S.No: 1	Exp. Name: ***C program to insert and delete the element of one dimensional array.***	Date: 2023-04-26

ID: 22K61A05C2

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>
int main()
{ int a[100],n,i,key,pos;
  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",&key);
  for(i=n-1;i>=pos;i--)
  a[i+1]=a[i];
  a[pos]=key;
  printf("Element inserted successfully!\n");
  printf("Enter the position of the element you want to delete: ");
  scanf("%d",&pos);
  for(i=pos;i<=n;i++)
  a[i]=a[i+1];
  printf("Element deleted successfully!\n");
  printf("Updated 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: 2023-05-13

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\n");
   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);//dynamically allocate memory using malloc()
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: 2023-05-13

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,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: 2023-03-25

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

```
#include<stdio.h>
int main()
{ int a[20],i,n,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: 2023-04-26**

ID: 22K61A05C2

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
void 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 key)
{ int pos=0;
  if(i>=n)
  return -1;
  else
  { if(a[i]==key)
      { pos=i;
        return pos;
      }
      else
      return linearSearch(a,i+1,n,key);
  }
}
```

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: 2023-04-08

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

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>
int main()
{ int a[20],i,n,pos=-1,key,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("Enter key element : ");
  scanf("%d",&key);
for(i=0;i<n;i++)
   { temp=a[i];
        j=i;
     while(j>0&&temp<a[j-1])
    {
      a[j]=a[j-1];
      j=j-1;
    }
    a[j]=temp;
  }
  printf("After sorting the elements in the array are\n");
  for(i=0;i<n;i++)
  { printf("Value of a[%d] = %d\n",i,a[i]);
  }
  for(i=0;i<n;i++)
  {if(a[i]==key)
    { 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] :
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: 2023-04-26**

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=i+1;j<n;j++)
   { if(a[i]>a[j])
   {  temp=a[i];
      a[i]=a[j];
      a[j]=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 u,int key)
{
        int mid,flag;
        if(l>u)
        return -1;
        else
        { mid=(l+u)/2;
          if(a[mid]==key)
          { flag=mid;
            return flag;
          }
          else
          { if(a[mid]<key)
            binarySearch(a,mid+1,u,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: 2023-04-08

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

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: 2023-04-08

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

void swap(int *a,int *b)
{ int temp=*a;
       *a=*b;
       *b=temp;
}

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=i+1;
              swap(&arr[i],&arr[j]);
            }
        }
        swap(&arr[i+1],&arr[ub]);
        return i+1;
}
void quickSort(int a[],int lb,int ub)
{ if(lb<ub)
   { int pi=partition(a,lb,ub);
      quickSort(a,lb,pi-1);
      quickSort(a,pi+1,ub);
   }
}
```

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: 2023-04-01

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

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

```
#include<stdio.h>
int main()
{ int n,a[50],i,temp,j;
  printf("Enter value of n : ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  { printf("Enter element for a[%d] : ",i);
    scanf("%d",&a[i]);
  }
  printf("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++)
   { temp=a[i];
       j=i;
      while(j>0&&temp<a[j-1])
      { a[j]=a[j-1];
         j=j-1;
      }
      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 :
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) {
        int i,b[15];
        for(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];
        int 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) {
        int i,mid;
        if(low<high)
        { 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 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 createNode() {
     NODE temp;
     temp=(NODE)malloc(sizeof(struct node));
     temp->next=NULL;
     return temp;
}
NODE first=NULL;
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");
  }
  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: 2023-05-26

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 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");
        }
        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: 2023-05-16

Aim:

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

Source Code:

reverseElements.c

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        int info;
        struct node*next;
};
struct node*start=NULL;
struct node*new_node();
void create();
void display();
void reverse();
int main()
{ int choice;
  create();
  display();
  printf("Press 1 to reverse the order of singly linked list\n");
  scanf("%d",&choice);
  reverse();
}
struct node*new_node()
{ struct node*temp;
  temp=(struct node*)malloc(sizeof(struct node));
  temp->next=NULL;
  return temp;
}
void create()
{
        struct node*temp,*ptr;
        int i,n;
        printf("Enter the total number of nodes: ");
        scanf("%d",&n);
        for(i=1;i<n+1;i++)
        { temp=new_node();
          printf("Enter the data of node %d: ",i);
          scanf("%d",&temp->info);
          if(start==NULL)
          { start=temp;
          }
          else
          { ptr=start;
            while(ptr->next!=NULL)
            { ptr=ptr->next;
            }
            ptr->next=temp;
          }
        }
}
void display()
{ struct node*ptr;
    ptr=start;
    printf("Data in the list\n");
    while(ptr!=NULL)
    { printf("Data = %d\n",ptr->info);
       ptr=ptr->next;
    }
```

```
{ struct node *current,*front,*prev,*ptr;
  current=NULL;
  front=NULL;
  prev=NULL;
  current=start;
  while(current!=NULL)
  { front=current->next;
    current->next=prev;
    prev=current;
    current=front;
  }
  start=prev;
  ptr=start;
  printf("Data in the list\n");
  while(ptr!=NULL)
  { printf("Data = %d\n",ptr->info);
    ptr=ptr->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: 2023-05-26

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

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

```
#define Size 100
int queue[Size];
int front=0,rear=Size-1,temp=0;
int enqueue(int key)
{
   if(temp==Size)
   { printf("Queue is overflow.\n");
   }
     rear=(rear+1)%Size;
     temp++;
    queue[rear]=key;
    printf("Successfully inserted.\n");
    return 1;
}
int dequeue()
   { int key;
      if(temp==0)
      { printf("Queue is underflow.\n");
      }
      else
      { key=queue[front];
         front++;
         if(temp!=0)
         { printf("Deleted element = %d\n",key);
            temp--;
         }
      }
         return key;
   }
   int display()
   { int i;
      if(temp==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",temp);
   }
   int isEmpty()
     { if(temp==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: 2023-05-26

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

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

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

QueueOperationsLL.c

```
struct node
{ int data;
   struct node *next;
};
struct node *front=NULL,*rear=NULL;
void enqueue(int key)
{ struct node *newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=key;
   newnode->next=NULL;
   if(front==NULL)
   { front=rear=newnode;
   }
   else
   { rear->next=newnode;
     rear=newnode;
   }
   printf("Successfully inserted.\n");
}
void dequeue()
{ if(front==NULL)
    { printf("Queue is underflow.\n");
    }
   else
   { struct node *temp=front;
      front=front->next;
      printf("Deleted value = %d\n",temp->data);
      free(temp);
   }
}
void display()
{ if(front==NULL)
   { printf("Queue is empty.\n");
   }
   else
   { struct node *temp=front;
      printf("Elements in the queue : ");
      while(temp->next!=NULL)
      { printf("%d ",temp->data);
         temp=temp->next;
      }
      printf("%d ",temp->data);
      printf("\n");
   }
}
void isEmpty()
{ if(front==NULL)
    { printf("Queue is empty.\n");
    }
    else
    { printf("Queue is not empty.\n");
    }
}
void size()
{ int len=0;
```

```
    }
    else
    { struct node *temp=front;
      while(temp->next!=NULL)
      { len++;
         temp=temp->next;
      }
      printf("Queue size : %d\n",len+1);
    }
}
```

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: 2023-05-26

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

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

```
#define size 10
int stack[size];
int top=-1;
void push(int key)
{ if(top==size-1)
   { printf("Stack is overflow.\n");
   }
   else
    { top++;
       stack[top]=key;
       printf("Successfully pushed.\n");
    }
}
int pop()
{ if(top<0)
   { printf("Stack is underflow.\n");
   }
   else
   { printf("Popped value = %d\n",stack[top]);
     top--;
   }
}
void display()
{   int i;
        if(top>=0)
   { printf("Elements of the stack are : ");
     for(i=top;i>=0;i--)
     { printf("%d ",stack[i]);
     }
     printf("\n");
   }
   else
   { printf("Stack is empty.\n");
   }
}
void peek()
{ if(top==-1)
   { printf("Stack is underflow.\n");
   }
   else
   { printf("Peek value = %d\n",stack[top]);
   }
}
void isEmpty()
{ if(top==-1)
     { printf("Stack is empty.\n");
     }
    else
    { printf("Stack is not empty.\n");
    }
}
```

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: 2023-05-26

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 *next;
};
typedef struct node *NODE;
NODE top=NULL;
NODE push(int key)
{ NODE temp;
  temp=(NODE)malloc(sizeof(struct node));
  if(temp==NULL)
  { printf("Stack is overflow.\n");
  }
  else
  { temp->data=key;
    temp->next=top;
    top=temp;
    printf("Successfully pushed.\n");
  }
}
NODE pop()
{ NODE temp;
  if(top==NULL)
  { printf("Stack is underflow.\n");
  }
  else
  { temp=top;
     top=top->next;
     printf("Popped value = %d\n",temp->data);
     free(temp);
  }
}
void peek()
{ NODE temp=top;
  if(top==NULL)
  { printf("Stack is underflow.\n");
  }
  else
  { printf("Peek value = %d\n",temp->data);
  }
}
void isEmpty()
{ NODE temp=top;
  if(top==NULL)
  { printf("Stack is empty.\n");
  }
  else
  { printf("Stack is not empty.\n");
  }
}
void display()
{ NODE temp=top;
   if(temp==NULL)
   { printf("Stack is empty.\n");
   }
   else
```

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

Execution Results - All test cases have succeeded!

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

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

Test Case - 2
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
2
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
3
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
5
Stack is underflow.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
4
Stack is empty.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
23
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
Enter your option :
1
Enter element :
24
Successfully pushed.
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit

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

S.No: 20	Exp. Name: ***Write a Program to evaluate given Postfix expression.***	Date: 2023-06-09

Aim:

Write a Program to evaluate given Postfix expression.

Constraint: The entered expression must be a valid postfix expression, according to the test scenarios.

Source Code:

`PostfixEvaluation.c`

```
#include<stdio.h>
#include<ctype.h>
int s[20];
int top=-1;
void push(int x);
int pop();
void push(int x)
{ top++;
  s[top]=x;
}
int pop()
{ int a;
  a=s[top];
  top--;
  return a;
}
void main()
{ char exp[30];
  int i,v,n1,n2;
  printf("Enter the postfix expression ::");
  gets(exp);
  for(i=0;exp[i]!='\0';i++)
  { if(isalpha(exp[i]))
     { printf("Enter the value of %c:",exp[i]);
       scanf("%d",&v);
       push(v);
     }
     else if(isdigit(exp[i]))
     { push(exp[i]-'0');
     }
     else
     { n2=pop();
       n1=pop();
       switch(exp[i])
       { case'+' : { push(n1+n2);
                     break;
       }
         case'-' : { push(n1-n2);
                     break;
         }
         case'*' : { push(n1*n2);
                     break;
         }
         case'%' : { push(n1%n2);
                     break;
         }
         case'^' : { push(n1^n2);
                      break;
         }
         default : printf("Enter valid option\n");
       }
     }
  }
  printf("The result of expression %s = %d\n",exp,s[top]);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the postfix expression ::
ab+cd-*
Enter the value of a:
2
Enter the value of b:
3
Enter the value of c:
2
Enter the value of d:
4
The result of expression ab+cd-* = -10

Test Case - 2
User Output
Enter the postfix expression ::
a*b*c
Enter the value of a:
1
Enter the value of b:
2
Enter the value of c:
3
The result of expression a*b*c = 3

S.No: 21	Exp. Name: ***Write the code for traversing a binary tree in preorder, inorder and postorder***	Date: 2023-06-09

Aim:

Write a recursive C program for traversing a binary tree in preorder, inorder and postorder.

Source Code:

`binaryTree.c`

ID: 22K61A05C2

2022-2026-CSE-B

Sasi Institute of Technology and Engineering (Autonomous)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{ int data;
  struct node *left;
  struct node *right;
};
struct node *root=NULL;
void inorder(struct node *temp)
{ if(temp)
   { inorder(temp->left);
     printf("%d->",temp->data);
     inorder(temp->right);
   }
}
void preorder(struct node *temp)
{ if(temp)
    { printf("%d->",temp->data);
      preorder(temp->left);
      preorder(temp->right);
    }
}
void postorder(struct node *temp)
{  if(temp)
     { postorder(temp->left);
       postorder(temp->right);
       printf("%d->",temp->data);
     }
}
void create()
{ root=NULL;
  insert();
}
struct node *createnode()
{ struct node *r;
  r=(struct node*)malloc(sizeof(struct node));
  return r;
}
void insert()
{ struct node *temp,*r;
  r=createnode();
  printf("Enter the data: ");
  scanf("%d",&r->data);
  r->left=NULL;
  r->right=NULL;
  if(root==NULL)
  { root=r;
  }
  else
  { temp=root;
    while(temp!=NULL)
    { if(temp->data>r->data)
       { if(temp->left==NULL)
          { temp->left=r;
            temp=temp->left;
```

```
        }
        else
        { if(temp->right==NULL)
           { temp->right=r;
             temp=temp->right;
           }
           temp=temp->right;
        }
     }
   }
}
int main()
{ root=NULL;
  int x,op;
  do
  { printf("0.create\n1.insert\n2.preorder\n3.postorder\n4.inorder\n5.exit\n");
    printf("Enter your choice: ");
    scanf("%d",&op);
    switch(op)
    { case 0:create();
             break;
      case 1:insert();
             break;
      case 2: { printf("Display tree in Preorder ");
                preorder(root);
                printf("\n");
                break;
      }
      case 3: { printf("Display tree in Postorder ");
                postorder(root);
                printf("\n");
                break;
      }
      case 4: { printf("Display tree in Inorder ");
                inorder(root);
                printf("\n");
                break;
      }
      case 5: exit(0);
      default:printf("Enter valid input\n");
    }
  }while(op!=5);
  return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
0.create
1.insert
2.preorder

3.postorder
4.inorder
5.exit
Enter your choice:
0
Enter the data:
25
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
245
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
0
Enter the data:
345
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
36
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
589
0.create
1.insert
2.preorder

3.postorder
4.inorder
5.exit
Enter your choice:
2
Display tree in Preorder 345->36->589->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
3
Display tree in Postorder 36->589->345->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
4
Display tree in Inorder 36->345->589->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
5

Test Case - 2
User Output
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
0
Enter the data:
21
0.create
1.insert
2.preorder
3.postorder

4.inorder
5.exit
Enter your choice:
0
Enter the data:
325
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
586
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
0
Enter the data:
26
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
478
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
213
0.create
1.insert
2.preorder
3.postorder

4.inorder
5.exit
Enter your choice:
1
Enter the data:
36
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
21
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
1
Enter the data:
2245
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
2
Display tree in Preorder 26->21->478->213->36->2245->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
3
Display tree in Postorder 21->36->213->2245->478->26->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit

Enter your choice:
4
Display tree in Inorder 21->26->36->213->478->2245->
0.create
1.insert
2.preorder
3.postorder
4.inorder
5.exit
Enter your choice:
5

S.No: 22	Exp. Name: ***Program to insert into BST and traversal using In-order, Pre-order and Post-order***	Date: 2023-06-09

Aim:

Write a program to create a binary search tree of integers and perform the following operations using linked list.

1. Insert a node
2. In-order traversal
3. Pre-order traversal
4. Post-order traversal

Source Code:

`BinarySearchTree.c`

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

void main() {
        int x, op;
        BSTNODE root = NULL;
        while(1) {
                printf("1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder
Traversal 5.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1: printf("Enter an element to be inserted : ");
                                        scanf("%d", &x);
                                        root = insertNodeInBST(root,x);
                                        break;
                        case 2:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (in-order
traversal): ");
                                                inorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 3:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (pre-order
traversal): ");
                                                preorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 4:
                                        if(root == NULL) {
                                                printf("Binary Search Tree is empty.\n");
                                        }
                                        else {
                                                printf("Elements of the BST (post-order
traversal): ");
                                                postorderInBST(root);
                                                printf("\n");
                                                }
                                        break;
                        case 5:
                                        exit(0);
                }
        }
}
```

InsertAndTraversals.c

```
struct node {
        int data;
        struct node *left, *right;
};

typedef struct node *BSTNODE;

BSTNODE newNodeInBST(int item) {
        BSTNODE temp =  (BSTNODE)malloc(sizeof(struct node));
        temp->data = item;
        temp->left = temp->right = NULL;
        return temp;
}

void inorderInBST(BSTNODE root) {
        if(root!=NULL)
        { inorderInBST(root->left);
          printf("%d ",root->data);
          inorderInBST(root->right);
        }
}

void preorderInBST(BSTNODE root) {
        if(root!=NULL)
     { printf("%d ",root->data);
       preorderInBST(root->left);
       preorderInBST(root->right);
     }
}

void postorderInBST(BSTNODE root) {
        if(root!=NULL)
        { postorderInBST(root->left);
          postorderInBST(root->right);
          printf("%d ",root->data);
        }
}
BSTNODE insertNodeInBST(BSTNODE node, int ele) {
    if(node==NULL)
    { printf("Successfully inserted.\n");
      return newNodeInBST(ele);
    }
    else if(ele<node->data)
    { node->left=insertNodeInBST(node->left,ele);
    }
    else if(ele>node->data)
    { node->right=insertNodeInBST(node->right,ele);
    }
    else
    { printf("Element already exists in BST\n");
    }
    return node;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
54
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
28
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
62
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
2
Elements of the BST (in-order traversal): 28 54 62
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 54 28 62
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
4
Elements of the BST (post-order traversal): 28 62 54
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
5

Test Case - 2
User Output
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
100
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit

Enter your option :
1
Enter an element to be inserted :
20
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
200
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
10
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
30
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
150
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
1
Enter an element to be inserted :
300
Successfully inserted.
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
2
Elements of the BST (in-order traversal): 10 20 30 100 150 200 300
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 100 20 10 30 200 150 300
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :
4
Elements of the BST (post-order traversal): 10 30 20 150 300 200 100
1.Insert 2.Inorder Traversal 3.Preorder Traversal 4.Postorder Traversal 5.Exit
Enter your option :

S.No: 23	Exp. Name: ***Implementation of BST Operations - Deletion and Pre-order traversal***	Date: 2023-06-09

Aim:

Fill in the missing code in the below functions `deleteNodeInBST(BSTNODE root, int ele)` and `preorderInBST(BSTNODE root)` in the file `BSTDeleteAndPreOrder.c`.

The `deleteNodeInBST(BSTNODE root, int ele)` function deletes a node from the binary search tree.

The `preorderInBST(BSTNODE root)` performs and pre-order traversal on the tree.

Source Code:

BSTmain2.c

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

void main() {
        int x, op;
        BSTNODE root = NULL;
        while(1)
        {
                printf("1.Insert 2.Delete 3.Preorder Traversal 4.Exit\n");
                printf("Enter your option : ");
                scanf("%d", &op);
                switch(op) {
                        case 1: printf("Enter an element to be inserted : ");
                                scanf("%d", &x);
                                root = insertNodeInBST(root,x);
                                break;
                        case 2: printf("Enter an element to be deleted : ");
                                scanf("%d", &x);
                                root = deleteNodeInBST(root,x);
                                break;
                        case 3:
                                if(root == NULL) {
                                        printf("Binary Search Tree is empty.\n");
                                }
                                else {
                                        printf("Elements of the BST (pre-order
traversal): ");
                                        preorderInBST(root);
                                        printf("\n");
                                }
                                break;
                        case 4: exit(0);
                }
        }
}
```

BSTDeleteAndPreOrder.c

```
struct node {
        int data;
        struct node *left, *right;
};

typedef struct node * BSTNODE;

BSTNODE newNodeInBST(int item) {
        BSTNODE temp =  (BSTNODE)malloc(sizeof(struct node));
        temp->data = item;
        temp->left = temp->right = NULL;
        return temp;
}

void preorderInBST(BSTNODE root) {
        if(root!=NULL)
        { printf("%d ",root->data);
          preorderInBST(root->left);
          preorderInBST(root->right);
        }
}

BSTNODE insertNodeInBST(BSTNODE node, int ele) {
        if (node == NULL) {
                printf("Successfully inserted.\n");
                return newNodeInBST(ele);
        }
        if (ele < node->data)
                node->left  = insertNodeInBST(node->left,ele);
        else if (ele > node->data)
                node->right = insertNodeInBST(node->right,ele);
        else
                printf("Element already exists in BST.\n");
        return node;
}


BSTNODE minValueNode(BSTNODE node) {
        BSTNODE curr=node;
        while(curr&&curr->left!=NULL)
        { curr=curr->left;
        }
}

BSTNODE deleteNodeInBST(BSTNODE root, int ele) {
        if(root==NULL)
        { printf("Cannot find %d in the binary search tree.\n",ele);
          return root;
        }
         if(ele<root->data)
         { root->left=deleteNodeInBST(root->left,ele);
         }
         else if(ele>root->data)
         { root->right=deleteNodeInBST(root->right,ele);
         }
```

```
            { BSTNODE temp=root->right;
              printf("Deleted %d from binary search tree.\n",root->data);
              free(root);
              return temp;
            }
            else if(root->right==NULL)
            { BSTNODE temp=root->left;
              printf("Deleted %d from binary search tree.\n",root->data);
              free(root);
              return temp;
            }
            else
            { BSTNODE temp=minValueNode(root);
              root->data=temp->data;
              root->right=deleteNodeInBST(root->right,ele);
            }
          }
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
1
Enter an element to be inserted :
56
Successfully inserted.
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
2
Enter an element to be deleted :
35
Cannot find 35 in the binary search tree.
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 56
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
4

Test Case - 2
User Output
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :

1
Enter an element to be inserted :
25
Successfully inserted.
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
1
Enter an element to be inserted :
65
Successfully inserted.
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
2
Enter an element to be deleted :
65
Deleted 65 from binary search tree.
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
3
Elements of the BST (pre-order traversal): 25
1.Insert 2.Delete 3.Preorder Traversal 4.Exit
Enter your option :
4