field of node temp.
Step-3: Assign the address contained in the first node to the next field of temp.
Step-4: Now treat the temp node as first node.
Step-5: Finally return the first node.
```

The `count(NODE first)` function counts the number of nodes linked in a singly linked list.

The algorithm for `count(node first)` is as follows:

```
Step-1: Assign the address contained in first node to temp node.
Step-2: Initialize a variable sum to 0 (zero).
Step-3: Repeat Step-4 and Step-5 until temp reaches the NULL.
Step-4: Increment the sum by 1.
Step-5: Move to the next node by placing the address of the next node in temp node.
Step-6: Finally return sum.
```

Source Code:

SingleLL2.c

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

#include "InsAtBeginAndCount.c"

void main() {
        NODE first = NULL;
        int x, op;
        while(1) {
                printf("1.Insert At Begin 2.Count Number of Nodes 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("The number of nodes in a SLL are : %d\n",
count(first));
                                        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);
                }
        }
}
```

InsAtBeginAndCount.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 count(NODE first) {
NODE temp=first;
int sum=0;;
while(temp !=NULL){
        sum++;
        temp = temp -> next;
}
return sum;
}

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.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
10
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :

1
Enter an element :
20
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
30
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
2
The number of nodes in a SLL are : 3
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 30 --> 20 --> 10 --> NULL
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
40
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
2
The number of nodes in a SLL are : 4
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 40 --> 30 --> 20 --> 10 --> NULL
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
4

Test Case - 2
User Output
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
99
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
89
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
3

The elements in SLL are : 89 --> 99 --> NULL
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
2
The number of nodes in a SLL are : 2
1.Insert At Begin 2.Count Number of Nodes 3.Traverse the List 4.Exit
Enter your option :
4

S.No: 14	Exp. Name: ***Write a C program to Delete an element at Begin from Singly Linked List***	Date: 2023-05-13

Aim:

Fill in the missing code in the below function `deleteAtBegin(NODE first)`, which deletes the node at the beginning of singly linked list.

Source Code:

SingleLL5.c

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

#include "DelAtBegin.c"

void main() {
        NODE first = NULL;
        int x, op;
        while(1) {
                printf("1.Insert At Begin 2.Delete at Begin 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: if (first == NULL) {
                                                printf("Single Linked List is empty so
deletion is not possible\n");
                                        } else {
                                                first = deleteAtBegin(first);
                                        }
                                        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);
                }
        }
}
```

DelAtBegin.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 FIRST = NULL;
NODE insertAtBegin(NODE first, int x)
{
        NODE temp;
        temp=createNode();
        temp->data=x;
        temp->next=first;
        first=temp;
        return first;
}

NODE deleteAtBegin(NODE first)
{
        if(first==NULL)
        {
                printf("List is Empty,Deletion is not possible\n");
        }
        else
        {
                NODE temp = first;
                first = first->next;
                printf("The deleted element from SLL : %d\n",temp->data);
                free(temp);
                return first;
        }

}

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.Delete at Begin 3.Traverse the List 4.Exit
Enter your option :
1
Enter an element :
10
1.Insert At Begin 2.Delete at Begin 3.Traverse the List 4.Exit
Enter your option :
3
The elements in SLL are : 10 --> NULL
1.Insert At Begin 2.Delete at Begin 3.Traverse the List 4.Exit
Enter your option :
2
The deleted element from SLL : 10
1.Insert At Begin 2.Delete at Begin 3.Traverse the List 4.Exit
Enter your option :
3
Single Linked List is empty
1.Insert At Begin 2.Delete at Begin 3.Traverse the List 4.Exit
Enter your option :
4

S.No: 15	Exp. Name: ***Write the code to reverse elements of a single linked list***	Date: 2023-06-11

Aim:

Write a C program to reverse elements of a single linked list.

Source Code:

reverseElements.c

```
#include<stdio.h>
#include<stdlib.h>
int main(){
struct node
{
        int data;
        struct node *next;
};
typedef struct node * NODE;
NODE head,temp,newnode;
int i ,n, s;
head=NULL;
printf("Enter the total number of nodes: ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
        newnode=(NODE)malloc(sizeof(struct node));
printf("Enter the data of node %d: ",i);
scanf("%d",&newnode->data);
newnode->next=NULL;
if(head==NULL)
{
        temp=head=newnode;
}
else
{
        temp->next=newnode;
        temp=newnode;
}
}
temp=head;
printf("Data in the list\n");
while(temp!=NULL)
{
        printf("Data = %d\n",temp->data);
        temp=temp->next;
}
printf("Press 1 to reverse the order of singly linked list\n");
scanf("%d",&s);
if(s==1)
{
        NODE pre,current,next;
        pre=NULL;
        current=next=head;
        while(next!=NULL)
        {
                next=next->next;
                current->next=pre;
                pre=current;
                current=next;
        }
        head=pre;
}
temp=head;
printf("Data in the list\n");
```

```
        printf("Data = %d\n",temp->data);
        temp=temp->next;
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the total number of nodes:
5
Enter the data of node 1:
26
Enter the data of node 2:
394
Enter the data of node 3:
145
Enter the data of node 4:
624
Enter the data of node 5:
731
Data in the list
Data = 26
Data = 394
Data = 145
Data = 624
Data = 731
Press 1 to reverse the order of singly linked list
1
Data in the list
Data = 731
Data = 624
Data = 145
Data = 394
Data = 26

Test Case - 2
User Output
Enter the total number of nodes:
8
Enter the data of node 1:
21
Enter the data of node 2:
94
Enter the data of node 3:
214

```
Enter the data of node 4:
24
Enter the data of node 5:
45
Enter the data of node 6:
694
Enter the data of node 7:
321
Enter the data of node 8:
356
Data in the list
Data = 21
Data = 94
Data = 214
Data = 24
Data = 45
Data = 694
Data = 321
Data = 356
Press 1 to reverse the order of singly linked list
1
Data in the list
Data = 356
Data = 321
Data = 694
Data = 45
Data = 24
Data = 214
Data = 94
Data = 21
```

S.No: 16	Exp. Name: ***Write a C program to implement different Operations on Queue using Array representation***	Date: 2023-06-10

Aim:

Write a program to implement queue using **arrays**.

```
Sample Input and 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
```

Source Code:

QueueUsingArray.c

```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

```
#include<stdio.h>
#include<limits.h>
#define Max 100
int queue [Max];
unsigned int Size=0,rear=Max-1,front=0;
int enqueue(int x)
{ if(Size==Max)
   {printf("Queue is overflow.\n");}
   rear=(rear+1)%Max;
   Size++;
   queue[rear]=x;
   printf("Successfully inserted.\n");
   return -1;
}
int dequeue()
{
        int x;
        if(Size==0)
        {printf("Queue is underflow.\n");}
        else
        {
                x=queue[front];
                front=(front=1)%Max;
                if(Size!=0)
                {printf("Deleted element = %d\n",x);
                Size--;
                }
        }
        return x;
}
int display()
{ int i;
  if(Size==0)
  {printf("Queue is empty.\n");}
  else
  { printf("Elements in the queue : ");
    for(i=front;i<=rear;i++)
    {printf("%d ",queue[i]);}
    printf("\n");
  }
}
int* size()
{printf("Queue size : %d\n",Size);}
int isEmpty()
{ if(Size==0)
  {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 :
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 - 2

User Output
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 :
2
Deleted element = 25
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 :
1
Enter element :
65
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
3
Elements in the queue : 65
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 = 65
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 :

63
Successfully inserted.
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
5
Queue size : 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
Enter your option :
6

S.No: 17	Exp. Name: ***Write a C program to implement different Operations on Queue using Linked Lists***	Date: 2023-06-10

Aim:

Write a program to implement queue using **linked lists**.

```
Sample Input and Output:
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 57
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 1
    Enter element : 87
    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 : 3
    Elements in the queue : 57 87
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted value = 57
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 2
    Deleted value = 87
    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 : 5
    Queue size : 0
    1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
    Enter your option : 6
```

Source Code:

QueueUsingLL.c

ID: 22K61A05B4
2022-2026-CSE-B
Sasi Institute of Technology and Engineering (Autonomous)

```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

```
#include<stdio.h>
#include<stdlib.h>
struct stack
{
        int data;
        struct stack *next;
};
typedef struct stack *stk;
stk back,top=NULL;
void dequeue()
{
        stk temp;
        temp=top;
        if(temp==NULL)
        printf("Queue is underflow.\n");
        else
        {
                if(top==back)
                top=back=NULL;
                else
                top=top->next;
                printf("Deleted value = %d\n",temp->data);
                free(temp);
        }
}
void size()
{
        stk temp;
        int count=0;
        temp=top;
        while(temp!=NULL)
        {
                count++;
                temp=temp->next;
        }
        printf("Queue size : %d\n",count);
}
void isEmpty()
{
        if(top==NULL)
        printf("Queue is empty.\n");
        else
        printf("Queue is not empty.\n");
}
stk enqueue(int x)
{
        stk temp;
        temp=(stk)malloc(sizeof(struct stack));
        temp->data=x;
        temp->next=NULL;
        if(top==NULL)
        top=temp;
        else
        back->next=temp;
        back=temp;
```

```
void display()
{
        stk temp=top;
        if(temp==NULL)
        printf("Queue is empty.\n");
        else
        {
                printf("Elements in the queue : ");
                while(temp!=NULL)
                {
                        printf("%d ",temp->data);
                        temp=temp->next;
                }
                printf("\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

S.No: 18	Exp. Name: ***Write a C program to implement different Operations on Stack using Array representation***	Date: 2023-06-11

Aim:

Write a program to implement `stack` using **arrays**.

```
Sample Input and Output:
    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 : 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 : 1
    Enter element : 25
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 26
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Elements of the stack are : 26 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Popped value = 26
    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 : 5
    Peek value = 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 6
```

Source Code:

StackUsingArray.c

```c
#include <stdio.h>
#include <stdlib.h>
#define STACK_MAX_SIZE 10
#include "StackOperations.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);
                }
        }
}
```

StackOperations.c

```
int st[STACK_MAX_SIZE],top=-1;
void push(int ele)
{
        if(top==STACK_MAX_SIZE-1)
        printf("Stack is overflow.\n");
        else
        {
                top++;
                st[top]=ele;
                printf("Successfully pushed.\n");
        }
}
void pop()
{
        if(top==-1)
        printf("Stack is underflow.\n");
        else
        {
                printf("Popped value = %d\n",st[top]);
                top--;
        }
}
void display()
{
        int i;
        if(top==STACK_MAX_SIZE-1)
    printf("Stack is full.\n");
    else if(top==-1)
    printf("Stack is empty.\n");
    else
    {
        printf("Elements of the stack are : ");
        for(i=top;i>-1;i--)
        {
                printf("%d ",st[i]);
        }printf("\n");
    }
}
void isEmpty()
{
        if(top==STACK_MAX_SIZE-1)
        printf("Stack is full.\n");
        else if(top!=-1)
        printf("Stack is not empty.\n");
        else
        printf("Stack is empty.\n");
}
void peek()
{
        if(top==-1)
        printf("Stack is underflow.\n");
        else
        printf("Peek value = %d\n",st[top]);
}
```

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 :
10
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
20
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
30
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 30 20 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 30
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 30
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Popped value = 20
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Elements of the stack are : 10
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Peek value = 10
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 :
2
Popped value = 10
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 :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
6

S.No: 19	Exp. Name: ***Write a C program to implement different Operations on Stack using Linked Lists***	Date: 2023-06-11

Aim:

Write a program to implement `stack` using **linked lists**.

```
Sample Input and 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
```

Source Code:

StackUsingLL.c

```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 *link;
};
struct node *top=NULL;
void push(int item)
{
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=item;
        if(top==NULL)
        temp->link=NULL;
        else
        temp->link=top;
        top=temp;printf("Successfully pushed.\n");
}
void pop()
{
        struct node *temp;
        temp=top;
        if(temp==NULL)
        printf("Stack is underflow.\n");
        else
        {
                printf("Popped value = %d\n",temp->data);
                top=top->link;
                free(temp);
        }
}
void display()
{
        struct node *temp;
        int count=0;
        temp=top;
        if(temp==NULL)
        printf("Stack is empty.\n");
        else
        {
                printf("Elements of the stack are : ");
                while(temp!=NULL)
                {
                        printf("%d ",temp->data);
                        temp=temp->link;
                }
                printf("\n");
        }
}
void isEmpty()
{
        if(top==NULL)
        printf("Stack is empty.\n");
        else
        printf("Stack is not empty.\n");
}
```

```
        struct node *temp;
        temp=top;
        if(temp==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