Banarsidas Chandiwala Institute of Information Technology

Affiliated to

GuruGovind Singh Indraprastha University, New Delhi

MASTEROFCOMPUTER APPLICATION

Subject–Data and File Structures(MCA-102)

Batch (2023-25)

Submitted by:
Name: Akansha
Roll no.: 02011104423

Submitted to:
Name: Dr. Anu Taneja
Designation: Asst. Professor

INDEX

9.	**Write a program to following operations on singly linked list** 1. Addition of a new node when list is empty. 2. Addition of a node at the end of the list. 3. Addition of a node at the beginning of linked list. 4. Addition of a node at the specific position. 5. Traversing a linked list. 6. Counting number of nodes in linked list. 7. Deleting a node from the linked list by specifying the value of the deleting node. 8. Reverse A linked List. 9. Sorting. 10. Search 11. Insert a node in a sorted linked list such that linked list remains sorted		
10.	**Write a program to following operations on Doubly linked list** 1. Addition of a new node when list is empty. 2. Addition of a node at the end of the list. 3. Addition of a node at the beginning of linked list. 4. Addition of a node at the specific position. 5. Traversing a linked list. 6. Counting number of nodes in linked list. 7. Deleting a node from the linked list at beginning, at end, at specific position. 8. Reverse A linked List. 9. Sorting 10. Search		
11.	**Write a program to following operations on Circular linked list** 1. Addition of a new node when list is empty. 2. Addition of a node at the end of the list. 3. Addition of a node at the beginning of linked list. 4. Addition of a node at the specific position. 5. Traversing a linked list. 6. Counting number of nodes in linked list. 7. Deleting a node from the linked list at beginning, at end, at specific position. 8. Reverse A linked List. 9. Sorting. 10. Search		

12.	**Write a program to following operations on Doubly Circular linked list** 1. Addition of a new node when list is empty. 2. Addition of a node at the end of the list. 3. Addition of a node at the beginning of linked list. 4. Addition of a node at the specific position. 5. Traversing a linked list. 6. Counting number of nodes in linked list. 7. Deleting a node from the linked list at beginning, at end, at specific position. 8. Reverse A linked List. 9. Sorting 10. Search		
13.	WAP to perform addition and multiplication of two polynomials by creating linked list for every polynomial.		
14.	**Write a program to perform following operations on Stack using Arrays.** 1. Push. 2. Pop. 3. Peek. 4. Traverse. 5. Search.		
15.	**Write a program to implement stack using linked list and perform following operations on it.** 1. Push. 2. Pop. 3. Peek. 4. Traverse. 5. Search.		
16.	Write a program to convert an expression from Infix to Postfix using Stack.		
17.	Write a program to evaluate a Postfix expression using Stack		
18.	**Write a program to implement Linear queue using array and perform the following operation on it.** 1. Enqueue. 2. Dequeue. 3. Traverse. 4. Peek 5. Search 6. Max element in Queue		

19.	**Write a program to implement Linear queue using linked list and perform the following operation on it.** 1. Enqueue. 2. Dequeue. 3. Traverse. 4. Peek 5. Search 6. Max element in Queue		
20.	**Write a program to implement Circular queue using array and perform the following operation on it.** 1. Enqueue. 2. Dequeue. 3. Traverse. 4. Peek 7. Search 5. Max element in Queue		
21.	**Write a program to implement Circular queue using Linked List and perform the following operation on it.** 1. Enqueue. 2. Dequeue. 3. Traverse. 4. Peek		
22.	Write a program to implement concept of Muti-Stack and Multi-Queue		
23.	Write a program to implement Merge Sort.		
24.	Write a program to implement Quick Sort.		
25.	Write a program to implement Shell Sort.		

Ques 1. Write A C Program to Perform These Operation on An Array.

```
#include<stdio.h>
void traversearr(int arr[],int size)
{
   int i;
   printf("ARRAY is :");
   for(i=0;i<size;i++)
   {
      printf(" %d",arr[i]);
   }
   printf("\n");
}
int insertion(int arr[],int el,int pos,int size)
{
   int i;
   for(i=size-1;i>=pos;i--)
   {
      arr[i+1]=arr[i];
   }
   arr[pos]=el;

   return size+1;
}
int deletion(int arr[],int pos,int size)
{
   int i;
   for (i=pos-1;i<size;i++)
   {
      arr[i]=arr[i+1];
   }
   return size-1;
}
int linearsearch(int arr[],int size,int search)
{
   int i,count;
   for(i=0;i<size;i++)
   {
      if(arr[i]==search)
      {
         printf("\nElement Found at index %d ",i);
         count++;
      }
   }
      if(count==0)
```

```
            printf("\nElement is Not present ");
        else
            printf("\nElement is present %d times",count);
    return count;
}
int binarysearch(int arr[],int size,int search)
{
    int i,low,high,mid;
    low=0;
    high=size-1;
    while(low<=high)
    {
        mid=low+high/2;
        if(search==arr[mid]){
            return mid;
        }
        else if(search>arr[mid]){
            low=mid+1;
        }
        else
            high=mid-1;
    }
}
int binarySearchwithrec(int arr[], int search, int low, int high) {
  if (high >= low) {
    int mid = low + (high - low) / 2;
    if (arr[mid] == search)
      return mid;
    if (arr[mid] > search)
      return binarySearchwithrec(arr, search, low, mid - 1);
    return binarySearchwithrec(arr, search, mid + 1, high);
  }
  return -1;
}
void bubblesort(int arr[],int size)
{
    int i,j,temp;
    for(i=0;i<size-1;i++)
    {
        for(j=0;j<size-i-1;j++)
        if(arr[j]>arr[j+1])
        {
            temp=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=temp;
```

```
        }
    }}
void insertionsort(int arr[],int size)
{
    int i,j,temp;
    for(i=1;i<=size-1;i++)
    {
        temp=arr[i];
        j=i-1;
        while(j>=0&&arr[j]>temp)
        {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=temp;
    }
}
void selectionsort(int arr[],int size)
{
    int i,j,temp,min;
    for(i=0;i<size-1;i++)
    {
        min=i;
        for(j=i+1;j<size;j++)
        {
            if(arr[min]>arr[j])
            {
                min=j;
            }
        }
        if(min!=i)
        {
            temp=arr[i];
            arr[i]=arr[min];
            arr[min]=temp;
        }
    }}
int main()
{
    int size,i,ch,el,pos ,search,result;
    char op;
    int arr[10];
    printf("Enter the size of an Array :");
    scanf("%d",&size);
    printf("Enter the elements of an Array :");
```

```
for(i=0;i<size;i++)
{
   scanf("%d",&arr[i]);
}
do{
   printf("***ENTER YOUR CHOICE*** \n");
   printf("1 : Traversing an Array \n");
   printf("2 : Insertion in an Array\n ");
   printf("3 : Deletion in an Array \n");
   printf("4 : Linear Search \n");
   printf("5 : Binary Search without Recursion \n");
   printf("6 : Binary Search with Recursion \n");
   printf("7 : Sorting through Bubble Sort technique \n");
   printf("8 : Sorting through Insertion Sort technique \n");
   printf("9 : Second largest element through Sorting \n");
   printf("10 : Sorting through Selection Sort Technique\n");
   printf("Enter your Choice :");
   scanf("%d",&ch);
   switch (ch)
   {
      case 1:
      traversearr(arr,size);
      break;
      case 2:
      printf("Enter the element ehich you want to insert in an Array :");
      scanf("%d",&el);
      printf("Enter the position from which this element is to inserted :");
      scanf("%d",&pos);
      size=insertion(arr,el,pos,size);
      break;
      case 3:
      printf("Enter the position at which element should be deleted :");
      scanf("%d",&pos);
      size=deletion(arr,pos,size);
      break;
      case 4:
      printf("Enter the element you want to SEARCH :");
      scanf("%d",&search);
      linearsearch(arr,size,search);
      break;
      case 5:
      printf("Enter the element you want to SEARCH :");
      scanf("%d",&search);
      result=binarysearch(arr,size,search);
      if(result==-1)
```

```
        {
           printf("Elemet Not Found .");
        }
        else{
           printf("Element is found at index %d ",result);
        }
        Break;
        case 6:
        printf("Enter the element you want to SEARCH :");
        scanf("%d",&search);
        result = binarySearchwithrec(arr, search, 0, size - 1);
        if (result == -1)
         printf("Element Not found");
        else
        printf("Element is found at index %d", result);
        break;
        case 7:
        bubblesort(arr,size);
        printf("Array is Sorted, Please Enter Y/y and then entered choice 1");
        break;

        case 8:
        insertionsort ( arr , size);
        printf("Array is Sorted, Please Enter Y/y and then entered choice 1");
        break;

        case 9:
        bubblesort(arr,size);
        printf("Second Largest Element Through sorting Technique: %d",arr[size-2]);
        break;

        case 10:
        selectionsort(arr,size);
        break;

        default:
        printf("Invalid Choice,Please Enter Valid Choice ");
      }
      printf("\n Do you want to run the program again?(Y/N):");
      scanf(" %c",&op)
   }while (op=='Y'||op=='y');
   return 0;
}
```

OUTPUT:-

```
Enter the size of an Array :6
Enter the elements of an Array :45
78
32
89
67
22
***ENTER YOUR CHOICE***
1 : Traversing an Array
2 : Insertion in an Array
 3 : Deletion in an Array
4 : Linear Search
5 : Binary Search without Recursion
6 : Binary Search with Recursion
7 : Sorting through Bubble Sort technique
8 : Sorting through Insertion Sort technique
9 : Second largest element through Sorting
10 : Sorting through Selection Sort Technique
Enter your Choice :2
Enter the element ehich you want to insert in an Array :44
Enter the position from which this element is to inserted :2
```

```
ARRAY is : 45 78 44 32 89 67 22
```

```
 Do you want to run the program again?(Y/N):y
***ENTER YOUR CHOICE***
1 : Traversing an Array
2 : Insertion in an Array
 3 : Deletion in an Array
4 : Linear Search
5 : Binary Search without Recursion
6 : Binary Search with Recursion
7 : Sorting through Bubble Sort technique
8 : Sorting through Insertion Sort technique
9 : Second largest element through Sorting
10 : Sorting through Selection Sort Technique
Enter your Choice :4
Enter the element you want to SEARCH :89
```

```
 Do you want to run the program again?(Y/N):y
***ENTER YOUR CHOICE***
1 : Traversing an Array
2 : Insertion in an Array
 3 : Deletion in an Array
4 : Linear Search
5 : Binary Search without Recursion
6 : Binary Search with Recursion
7 : Sorting through Bubble Sort technique
8 : Sorting through Insertion Sort technique
9 : Second largest element through Sorting
10 : Sorting through Selection Sort Technique
Enter your Choice :9
Second Largest Element Through sorting Technique: 75ARRAY is : 11 24 43 67 75 89
```

Ques 2. Write A C Program to Check Whether a Matrix Is Sparse or Not.

```
#include<stdio.h>
int main()
{
   int arr[10][10],b[10][10],row,col,count,i,j,k,total,size;
   count=0,size=0,k=0;
   printf("Enter the size of rows :");
   scanf("%d",&row);
   printf("Enter the size of columns :");
   scanf("%d",&col);
   printf("Enter the Elements of an Array:");
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         scanf("%d",&arr[i][j]);
      }
      printf("\n");
   }
   printf("Matrix is :\n");
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         printf(" %d",arr[i][j]);
      }
      printf("\n");
   }
   total=row*col;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         if(arr[i][j]==0)
         {
            count++;
         }
         else{
            size++;
         }
      }
   }
   if(total/2<count)
   {
```

```
    printf("Matrix is Sparse");
  }
  else
    printf("Not a Sparse Matrix");
return 0;
}
```

OUTPUT:-

```
Enter the size of rows :3
Enter the size of columns :3
Enter the Elements of an Array:1

0
8

0
8
0

0
0
0

Matrix is :
 1 0 8
 0 8 0
 0 0 0
Matrix is Sparse
```

Ques 3. Write A C Program to Covert Sparse Matrix Into 3-Triplet Representation.

```
#include<stdio.h>
void rowtriplet(int arr[10][10],int row,int col,int size)
{
   int b[10][10],i,j;
   int k=0;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         if(arr[i][j]!=0)
         {
            b[0][k]=i;
            b[1][k]=j;
            b[2][k]=arr[i][j];
            k++;
         }}}
   printf("\n***3-ROW TRIPlET REPRESENTATION***\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<size;j++)
      {
         printf(" %d",b[i][j]);
      }
      printf("\n");
   }}
int main()
{
   int arr[10][10],b[10][10],row,col,count,i,j,k,total,size;
   count=0,size=0,k=0;
   printf("Enter the size of rows :");
   scanf("%d",&row);
   printf("Enter the size of columns :");
   scanf("%d",&col);
   printf("Enter the Elements of an Array:");
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         scanf("%d",&arr[i][j]);
      }
      printf("\n");
   }
   printf("Matrix is :\n");
```

```
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         printf(" %d",arr[i][j]);
      }
      printf("\n");}
   total=row*col;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         if(arr[i][j]==0)
         {
            count++;
         }
         else{
            size++;
         }}
   if(total/2<count)
   {
      printf("Matrix is Sparse");
      rowtriplet(arr,row,col,size);
   }
   else
      printf("Not a Sparse Matrix");
return 0;
}
```

OUTPUT:-

```
Enter the size of rows :3
Enter the size of columns :3
Enter the Elements of an Array:1

0
8

0
8
0

0
0
0

Matrix is :
 1 0 8
 0 8 0
 0 0 0
Matrix is Sparse
***3-ROW TRIPlET REPRESENTATION***
 0 0 1
 0 2 1
 1 8 8
```

Ques 4. Write A C Program to Perform Following Operation on Matrix

```
#include<stdio.h>

void traverse(int a[10][10],int row,int col)
{
   int i,j;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         printf(" %d",a[i][j]);
      }
      printf("\n");
   }
}
void addition(int a[10][10],int b[10][10],int row,int col)
{
   int i,j;
   int add[10][10];
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         add[i][j]=a[i][j]+b[i][j];
      }
   }
   printf("Addition of two Matrices is : \n");
   traverse(add,row,col);
}
void subtraction(int a[10][10],int b[10][10],int row,int col)
{
   int i,j;
   int sub[10][10];
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         sub[i][j]=a[i][j]-b[i][j];
      }
   }
   printf("Subtraction of two Matrices is : \n");
   traverse(sub,row,col);
}
void multiplication(int a[10][10],int b[10][10],int row,int col)
```

```
{
   int mul[10][10];
   int i,j,k;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         mul[i][j]=0;
         for(k=0;k<col;k++)
         {
            mul[i][j]+=a[i][k]*b[k][j];
         }
      }
   }
   printf("Result of Multiplication Mtrix is :\n");
   traverse(mul,row,col);
}
void transpose(int a[10][10],int row,int col)
{
   int trans[10][10];
   int i,j;
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         trans[j][i]=a[i][j];
      }
   }
   printf("After Transpose of the Matrix is : \n");
   traverse(trans,col,row);
}
int identity(int a[10][10],int row,int col)
{
   int i,j;
   int flag=1;
   if(row!=col)
   {
      printf("Matrix Should be Squared!!");
   }
   else{
   for(i=0;i<row;i++)
   {
      for(j=0;j<col;j++)
      {
         if(i==j && a[i][j]!=1)
```

```
            {
                flag=0;
                break;
            }
            if(i!=j && a[i][j]!=0)
            {
                flag=0;
                break;
            }
        }
    }
    }
    return flag;
}
void diagonalsum(int a[10][10],int b[10][10],int row,int col)
{
    int res[10][10];
    int i,j;
    if(row==col)
    {
        for(i=0;i<row;i++)
        {
            for(j=0;j<col;j++)
            {
                if(i==j)
                {
                    res[i][j]=a[i][j]+b[i][j];
                }
            }
        }
    }
    printf("Addition of Diagonal Elements is :\n");
    traverse(res,row,col);
}
int main()
{
    int a[10][10],b[10][10];
    int row,col,i,j,choice;
    char op;
    printf("Enter the size of rows :");
    scanf("%d",&row);
    printf("Enter the size of columns :");
    scanf("%d",&col);
    printf("Enter the elements of Array A :");
    for(i=0;i<row;i++)
```

```
    {
        for(j=0;j<col;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter the elements of Array B :");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            scanf("%d",&b[i][j]);
        }
}
    do
    {
        printf("***Enter Your Choice***\n");
        printf("0 . Traverse the Matrices .\n");
        printf("1 . Addition of Two Matrices.\n");
        printf("2 . Subtraction of Two Matrices.\n");
        printf("3 . Multiplication of Two Matrices.\n");
        printf("4 . Transpose of a Matrices.\n");
        printf("5 . Check Matrix is Identity or not.\n");
        printf("6 . Print Sum of Diagonal Elements.\n");
        printf("Enter your choice from the Above points : \n");
        scanf("%d",&choice);
        switch (choice)
        {
            case 0:
            printf("Matrix A is :\n");
            traverse(a,row,col);
            printf("Matrix B is :\n");
            traverse(b,row,col);
            break;

            case 1:
            addition(a,b,row,col);
            break;

            case 2:
            subtraction(a,b,row,col);
            break;

            case 3:
            multiplication(a,b,row,col);
```

```
        break;
        case 4:char ch;
        printf("Which Matrix you want to transpose a or b ?: ");
        scanf(" %c",&ch);
        if(ch=='a')
           transpose(a,row,col);
        else
           transpose(b,row,col);
        break;

        case 5:char ch1;
        int res;
        printf("Which Matrix you want to Check for an IDENTITY, a or b ?: ");
        scanf(" %c",&ch1);
        if(ch1=='a')
           res=identity(a,row,col);
        else
           res=identity(b,row,col);
        if(res==1)
           printf("\nMatrix is Identity ");
        else
           printf("\nMatrix is not Identity ");
        break;

        case 6:
        diagonalsum(a,b,row,col);
        break;

        default:
        printf("Wrong Choice!,Please Enter the coorect Choice..");
        break;
     }
     printf("\nDo You want to Run the program again? Y/N :");
     scanf(" %c",&op);
  } while (op=='Y'||op=='y');

  return 0;
}
```

OUTPUT:-

```
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
0
Matrix A is :
 1 2 3
 4 5 4
 3 2 1
Matrix B is :
 9 8 7
 6 5 4
 3 2 1
```

```
Do You want to Run the program again? Y/N :y
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
1
Addition of two Matrices is :
 10 10 10
 10 10 8
 6 4 2
```

```
Do You want to Run the program again? Y/N :y
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
2
Subtraction of two Matrices is :
 -8 -6 -4
 -2 0 0
 0 0 0
```

```
Do You want to Run the program again? Y/N :y
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
3
Result of Multiplication Mtrix is :
 30 24 18
 78 65 52
 42 36 30
```

```
Do You want to Run the program again? Y/N :y
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
4
Which Matrix you want to transpose a or b ?: a
After Transpose of the Matrix is :
 1 4 3
 2 5 2
 3 4 1
```

```
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
5
Which Matrix you want to Check for an IDENTITY, a or b ?: b

Matrix is not Identity
Do You want to Run the program again? Y/N :y
***Enter Your Choice***
0 . Traverse the Matrices .
1 . Addition of Two Matrices.
2 . Subtraction of Two Matrices.
3 . Multiplication of Two Matrices.
4 . Transpose of a Matrices.
5 . Check Matrix is Identity or not.
6 . Print Sum of Diagonal Elements.
Enter your choice from the Above points :
6
Addition of Diagonal Elements is :
 10 0 0
 0 10 0
 0 0 2
```

Ques 5. Write A C Program to Perform Count Sort on An Unsorted Array.

```c
#include<stdio.h>

int main()
{
    int a[10],b[10],size,i,max;
    printf("Enter the size of array :");
    scanf("%d",&size);
    printf("Enter the single digits Elements of an Array :");
    for(i=0;i<size;i++)
    {
        scanf("%d",&a[i]);
    }
    // Print the Array
    printf("The Elements of an Array :");
    for(i=0;i<size;i++)
    {
        printf(" %d",a[i]);
    }
    // set maximum element at index 0
    max=a[0];
    for(i=0;i<size;i++)
    {
        if(a[i]>max)
        {
            max=a[i];
        }
    }
    // print the maximum element
    printf("\n The maximum element is : %d",max);
    int count[max];
    For (i=0; i<=max; i++)
    {
        count[i]=0;
    }
    // update count array --> means how many times an element occur in an
array
    for(i=0;i<size;i++)
    {
        count[a[i]]++;

    }
    // Find cumulative Count
    for(i=1;i<=max;i++)
```

```
    {
        count[i]=count[i]+count[i-1];
    }
    // Trace the input array from last Index and Store the sorted values in
new array B[]
    for(i=size-1;i>=0;i--)
    {
        b[count[a[i]]-1]=a[i];
        count[a[i]]--;
    }
    for(i=0;i<size;i++)
    {
        a[i]=b[i];
    }
    printf("\nSorted Array is :\n");
    for(i=0;i<size;i++)
    {
        printf("%d ",a[i]);
    }

    return 0;
}
```

OUTPUT:-

```
Enter the size of array :7
Enter the single digits Elements of an Array :3
6
8
5
3
5
6
The Elements of an Array : 3 6 8 5 3 5 6
The maximum element is : 8
Sorted Array is :
3 3 5 5 6 6 8
```

Ques 6. Write A C Program to Perform Radix Sort on An Unsorted Array.

```
#include<stdio.h>
#include<stdlib.h>
void printarr(int b[],int size)
{
    int i;
    printf("\nAfter Pass The Sorted ARRAY is :" );
    for(i=0;i<size;i++)
    {
        printf(" %d",b[i]);
    }
    printf("\n");

    return;}
void counting_sort(int arr[],int size,int pos)
{
    int i;
    int count[10];
    int *b;
    b=(int *)malloc(size*sizeof(int));
    for(i=0;i<10;i++)
    {
        count[i]=0;
    }
    for(i=0;i<size;i++)
    {
        count[(arr[i]/pos)%10]++;
    }
    for(i=1;i<10;i++)
    {
        count[i]=count[i]+count[i-1];
    }
    for(i=size-1;i>=0;i--)
    {
        b[count[(arr[i]/pos)%10]-1]=arr[i];
        count[(arr[i]/pos)%10]--;
    }
    for(i=0;i<size;i++)
    {
        arr[i]=b[i];
    }
    free(b);
return;}
int main()
{
```

```
    int size,i,max,pos;
    int *arr;
    printf("Enter the size of array :");
    scanf("%d",&size);
    arr=(int*)malloc(size*sizeof(int));
    printf("Enter the elements of an Array :");
    for(i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("The elements of an Array :\n");
    for(i=0;i<size;i++)
    {
        printf(" %d",arr[i]);
    }
    max=arr[0];
    for(i=0;i<size;i++)
    {
        if(max<arr[i])
        {
            max=arr[i];
        }
    }
    printf("\nMaximum element is %d:",max);
    for(pos=1;max/pos>0;pos=pos*10)
    {
        counting_sort(arr,size,pos);
    }
    printarr(arr,size);
return 0;}
```

OUTPUT:

```
Enter the size of array :6
Enter the elements of an Array :45
333
78
11
233
67
The elements of an Array :
 45 333 78 11 233 67
Maximum element is 333:
After Pass The Sorted ARRAY is : 11 45 67 78 233 333
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 7. Write A C To Dynamically Allocate an Array and Calculate Sum of Its Elements.

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

int main()
{
    int size,i,sum;
    int *p;
    sum=0;
    printf("Enter the size of Array :");
    scanf("%d",&size);
    //p=(int*)malloc((size*sizeof(int)));
    p=(int*)calloc(size,sizeof(int));
    printf("Enter the elements :");
    for(i=0;i<size;i++)
    {
        scanf("%d",p+i);
    }
    printf("The Array is :");
    for(i=0;i<size;i++)
    {
        printf(" %d",*(p+i));
    }
    for(i=0;i<size;i++)
    {
        sum+=*(p+i);
    }
    printf("\nThe sum of elements in an Array : %d",sum);
    return 0;
}
```

OUTPUT:-

```
Enter the size of Array :5
Enter the elements :2
3
5
6
4
The Array is : 2 3 5 6 4
The sum of elements in an Array : 20
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 8. Write A C Program to Remove Duplicate Elements from An Array.

```c
#include <stdio.h>

int removeDuplicates(int arr[], int n) {
    if (n <= 1)
        return n;

    int i, j, k;
    int flag;

    for (i = 0; i < n; i++) {
        flag = 0;
        for (j = i + 1; j < n;) {
            if (arr[i] == arr[j]) {
                for (k = j; k < n; k++) {
                    arr[k] = arr[k + 1];
                }
                n--;
                flag = 1;
            } else {
                j++;
            }
        }
        if (flag) {
            i--;
        }
    }
    return n;
}

int main() {
    int arr[100];
    int size;

    printf("Enter the size of the array: ");
    scanf("%d", &size);

    printf("Enter the elements of the array: ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    printf("The original array: ");
    for (int i = 0; i < size; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");

    size = removeDuplicates(arr, size);

    printf("Array after removing duplicates: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:-

```
Enter the size of the array: 6
Enter the elements of the array: 4
4
5
3
6
3
The original array: 4 4 5 3 6 3
Array after removing duplicates: 4 5 3 6
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 9. Write a program to following operations on singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
struct node *start;
struct node *newnode;
struct node *temp;
void create()
{
   int choice;
   start=NULL;
   do{
      newnode=(struct node*)malloc(sizeof(struct node));
      printf("Enter the data :\n");
      scanf(" %d",&newnode->data);
      newnode->next=NULL;
      if(start==NULL)
      {
         start=newnode;
         temp=newnode;
      }
      else{
         temp->next=newnode;
         temp=newnode;
      }
      printf("Do You Want to Enter More Data? If Yes, enter 1 otherwise 0..");
      scanf("%d",&choice);
   }while(choice);
}

void display()
{
   temp=start;
   printf("Entered data is :\n");
   while(temp!=NULL)
   {
      printf(" %d",temp->data);
      temp=temp->next;
   }
   printf("\n");}
void insertatbeg()
{
```

```
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("\nEnter the Data which You want to insert at begining :\n");
   scanf(" %d",&newnode->data);
   newnode->next=NULL;
   if(start==NULL)
   {
      start=newnode;
      temp=newnode;
   }
   else{
      newnode->next=start;
      start=newnode;
   }}
void insertatend()
{
   int count;
   count=length();
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("\nEnter the which You want to insert at End :\n");
   scanf(" %d",&newnode->data);
   newnode->next=NULL;
   temp=start;
   while(temp->next!=NULL)
   {
      temp=temp->next;
   }
   temp->next=newnode;
   temp=newnode;
}
void insertatpos()
{
   int pos,i,count;
   i=1;
   count=length();
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("\nEnter the position at which you want to insert data :\n");
   scanf(" %d",&pos);
   if(pos>count){
      printf("Invalid Position!!");
      printf("The Linked list has only %d nodes,So Please Enter from this range",count);
   }
   else{
      printf("\nEnter the data :\n");
      scanf(" %d",&newnode->data);
      temp=start;
```

```
        while(i<pos-1){
            temp=temp->next;
            i++;
        }
        newnode->next=temp->next;
        temp->next=newnode;
        printf("\nAfter Insert an element at Specific poisition, The List is :\n");
        display();
    }
}
void delete_at_beg()
{
    temp=start;
    start=temp->next;
    free(temp);
    printf("\nAfter delete an element from begining, The List is :\n");
    display();
}
void delete_at_end()
{
    struct node *prev;
    temp=start;
    while(temp->next!=NULL)
    {
        prev=temp;
        temp=temp->next;
    }
    prev->next=NULL;
    free(temp);
    printf("\nAfter delete an element from ending, The List is :\n");
    display();
}
void delete_at_pos()
{
    int pos,i,count;
    struct node *prev;
    temp=start;
    i=1;
    count=length();
    printf("\nEnter the position at which you want to delete an element :\n");
    scanf(" %d",&pos);
    if(pos>count){
        printf("Invalid Position!!");
        printf("The Linked list has only %d nodes,So Please Enter from this range",count);
    }
```

```
    else{
        while(i<=pos-1)
        {
            prev=temp;
            temp=temp->next;
            i++;
        }
        prev->next=temp->next;
        free(temp);
        printf("\nAfter delete an element from specific Position, The List is :\n");
        display();
    }}
void search(int ser)
{
    int i,flag;
    temp=start;
    i=1;
    flag=0;
    while(temp!=NULL)
    {
        if(temp->data==ser)
        {
            printf("\nElement is Found at index %d=",i);
            flag=1;
        }
            i++;
            temp=temp->next;
    }
    if(flag==0)
    {
        printf("\nElement is not Found");
    }}
void sorting()
{
    int val;
    struct node *temp1;
    temp=start;
    while(temp->next!=NULL)
    {
        temp1=temp->next;
        while(temp1!=NULL)
        {
            if(temp->data > temp1->data)
            {
                val=temp->data;
```

```
            temp->data=temp1->data;
            temp1->data=val;
          }
          temp1=temp1->next;
       }
       temp=temp->next;
    }
    printf("After Sorting The Linked list is :\n");
    display();
}
int length()
{
    int count;
    count=0;
    temp=start;
    while(temp!=NULL)
    {
       count++;
       temp=temp->next;
    }
return count;
}
void reverse()
{
    struct node *current;
    struct node *Next;
    struct node *prev;
    current = start;
    Next=NULL;
    prev=NULL;
    while(current!=NULL)
    {
       Next=current->next;
       current->next=prev;
       prev=current;
       current=Next;
    }
    start=prev;
    printf("\nAfter Reverse .");
    display();
}
int main()
{
    int choice;
    char ch;
```

```
do{
   printf("\n***ENTER YOUR CHOICE***\n");
   printf("1. Create a Linked List\n");
   printf("2. Display a Linked List\n");
   printf("3. Insert At Begining\n");
   printf("4. Insert At End\n");
   printf("5. Insert At Specific Position\n");
   printf("6. Remove from Begining\n");
   printf("7. Remove from Ending\n");
   printf("8. Remove At Specific Position\n");
   printf("9. Search Element from Linked List\n");
   printf("10. Sort the Linked List\n");
   printf("11. Count the Length of the Linked List\n");
   printf("12. Reverse the Linked List\n");
   printf("Enter your choice from the Above points : \n");
   scanf("%d",&choice);
   switch (choice)
   {
      case 1:
      create();
      break;
      case 2:
      display();
      break;
      case 3:
      insertatbeg();
      break;
      case 4:
      insertatend();
      break;
      case 5:
      insertatpos();
      break;
      case 6:
      delete_at_beg();
      break;
      case 7:
      delete_at_end();
      break;
      case 8:
      delete_at_pos();
      break;
      case 9:
      int ser;
      printf("\nEnter the Element which you want to search :");
```

```
            scanf(" %d",&ser);
            search(ser);
            break;
            case 10:
            sorting();
            break;
            case 11:int count;
            count=length();
            printf("\nThe total number of elements in a Linked List is %d: ",count);
            break;
            case 12:
            reverse();
            break;
            default: printf("Invalid Choice!! Please Enter Valid Choice...");
            break;
        }
        printf("\nDo You Want to Perform more operations? If yes Enter y/Y : ");
        scanf(" %c",&ch);

    }while(ch=='y'||ch=='Y');
    return 0;
}
```

OUTPUT:-

```
***ENTER YOUR CHOICE***
1. Create a Linked List
2. Display a Linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Remove from Begining
7. Remove from Ending
8. Remove At Specific Position
9. Search Element from Linked List
10. Sort the Linked List
11. Count the Length of the Linked List
12. Reverse the Linked List
Enter your choice from the Above points :
2
Entered data is :
 23 67 345 1 5345

Do You Want to Perform more operations? If yes Enter y/Y : y
```

```
***ENTER YOUR CHOICE***
1. Create a Linked List
2. Display a Linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Remove from Begining
7. Remove from Ending
8. Remove At Specific Position
9. Search Element from Linked List
10. Sort the Linked List
11. Count the Length of the Linked List
12. Reverse the Linked List
Enter your choice from the Above points :
2
Entered data is :
 33 23 67 345 1 5345

Do You Want to Perform more operations? If yes Enter y/Y :
```

Ques 10. Write a program to following operations on Doubly linked list

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

struct node{
   int data;
   struct node *prev;
   struct node *next;
};
struct node *newnode;
struct node *temp;
struct node *start;

void create()
{
   start=NULL;
   int op;
   do{
   newnode=(struct node *)malloc(sizeof(struct node));
   printf("Enter the data in the Doubly Linked list :\n");
   scanf(" %d",&newnode->data);
   newnode->next=NULL;
   newnode->prev=NULL;
   if(start==NULL)
   {
      start=newnode;
      temp=newnode;
   }
   else{
      temp->next=newnode;
      newnode->prev=temp;
      newnode->next=NULL;
      temp=newnode;
   }
   printf("\nDo you want to enter more data in Linked List,if yes enter 1 otherwise 0...");
   scanf("%d",&op);
   }while(op);
}

void display()
{
   temp=start;
   printf("\nYour Linked List is :");
   while(temp!=NULL)
```

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

void insertat_beg()
{
    newnode=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter the data at First position : ");
    scanf("%d",&newnode->data);
    newnode->next=NULL;
    newnode->prev=NULL;
    if(start==NULL)
    {
        start=newnode;
        temp=newnode;
    }
    else{
        newnode->next=start;
        start=newnode;
    }
}

void insertat_end()
{
    newnode=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter the data at the End : ");
    scanf("%d",&newnode->data);
    newnode->next=NULL;
    newnode->prev=NULL;
    temp=start;
    while(temp->next!=NULL)
    {
        temp=temp->next;
    }
    temp->next=newnode;
    newnode->prev=temp;
    temp=newnode;
}

void insertat_pos()
{
    int pos,i,count;
    i=1;
```

```
    printf("\nEnter the position at which you want to enter data :");
    scanf("%d",&pos);
    count=length();
    if (pos > count)
    {
        printf("Invalid Position !!\n");
    }
    else{
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("\nEnter the data at the specific Position : ");
        scanf("%d",&newnode->data);
        newnode->next=NULL;
        newnode->prev=NULL;
        temp=start;
        while(i<pos-1)
        {
            temp=temp->next;
           i++;
        }
        newnode->next=temp->next;
        temp->next=newnode;
        temp->next->prev=newnode;
        newnode->prev=temp;
    }
}

void deleteat_beg()
{
    temp=start;
    start=temp->next;
    start->prev=NULL;
    free(temp);
    display();
}

void deleteat_end() {
    struct node *tail;
    temp = start;
    while (temp->next != NULL) {
        tail = temp;
        temp = temp->next;
    }
    if (temp == start) { // Special case if there's only one node in the list
        start = NULL;
    } else {
```

```
        tail->next = NULL;
    }
    free(temp);
    display();
}

void deleteat_pos()
{
    int pos,i,count;
    i=1;
    printf("\nEnter the Position at which you want to delete an element :");
    scanf("%d",&pos);
    count=length();
    if(pos > count)
    {
        printf("\nInvalid Position !!\n");
    }
    else{
        temp=start;
        while(i<pos){
            temp=temp->next;
            i++;
        }
        temp->prev->next=temp->next;
        temp->next->prev=temp->prev;
        free(temp);
        printf("\nAfter delete from speific position ");
        display();
    }
}

void search()
{
    int ser,i,flag;
    i=1;flag=0;
    printf("\nEnter the element which you want to search :");
    scanf("%d",&ser);
    temp=start;
    while(temp!=NULL)
    {
        if(temp->data==ser)
        {
            printf("\nElement Found at Index :%d",i);
            flag=1;
        }
```

```
        i++;
        temp=temp->next;
    }
    if(flag==1)
    {
        printf("\nElement Found");
    }
    else{
        printf("\nElement Not Found");
    }
}

void sorting()
{
    struct node *temp1;
    int val;
    temp=start;
    while(temp!=NULL)
    {
        temp1=temp->next;
        while(temp1!=NULL)
        {
            if(temp->data > temp1->data)
            {
                val=temp1->data;
                temp1->data=temp->data;
                temp->data=val;
            }
            temp1=temp1->next;
        }
        temp=temp->next;
    }
    printf("After Sorting :");
    display();
}

int length()
{
    int count;
    count=0;
    temp=start;
    while(temp!=NULL)
    {
        temp=temp->next;
        count++;
```

```
    }
    return count;
}

void reverse()
{
    struct node *current;
    struct node *Prev;
    Prev=NULL;
    current=start;
    while(current!=NULL)
    {
        Prev=current->prev;
        current->prev=current->next;
        current->next=Prev;
        current=current->prev;
    }
    if(Prev!=NULL){
        start=Prev->prev;
    }
    printf("\nAfter Reverse. ");
    display();
}

int main()
{
    int choice;
    char ch;

    do{
        printf("\n***Enter Your Choice for Doubly Linked List***\n");
        printf("1. Create a Doubly linked List\n");
        printf("2. Display a Doubly linked List\n");
        printf("3. Insert At Begining\n");
        printf("4. Insert At End\n");
        printf("5. Insert At Specific Position\n");
        printf("6. Delete an element from Begining\n");
        printf("7. Delete an element from End\n");
        printf("8. Delete an element from specific position\n");
        printf("9. Search an element from the List\n");
        printf("10. Sort the Linked List\n");
        printf("11. Count the number of nodes in the list\n");
        printf("12. Reverse the Linked List\n");

        printf("Enter your choice from Above points.... \n");
```

```
scanf(" %d",&choice);
switch(choice){
    case 1:
    create();
    break;

    case 2:
    display();
    break;

    case 3:
    insertat_beg();
    break;

    case 4:
    insertat_end();
    break;

    case 5:
    insertat_pos();
    break;

    case 6:
    deleteat_beg();
    break;

    case 7:
    deleteat_end();
    break;

    case 8:
    deleteat_pos();
    break;

    case 9:
    search();
    break;

    case 10:
    sorting();
    break;


    case 11:int count;
    count=length();
```

```
        printf("\nThe total number of elements in a Linked List is %d: ",count);
        break;

        case 12:
        reverse();
        break;

        default:
        printf("\nInvalid Choice !! Please enter valid choice!!");
        break;
    }
    printf("\nDo You Want to Perform more operations? If yes Enter y/Y :");
    scanf(" %c",&ch);

  }while(ch=='y'||ch=='Y');

  return 0;
}
```

OUTPUT:

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly linked List
2. Display a Doubly linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Delete an element from Begining
7. Delete an element from End
8. Delete an element from specific position
9. Search an element from the List
10. Sort the Linked List
11. Count the number of nodes in the list
12. Reverse the Linked List
Enter your choice from Above points....
2

Your Linked List is :  23  54  56  345  867  231
Do You Want to Perform more operations? If yes Enter y/Y :
```

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly linked List
2. Display a Doubly linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Delete an element from Begining
7. Delete an element from End
8. Delete an element from specific position
9. Search an element from the List
10. Sort the Linked List
11. Count the number of nodes in the list
12. Reverse the Linked List
Enter your choice from Above points....
10
After Sorting :
Your Linked List is :  23  23  345  3334  6546
Do You Want to Perform more operations? If yes Enter y/Y :
```

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly linked List
2. Display a Doubly linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Delete an element from Begining
7. Delete an element from End
8. Delete an element from specific position
9. Search an element from the List
10. Sort the Linked List
11. Count the number of nodes in the list
12. Reverse the Linked List
Enter your choice from Above points....
7

Your Linked List is :  23  23  345  3334
Do You Want to Perform more operations? If yes Enter y/Y :
```

```
1. Create a Doubly linked List
2. Display a Doubly linked List
3. Insert At Begining
4. Insert At End
5. Insert At Specific Position
6. Delete an element from Begining
7. Delete an element from End
8. Delete an element from specific position
9. Search an element from the List
10. Sort the Linked List
11. Count the number of nodes in the list
12. Reverse the Linked List
Enter your choice from Above points....
2

Your Linked List is :  23  345  3334  6546  23
Do You Want to Perform more operations? If yes Enter y/Y :y
```

Ques 11. Write a program to following operations on Singly Circular linked list

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

struct node{
   int data;
   struct node *next;
};
// Circular Lined List using Single Pointer
struct node *tail=NULL;
struct node *newnode;
void create()
{
   int choice;
   tail=NULL;
   do{
      newnode=(struct node *)malloc(sizeof(struct node));
      printf("\nEnter the data into Linked List :\n");
      scanf("%d",&newnode->data);
      newnode->next=NULL;
      if(tail==NULL)
      {
         tail=newnode;
         tail->next=tail;
      }
      else{
         newnode->next=tail->next;
         tail->next=newnode;
         tail=newnode;
      }
      printf("\nDo you want to add more nodes in a Linked list...,if yes enter 1 otherwise 0...");
      scanf("%d",&choice);
   }while(choice==1);
}

void display()
{
   struct node *temp;
   if(tail==NULL)
   {
      printf("Circular Linked List is empty\n");
      return;
   }
   temp=tail->next;
   printf("Your Circular Singly linked List is \n");
```

```
    do{
        printf("%d ",temp->data);
        temp=temp->next;
    }while(temp!=tail->next);
}

int Count_Nodes()
{
    int count;
    struct node *temp;
    count=1;
    temp=tail->next;
    if(temp==NULL)
    {
        printf("\nLinked List is Empty!!\n");
    }
    else{
        while(temp!=tail)
        {
            count++;
            temp=temp->next;
        }
    }
    return count;
}
void insert_At_Beg()
{
    newnode = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data which you want to insert at Begining\n");
    scanf("%d ",&newnode->data);
    if(tail==NULL)
    {
        tail=newnode;
        tail->next=tail;
    }
    else{
        newnode->next=tail->next;
        tail->next=newnode;
    }
    printf("After Inserting a node at Begining .\n");
    display();
}
void insert_At_End()
{
    newnode = (struct node*)malloc(sizeof(struct node));
```

```
    printf("Enter the data which you want to insert at End\n");
    scanf("%d ",&newnode->data);
    if(tail==NULL)
    {
        tail=newnode;
        tail->next=tail;
    }
    else{
        newnode->next=tail->next;
        tail->next=newnode;
        tail=newnode;
    }
    printf("After Inserting a node at End..\n");
    display();
}

void insert_At_Pos()
{
    int pos,count,i;
    i=0;
    newnode = (struct node*)malloc(sizeof(struct node));
    count=Count_Nodes();
    printf("Enter the Position at which you want to insert the node\n");
    scanf("%d",&pos);
    printf("Enter the data which you want to insert at Position\n");
    scanf("%d",&newnode->data);
    if (pos > count)
    {
        printf("\nInvalid Position!! Position should be in range of a linked List :\n");
        printf("The Linked list has only %d nodes,So Please Enter from this range",count);
    }
    else{
        while(i<pos-1)
        {
            tail=tail->next;
            i++;
        }
        newnode->next=tail->next;
        tail->next=newnode;
        printf("\nAfter Inserting a node at Specific Position :\n");
        display();
    }
}

void remove_At_Beg()
```

```
{
   struct node *temp;
   temp=tail->next;
   tail->next=temp->next;
   free(temp);
   printf("\nAfter Remove a Node from  Begining");
   display();
}

void remove_At_End()
{
   struct node *temp1;
   struct node *temp;
   temp1=tail->next;
   while(temp1->next!=tail->next){
      temp=temp1;
      temp1=temp1->next;
   }
   temp->next=temp1->next;
   tail=temp;
   free(temp1);
   printf("\nAfter Remove a Node from  End");
   display();
}

void remove_At_Pos()
{
   int pos,count,i;
   i=1;
   struct node *temp, *temp1;
   count=Count_Nodes();
   printf("Enter the Position at which you want to insert the node\n");
   scanf("%d",&pos);
   if( pos > count)
   {
      printf("\nInvalid Position!! Position shoulb be in range of a linked List :\n");
      printf("The Linked list has only %d nodes,So Please Enter from this range",count);
   }
   else{
      temp=tail->next;
      while(i<pos-1)
      {
         temp=temp->next;
         i++;
      }
```

```
        temp1=temp->next;
        temp->next=temp1->next;
        free(temp1);
        printf("\nAfter Remove a Node from  Given Position ");
        display();
    }
}
void search()
{
    struct node *temp;
    int ser,i,flag;
    i=1;
    flag=0;
    printf("Enter the Element which You want to Search..\n");
    scanf("%d ",&ser);
    temp=tail->next;
    while(temp->next!=tail->next)
    {
        if(ser==temp->data)
        {
            printf("\nElement Found at index %d ",i);
            flag=1;
        }
        i++;
        temp=temp->next;
    }
    if(flag==0)
    {
        printf("\nElement not Found .");
    }
}

void sort() {
    struct node *current;
    struct node *index;
    int temp;
    if (tail == NULL) {
        printf("List is empty\n");
        return;
    }
    current = tail->next;
    if (current == tail) {
        printf("List has only one element\n");
        return;
    }
```

```
    do {
        index = current->next;
        while (index != tail->next) {
            if (current->data > index->data) {
                temp = current->data;
                current->data = index->data;
                index->data = temp;
            }
            index = index->next;
        }
        current = current->next;
    } while (current != tail->next);
    printf("\nAfter Sorting..");
    display();
}

void reverse()
{
    if (tail == NULL || tail->next == tail) {
        printf("List has either 0 or 1 element, no need to reverse.\n");
        return;
    }
    struct node *current;
    struct node *Next;
    struct node *Prev=NULL;
    current=tail->next;
    do {
        Next = current->next;
        current->next = Prev;
        Prev = current;
        current = Next;
    } while (current != tail->next);
    tail = Prev;
    printf("\nAfter Revrese. ");
    display();
}

int main()
{
    int choice;
    char ch;
    do{
        printf("\n***ENTER YOUR CHOICE***\n");
        printf("1. Creating a Circular Linked List\n");
        printf("2. Display a Circular Linked List\n");
```

```
    printf("3. Inserting  an Element at begining\n ");
    printf("4. Insert At End\n");
    printf("5. Insert At Specific Position\n");
    printf("6. Remove from Begining\n");
    printf("7. Remove from Ending\n");
    printf("8. Remove At Specific Position\n");
    printf("9. Search Element from Linked List\n");
    printf("10. Sort the Linked List\n");
    printf("11. Count the Length of the Linked List\n");
    printf("12. Reverse the Linked List\n");

    printf("Enter Your Choice from Above Points.. \n");
    scanf("%d",&choice);
    switch (choice)
    {
       case 1:
       create();
       break;

       case 2:
       display();
       break;

       case 3:
       insert_At_Beg();
       break;

       case 4:
       insert_At_End();
       break;

       case 5:
       insert_At_Pos();
       break;

       case 6:
       remove_At_Beg();
       break;

       case 7:
       remove_At_End();
       break;

       case 8:
       remove_At_Pos();
```

```
            break;

            case 9:
            search();
            break;

            case 10:
            sort();
            break;

            case 11:
            int nodes;
            nodes=Count_Nodes();
            printf("The total number of Nodes in a Circular Linked List is %d ",nodes);
            break;

            case 12:
            reverse();
            break;

            default:printf("\nInvalid Choice, Please enter a valid Choice!!!\n");
            break;
        }
        printf("\nDo You Want to perform more Operations, if Yes Please Enter y or Y...");
        scanf(" %c",&ch);

    }while(ch=='y'||ch=='Y');
    return 0;
}
```

OUTPUT:-

```
***ENTER YOUR CHOICE***
1. Creating a Circular Linked List
2. Display a Circular Linked List
3. Inserting  an Element at begining
 4. Insert At End
5. Insert At Specific Position
6. Remove from Begining
7. Remove from Ending
8. Remove At Specific Position
9. Search Element from Linked List
10. Sort the Linked List
11. Count the Length of the Linked List
12. Reverse the Linked List
Enter Your Choice from Above Points..
2
Your Circular Singly linked List is
12 13 14 15
```

```
1. Creating a Circular Linked List
2. Display a Circular Linked List
3. Inserting  an Element at begining
 4. Insert At End
5. Insert At Specific Position
6. Remove from Begining
7. Remove from Ending
8. Remove At Specific Position
9. Search Element from Linked List
10. Sort the Linked List
11. Count the Length of the Linked List
12. Reverse the Linked List
Enter Your Choice from Above Points..
4
Enter the data which you want to insert at End
55

5
After Inserting a node at End..
Your Circular Singly linked List is
12 13 14 15 55
```

```
 4. Insert At End
5. Insert At Specific Position
6. Remove from Begining
7. Remove from Ending
8. Remove At Specific Position
9. Search Element from Linked List
10. Sort the Linked List
11. Count the Length of the Linked List
12. Reverse the Linked List
Enter Your Choice from Above Points..
7

After Remove a Node from  EndYour Circular Singly linked List is
12 13 14 15
Do You Want to perform more Operations, if Yes Please Enter y or Y..
```

Ques 12. Write a program to following operations on Doubly Circular linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *prev;
   struct node *next;
};
struct node *tail = NULL;
struct node *newnode;
void create()
{
   int choice;
   do
   {
      newnode = (struct node *)malloc(sizeof(struct node));
      printf("\nEnter the data in the newnode : ");
      scanf("%d", &newnode->data);
      newnode->next = NULL;
      newnode->prev = NULL;
      if (tail == NULL)
      {
         tail = newnode;
         tail->next = tail;
         tail->prev = tail;
      }
      else
      {
         newnode->next = tail->next;
         newnode->prev = tail;
         tail->next = newnode;
         tail = newnode;
         tail->next->prev = newnode;
      }
      printf("\nDo you want to enter more Nodes,If yes Please enter 1...");
      scanf("%d", &choice);
   } while (choice == 1);
}
void display()
{
   struct node *temp;
   if (tail == NULL)
   {
      printf("Doubly Circular Linked List is empty\n");
```

```
        return;
    }
    temp = tail->next;
    printf("Your Doubly Circular linked List is \n");
    do
    {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp != tail->next);
}
int Count_Nodes()
{   int count;
    struct node *temp;
    count = 1;
    temp = tail->next;
    if (temp == NULL)
    {
        printf("\nLinked List is Empty!!\n");
    }
    else
    {
        while (temp != tail)
        {
            count++;
            temp = temp->next;
        }}
    return count;
}
void InsertAtBeg()
{
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("\nEnter the data in the newnode : ");
    scanf("%d", &newnode->data);
    newnode->next = NULL;
    newnode->prev = NULL;
    if (tail == NULL)
    {
        tail = newnode;
        tail->next = tail;
        tail->prev = tail;
    }
    else
    {
        newnode->next = tail->next;
        newnode->prev = tail;
```

```
        tail->next = newnode;
        tail->next->prev = newnode;
    }
    printf("\nAfter Inserting a node At BEgining. ");
    display();
}
void InsertAtEnd()
{
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("\nEnter the data in the newnode : ");
    scanf("%d", &newnode->data);
    newnode->next = NULL;
    newnode->prev = NULL;
    if (tail == NULL)
    {
        tail = newnode;
        tail->next = tail;
        tail->prev = tail;
    }
    else
    {
        newnode->prev = tail;
        newnode->next = tail->next;
        tail->next = newnode;
        tail = newnode;
    }
    printf("\nAfter Inserting a node At End. ");
    display();
}
void InsertAtPosition()
{
    int pos, i, count;
    i = 1;
    printf("\nEnter the position at which You want to insert a node \n");
    scanf("%d", &pos);
    count = Count_Nodes();
    if (pos > count)
    {
        printf("\nInvalid Position,Please Enter the position within the range.The Total No. of Node is %d
",count);
        return;
    }
    else
    {
        newnode = (struct node *)malloc(sizeof(struct node));
```

```
        printf("\nEnter the data in the newnode : ");
        scanf("%d", &newnode->data);
        newnode->next = NULL;
        newnode->prev = NULL;
        while (i < pos)
        {
            tail = tail->next;
            i++;
        }
        newnode->next=tail->next;
        newnode->prev=tail;
        tail->next->prev=newnode;
        tail->next=newnode;
    }
    printf("\nAfter Inserting a node At Specific Position. ");
    display();
}
void DeleteAtBeg()
{
    struct node *temp;
    if(tail==NULL)
    {
        printf("\nLinked List is Empty\n");
    }
    else if(tail->next==tail && tail->prev==tail)
    {
        temp=tail;
        free(temp);
        printf("\nNow Your Linked List is empty\n");
    }
    else{
        temp=tail->next;
        tail->next=temp->next;
        temp->next->prev=temp->prev;
        free(temp);
    }
    printf("\nAfter Delete a Node at Begining .");
    display();
}
void DeleteAtEnd()
{
    struct node *temp, *temp1;
    if(tail==NULL)
    {
        printf("\nLinked List is Empty\n");
```

```
    }
    else if(tail->next==tail && tail->prev==tail)
    {
        temp=tail;
        free(temp);
        printf("\nNow Your Linked List is empty\n");
    }
    else{
        temp=tail->next;
        while(temp->next!=tail->next)
        {
            temp1=temp;
            temp=temp->next;
        }
        temp1->next=temp->next;
        tail=temp1;
        tail->next->prev=temp1;
        free(temp);
    }
    printf("\nAfter Delete a Node at End .");
    display();
}
void DeleteAtPosition()
{
    struct node *temp,*temp1;
    int i,pos,count;
    i=1;
    printf("\nEnter the position at which You want to insert a node \n");
    scanf("%d", &pos);
    count = Count_Nodes();
    if (pos > count)
    {
        printf("\nInvalid Position,Please Enter the position within the range.The Total No. of Node is %d
",count);
        return;
    }
    else
    {
        temp=tail->next;
        while(i<pos-1)
        {
            temp=temp->next;
            i++;
        }
        temp1=temp->next;
```

```
        temp->next=temp1->next;
        temp1->next->prev=temp;
        free(temp1);
    }
    printf("\nAfter Deleting a node At Specific Position. ");
    display();
}
void Search()
{
    struct node *temp;
    int ser,i,flag;
    i=1;flag=0;
    printf("\nEnter the element which you want to search :");
    scanf("%d",&ser);
    temp=tail->next;
    while(temp->next!=tail->next)
    {
        if(temp->data==ser)
        {
            printf("\nElement Found at Index :%d",i);
            flag=1;
        }
        i++;
        temp=temp->next;
    }
    if(flag==1)
    {
        printf("\nElement Found");
    }
    else{
        printf("\nElement Not Found");
    }}
void Sorting()
{
    struct node *temp, *temp1;
    int val;
    temp=tail->next;
    while(temp->next!=tail->next)
    {
        temp1=temp->next;
        while(temp1->next!=tail->next)
        {
            if(temp->data > temp1->data)
            {
                val=temp1->data;
```

```
            temp1->data=temp->data;
            temp->data=val;
            if (temp1 == tail) {
                tail = temp;  // Update tail to temp
            } else if (temp == tail) {
                tail = temp1;  // Update tail to temp1
            }
        }
        temp1=temp1->next;
    }
    temp=temp->next;
  }
  printf("After Sorting :");
  display();
}
int main()
{
  int choice;
  char ch;
  do
  {
    printf("\n***Enter Your Choice for Doubly Linked List***\n");
    printf("1. Create a Doubly Circular linked List\n");
    printf("2. Display a Doubly Circular linked List\n");
    printf("3. Count the number of nodes in the list\n");
    printf("4. Insert At Begining\n");
    printf("5. Insert At End\n");
    printf("6. Insert At Specific Position\n");
    printf("7. Delete an element from Begining\n");
    printf("8. Delete an element from End\n");
    printf("9. Delete an element from specific position\n");
    printf("10. Search an element from the List\n");
    printf("11. Sort the Linked List\n");
    printf("12. Reverse the Linked List\n");
    printf("Enter your choice from Above points.... \n");
    scanf(" %d", &choice);
    switch (choice)
    {
    case 1:
      create();
      break;
    case 2:
      display();
    case 3:
      int count;
```

```
            count = Count_Nodes();
            printf("\nThe total number of elements in a Linked List is %d: ", count);
            break;
        case 4:
            InsertAtBeg();
            break;
        case 5:
            InsertAtEnd();
            break;
        case 6:
            InsertAtPosition();
            break;
        case 7:
            DeleteAtBeg();
            break;
        case 8:
            DeleteAtEnd();
            break;
        case 9:
            DeleteAtPosition();
            break;
        case 10:
            Search();
            break;
        case 11:
            Sorting();
            break;
        default:
            printf("InValid Choice Please Enter the Valid choice from above points..\n");
            break;
        }
        printf("\nDo You want to perform more operation?,if yes Enter y/Y...\n");
        scanf(" %c", &ch);

    } while (ch == 'y' || ch == 'Y');

    return 0;
}
```

OUTPUT:-

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly Circular linked List
2. Display a Doubly Circular linked List
3. Count the number of nodes in the list
4. Insert At Begining
5. Insert At End
6. Insert At Specific Position
7. Delete an element from Begining
8. Delete an element from End
9. Delete an element from specific position
10. Search an element from the List
11. Sort the Linked List
12. Reverse the Linked List
Enter your choice from Above points....
2
Your Doubly Circular linked List is
12 56 78 45 90 43
Do You want to perform more operation?,if yes Enter y/Y...
```

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly Circular linked List
2. Display a Doubly Circular linked List
3. Count the number of nodes in the list
4. Insert At Begining
5. Insert At End
6. Insert At Specific Position
7. Delete an element from Begining
8. Delete an element from End
9. Delete an element from specific position
10. Search an element from the List
11. Sort the Linked List
12. Reverse the Linked List
Enter your choice from Above points....
3

The total number of elements in a Linked List is 6:
Do You want to perform more operation?,if yes Enter y/Y..
```

```
1. Create a Doubly Circular linked List
2. Display a Doubly Circular linked List
3. Count the number of nodes in the list
4. Insert At Begining
5. Insert At End
6. Insert At Specific Position
7. Delete an element from Begining
8. Delete an element from End
9. Delete an element from specific position
10. Search an element from the List
11. Sort the Linked List
12. Reverse the Linked List
Enter your choice from Above points....
5

Enter the data in the newnode : 00

After Inserting a node At End. Your Doubly Circular linked List is
12 56 78 45 90 43 0
Do You want to perform more operation?,if yes Enter y/Y...
```

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly Circular linked List
2. Display a Doubly Circular linked List
3. Count the number of nodes in the list
4. Insert At Begining
5. Insert At End
6. Insert At Specific Position
7. Delete an element from Begining
8. Delete an element from End
9. Delete an element from specific position
10. Search an element from the List
11. Sort the Linked List
12. Reverse the Linked List
Enter your choice from Above points....
11
After Sorting :Your Doubly Circular linked List is
12 43 45 56 78 90 0
Do You want to perform more operation?,if yes Enter y/Y...
```

```
***Enter Your Choice for Doubly Linked List***
1. Create a Doubly Circular linked List
2. Display a Doubly Circular linked List
3. Count the number of nodes in the list
4. Insert At Begining
5. Insert At End
6. Insert At Specific Position
7. Delete an element from Begining
8. Delete an element from End
9. Delete an element from specific position
10. Search an element from the List
11. Sort the Linked List
12. Reverse the Linked List
Enter your choice from Above points....
10

Enter the element which you want to search :77

Element Not Found
Do You want to perform more operation?,if yes Enter y/Y.
```

Ques 13. WAP to perform addition and multiplication of two polynomials by creating linked list for every polynomial.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    float coef;
    int exp;
    struct node *next;
};
struct node *start3;
void Display(struct node *start)
{
    if(start == NULL)
      printf("No polynomial Exists");
    else{
        struct node *temp=start;
        while(temp!=NULL)
        {
            printf("(%.1fx^%d)",temp->coef,temp->exp);
            temp=temp->next;
            if(temp!=NULL)
               printf(" + ");
            else
               printf("\n");
        }
    }
}
struct node *insert_node(struct node *start,float coef , int exp)
{
    struct node *temp;
    struct node *newnode;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->coef=coef;
    newnode->exp=exp;
    newnode->next=NULL;

    if(start == NULL || exp > start->exp){
        newnode->next=start;
        start=newnode;
    }
    else{
        temp=start;
        while(temp->next!=NULL && temp->next->exp > exp)
```

```
        {
            temp = temp->next;
        }
        newnode->next = temp->next;
        temp->next = newnode;
    }
    return start;
}
void Add_Poly(struct node *start1,struct node *start2)
{
    struct node *ptr1=start1;
    struct node *ptr2=start2;
    struct node *ptr3=NULL;
    while(ptr1!=NULL && ptr2!=NULL)
    {
        if(ptr1->exp == ptr2->exp)
        {
            start3=insert_node(start3,ptr1->coef + ptr2->coef,ptr1->exp);
            ptr1=ptr1->next;
            ptr2=ptr2->next;
        }
        else if(ptr1->exp > ptr2->exp)
        {
            start3=insert_node(start3,ptr1->coef ,ptr1->exp);
            ptr1=ptr1->next;
        }
        else if(ptr1->exp < ptr2->exp)
        {
            start3=insert_node(start3,ptr2->coef ,ptr2->exp);
            ptr2=ptr2->next;
        }
    }
    while(ptr1!=NULL){
        start3=insert_node(start3,ptr1->coef ,ptr1->exp);
        ptr1=ptr1->next;
    }
    while(ptr2!=NULL){
        start3=insert_node(start3,ptr2->coef ,ptr2->exp);
        ptr2=ptr2->next;
    }
    printf("\nADDITION OF POLYNOMIALS ARE :\n");
    Display(start3);
struct node *create(struct node *start)
{
    int n,i,exp;
```

```
    float coef;
    printf("Enter the number of terms : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("\nEnter the coefficient of term %d :",i+1);
        scanf("%f",&coef);
        printf("\nEnter the exponent of term %d :",i+1);
        scanf("%d",&exp);
        start = insert_node(start,coef,exp);
    }
    return start;
}
void Mul_Poly(struct node *start1,struct node *start2)
{
    struct node *ptr1=start1;
    struct node *ptr2=start2;
    struct node *start3=NULL;

    if(start1 == NULL || start2==NULL)
    {
        printf("Polynomial doesn't exists.\n");
    }
    while(ptr1!=NULL){
        while(ptr2!=NULL)
        {
            start3=insert_node(start3,ptr1->coef * ptr2->coef, ptr1->exp +
ptr2->exp);
            ptr2=ptr2->next;
        }
        ptr1=ptr1->next;
        ptr2=start2;
    }
    printf("\nBefore Simplification, Resultant Polynomial is :\n");
    Display(start3);
    struct node *ptr3=start3;
    struct node *temp=NULL;
    while(ptr3->next !=NULL)
    {
        if(ptr3->exp == ptr3->next->exp){
            ptr3->coef = ptr3->coef + ptr3->next->coef;
            temp = ptr3->next;
            ptr3->next = ptr3->next->next;
            free(temp);
        }
```

```
        else{
            ptr3=ptr3->next;
        }}
    printf("\nAfter Simplification, Resultant Polynomial is :\n");
    Display(start3);
}
int main()
{
    struct node *start1 = NULL;
    struct node *start2 = NULL;
    printf("ENTER THE FIRST POLYNOMIAL\n");
    start1=create(start1);
    printf("\nFIRST POLYNOMIAL\n");
    Display(start1);
    printf("ENTER THE SECOND POLYNOMIAL\n");
    start2=create(start2);
    printf("\nSECOND POLYNOMIAL\n");
    Display(start2);
    Add_Poly(start1,start2);
    Mul_Poly(start1,start2);
    return 0;
}
```

OUTPUT:

```
ENTER THE FIRST POLYNOMIAL
Enter the number of terms : 3

Enter the coefficient of term 1 :3

Enter the exponent of term 1 :2

Enter the coefficient of term 2 :4

Enter the exponent of term 2 :1

Enter the coefficient of term 3 :1

Enter the exponent of term 3 :0

FIRST POLYNOMIAL
(3.0x^2) + (4.0x^1) + (1.0x^0)
```

```
ENTER THE SECOND POLYNOMIAL
Enter the number of terms : 2

Enter the coefficient of term 1 :1

Enter the exponent of term 1 :3

Enter the coefficient of term 2 :4

Enter the exponent of term 2 :4

SECOND POLYNOMIAL
(4.0x^4) + (1.0x^3)
```

```
ADDITION OF POLYNOMIALS ARE :
(4.0x^4) + (1.0x^3) + (3.0x^2) + (4.0x^1) + (1.0x^0)

Before Simplification, Resultant Polynomial is :
(12.0x^6) + (16.0x^5) + (3.0x^5) + (4.0x^4) + (4.0x^4) + (1.0x^3)

After Simplification, Resultant Polynomial is :
(12.0x^6) + (19.0x^5) + (8.0x^4) + (1.0x^3)
```

Ques 14. Write a program to perform following operations on Stack using Arrays.

```
#include <stdio.h>
#include <stdlib.h>
int top = -1;
int stk[10];
void Push(int size)
{
   int data;
   printf("Enter the Elements in Stack: \n");
   scanf("%d", &data);
   if (top == size-1)
   {
      printf("\nStack is FULL,Overflow!!");
   }
   else
   {
      top++;
      stk[top] = data;
   }
}
void Pop()
{
   if (top == -1)
   {
      printf("\nStack is Empty,UnderFlow!!");
   }
   else
   {
      printf("\nDeleted Element is %d", stk[top]);
      top--;
   }
}
void display()
{
   int i;
   printf("\nElements in Stack is :\n");
   for (i = top; i >= 0; i--)
   {
      printf("%d ", stk[i]);
   }
}
void Peek()
{
   if (top == -1)
   {
```

```
      printf("Stack is Empty!!\n");
   }
   else
   {
      printf("Top of the Element in Stack is %d", stk[top]);
   }}
void Search()
{
   int ser, i, flag;
   flag = 0;
   printf("Enter the Element which you want to Search..");
   scanf("%d", &ser);
   if (top == -1)
   {
      printf("Stack is Empty!!\n");
   }
   else
   {
      for (i = top; i >= 0; i--)
      {
         if (ser == stk[i])
         {
            printf("Element %d is Found at index %d", stk[i], i);
            flag = 1;
         }
      }
      if (flag == 0)
      {
         printf("\nElement Not Found ");
      }
   }}}
void Maximum()
{
   int max, i;
   if (top == -1)
   {
      printf("Stack is Empty!!\n");
   }
   else
   {
      max = stk[top];
      for (i = top ; i >= 0; i--)
      {
         if (max < stk[i])
         {
```

```
            max = stk[i];
         }
      }
      printf("Maximum Element in stack is %d at %d index", max,i);
   }}
void Minimum()
{
   int min, i;
   if (top == -1)
   {
      printf("Stack is Empty!!\n");
   }
   else
   {
      min = stk[top];
      for (i = top ; i >= 0; i--)
      {
         if (min > stk[i])
         {
            min = stk[i];
         }
      }
      printf("Minimum Element in stack is %d at %d index", min,i);
   }
}
void Count_Ele()
{
   if(top== -1)
   {
      printf("Stack is Empty\n");
   }
   else {
      printf("The Number of Elements in Stack is %d",top+1);
   }
}
void EXIT()
{
   printf("You have Successfully Executed the program.");
   exit(0);
}
int main()
{
   int size, choice;
   char ch;
   do
```

```
{
   printf("***ENTER YOUR CHOICE***\n");
   printf("1. Push the Elements into Stack\n");
   printf("2. Pop the Elements form the Stack\n");
   printf("3. Display the Stack\n");
   printf("4. Peek Element of the Stack\n");
   printf("5. Search an Element from the Stack\n");
   printf("6. Find Maximum Element\n");
   printf("7. Find Minimum Element\n");
   printf("8. Count the Elements of th Stack\n");
   printf("9. Exit");

   printf("\nEnter the Choice From Above Points....");
   scanf("%d", &choice);
   switch (choice)
   {
   case 1:
      int i;
      printf("\nEnter the size of Stack.\n");
      scanf("%d", &size);
      do
      {
         Push(size);
         printf("\nDo You want to PUSH more data in stack?,If yes Enter 1..");
         scanf("%d", &i);
      } while (i == 1);
      break;
   case 2:
      int j;
      do
      {
         Pop();
         printf("\nDo You want to POP more data from stack?,If yes Enter 1..");
         scanf("%d", &j);
      } while (j == 1);
      break;
      ;
   case 3:
      display();
      break;
   case 4:
      Peek();
      break;
   case 5:
      Search();
```

```
            break;
        case 6:
            Maximum();
            break;
        case 7:
            Minimum();
            break;
        case 8:
            Count_Ele();
            break;
        case 9:
            EXIT();
            break;
        default:
            printf("\nPlease Enter the Valid choice from above points\n");
            break;
        }
        printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
        scanf(" %c", &ch);
    } while (ch == 'Y' || ch == 'y');
    return 0;
}
```

OUTPUT:

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....3

Elements in Stack is :
44 33 22 11
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....2

Deleted Element is 44
Do You want to POP more data from stack?,If yes Enter 1..
```

```
Enter the Choice From Above Points....7
Minimum Element in stack is 11 at -1 index
Do You want to perform more Operations?,If yes Enter y/Y..
```

Ques 15. Write a program to implement stack using linked list and perform following operations on it.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *top = NULL;
void Push()
{
   struct node *newnode;
   newnode = (struct node *)malloc(sizeof(struct node));
   printf("Enter the data into the stack :\n");
   scanf("%d", &newnode->data);
   newnode->next = NULL;

   newnode->next = top;
   top = newnode;
}
void Pop()
{
   struct node *temp;
   if (top == NULL)
   {
      printf("\nStack is UnderFlow!!Stack is empty now");
   }
   else
   {
      temp = top;
      top = top->next;
      printf("Popped Element is %d ",temp->data);
      free(temp);
      printf("\nAfter Popped an element. ");
      Display();
   }
}
void Display()
{
   struct node *temp;
   temp=top;
   printf("The Stack is :\n");
   while(temp!=NULL)
```

```
    {
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
void Peek()
{
    printf("Top Element in the Stack is %d ",top->data);
}
void Search(int ser)
{
    struct node *temp;
    int i,flag;
    temp=top;
    i=1;
    flag=0;
    while(temp!=NULL)
    {
        if(temp->data==ser)
        {
            printf("\nElement is Found at index %d=",i);
            flag=1;
        }
            i++;
            temp=temp->next;
    }
    if(flag==0)
    {
        printf("\nElement is not Found");
    }
}
void Maximum()
{
    struct node *temp;
    int max;
    if(top==NULL)
    {
        printf("Stack is EMPTY !!");
    }
    else{
        temp=top;
        max=temp->data;
        while(temp!=NULL)
        {
            if(max < temp->data)
```

```
        {
            max = temp->data;
        }
        temp=temp->next;
    }
    printf("Maximum Element in Stack is %d :",max);
  }
}
void Minimum()
{
  struct node *temp;
  int min;
  if(top==NULL)
  {
    printf("Stack is EMPTY !!");
  }
  else{
    temp=top;
    min=temp->data;
    while(temp!=NULL)
    {
      if(min > temp->data)
      {
        min = temp->data;
      }
      temp=temp->next;
    }
    printf("Minimum Element in Stack is %d :",min);
  }}
int Count_Ele()
{
  int count;
  struct node *temp;
  count=0;
  if (top==NULL)
  {
    printf("Stack is empty,Elements in Stack is %d ",count);
  }
  else{
    temp=top;
    while (temp!=NULL)
    {
      count++;
      temp=temp->next;
    }
```

```
    printf("The numbers of elements in STack is %d",count);
  }}
void EXIT()
{
  printf("You have Successfully Executed the program.");
  exit(0);
}
int main()
{
  int choice;
  char ch;
  do
  {
    printf("\n***ENTER YOUR CHOICE***\n");
    printf("1. Push the Elements into Stack\n");
    printf("2. Pop the Elements form the Stack\n");
    printf("3. Display the Stack\n");
    printf("4. Peek Element of the Stack\n");
    printf("5. Search an Element from the Stack\n");
    printf("6. Find Maximum Element\n");
    printf("7. Find Minimum Element\n");
    printf("8. Count the Elements of th Stack\n");
    printf("9. Exit");
    printf("\nEnter the Choice From Above Points....");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      int i;
      do
      {
        Push();
        printf("\nDo You want to PUSH more data in stack?,If yes Enter 1..");
        scanf("%d", &i);
      } while (i == 1);
      break;
    case 2:
      int j;
      do
      {
        Pop();
        printf("\nDo You want to POP more data from stack?,If yes Enter 1..");
        scanf("%d", &j);
      } while (j == 1);
      break;
```

```
        case 3:
            Display();
            break;
        case 4:
            Peek();
            break;
        case 5:
            int ser;
            printf("Enter the Element which you want to search \n");
            scanf("%d",&ser);
            Search(ser);
            break;
        case 6:
            Maximum();
            break;
        case 7:
            Minimum();
            break;
        case 8:
            Count_Ele();
            break;
        case 9:
            EXIT();
            break;
        default:
            printf("\nPlease Enter the Valid choice from above points\n");
            break;
        }
        printf("\n\nDo You want to perform more Operations?,If yes Enter y/Y..");
        scanf(" %c", &ch);

    } while (ch == 'Y' || ch == 'y');
    return 0;
}
```

OUTPUT:

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....3
The Stack is :
32 89 67 34 11

Do You want to perform more Operations?,If yes Enter y/Y..
```

```
Do You want to perform more Operations?,If yes Enter y/Y..y

***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....4
Top Element in the Stack is 32

Do You want to perform more Operations?,If yes Enter y/Y..
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....5
Enter the Element which you want to search
32

Element is Found at index 1=
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....6
Maximum Element in Stack is 89 :
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....7
Minimum Element in Stack is 11 :
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack
4. Peek Element of the Stack
5. Search an Element from the Stack
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of th Stack
9. Exit
Enter the Choice From Above Points....2
Popped Element is 32
After Popped an element. The Stack is :
89 67 34 11
```

Ques 16. Write a program to convert an expression from Infix to Postfix using Stack.

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

#define MAX_SIZE 100

int top = -1;
char stack[MAX_SIZE];

void push(char c)
{
   if (top == MAX  SIZE - 1)
   {
      printf("Stack overflow\n");
      return;
   }
   stack[++top] = c;
}

char pop()
{
   if (top == -1)
   {
      printf("Stack underflow\n");
      return;
   }
   return stack[top--];
}

int isEmpty()
{
   return top == -1;
}

char peek()
{
   if (top == -1)
   {
      printf("\nStack underflow\n");
      return;
   }
   return stack[top];
}
```

```
int precedence(char op)
{
   switch (op)
   {
      case '+':
      case '-':
         return 1;
      case '*':
      case '/':
         return 2;
      case '^':
         return 3;
      default:
         return 0;
   }
}

void infixToPostfix(char *infix, char *postfix)
{
   int i, j = 0;
   char token, popped;

   for (i = 0; infix[i] != '\0'; i++)
   {
      token = infix[i];
      if (isalnum(token))
      {
         postfix[j++] = token;
      }
      else if (token == '(')
      {
         push(token);
      }
      else if (token == ')')
      {
         while ((popped = pop()) != '(')
          {
            postfix[j++] = popped;
         }
      }
       else
      {
         while (!isEmpty() && precedence(peek()) >= precedence(token))
          {
```

```
            postfix[j++] = pop();
        }
        push(token);
    }
  }

  while (!isEmpty())
   {
     postfix[j++] = pop();
  }
  postfix[j] = '\0';
}

int main()
{
  char infix[MAX_SIZE], postfix[MAX_SIZE];
  printf("\nEnter an infix expression: ");
  gets(infix);
  infixToPostfix(infix, postfix);
  printf("\nPostfix expression: %s\n", postfix);
  return 0;
}
```

OUTPUT:-

```
Enter an infix expression: (A+(B^C)*D+E)

Postfix expression: ABC^D*+E+
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 17. Write a program to evaluate a Postfix expression using Stack.

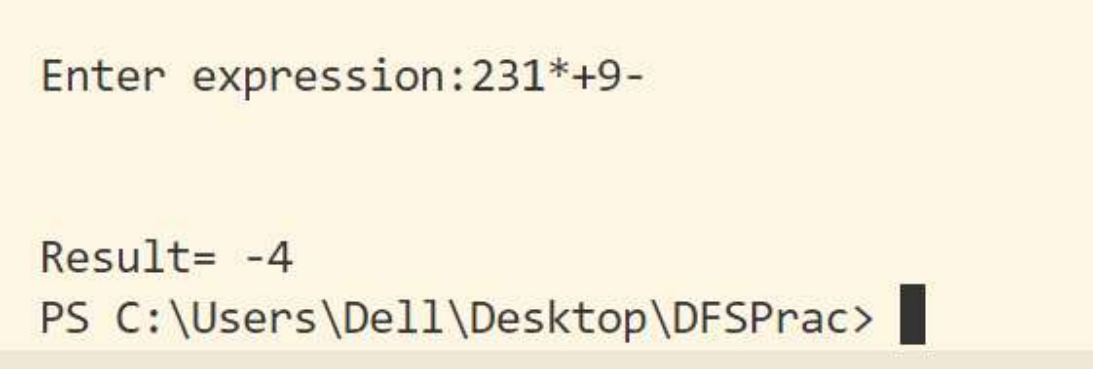
```
#include <stdio.h>
#include <stdlib.h>
int stack[10];
int top = -1;
void push(int item)
{
 if (top >= 9)
 {
  printf("Stack Overflow\n");
  return;
 }
 top++;
 stack[top] = item;
}
int pop()
{
 if (top < 0)
 {
  printf("Stack Underflow\n");
  return -1;
 }
 int item = stack[top];
 top--;
 return item;
}

int evaluate(char *expression)
{
 int i = 0;
 char symbol = expression[i];
 int operand1, operand2, result;

 while (symbol != '\0')
 {
  if (symbol >= '0' && symbol <= '9')
  {
   int num = symbol - '0';
   push(num);
  }
  else
  {
   operand2 = pop();
   operand1 = pop();
   switch (symbol)
```

```
    {
    case '+':
      result = operand1 + operand2;
      break;
    case '-':
      result = operand1 - operand2;
      break;
    case '*':
      result = operand1 * operand2;
      break;
    case '/':
      result = operand1 / operand2;
      break;
    }
    push(result);
   }
   i++;
   symbol = expression[i];
  }
  result = pop();
  return result;
}
int main()
{
  char expression[80];
  printf("\nEnter expression:");
  gets(expression);
  int result = evaluate(expression);
  printf("\n\nResult= %d", result);
  return 0;
}
```

OUTPUT: -



Ques 18. Write a program to implement Linear queue using array and perform the following operation on it.

```
#include <stdio.h>
int front, rear;
front = -1;
rear = -1;
int que[10];
void Enqueue(int size)
{
   int data;
   printf("Enter the data :\n");
   scanf("%d", &data);
   if (rear == size - 1)
   {
      printf("\nQueue is OverFlow!!");
   }
   else if (front == -1 && rear == -1)
   {
      front = 0;
      rear = 0;
      que[rear] = data;
   }
   else
   {
      rear = rear + 1;
      que[rear] = data;
   }}
void Dequeue()
{
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
   }
   else if (front == rear)
   {
      printf("Deleted Element is %d ", que[front]);
      front = -1;
      rear = -1;
   }
   else
   {
      printf("Deleted Element from front is %d ", que[front]);
      front = front + 1;
   }}
void Display()
```

```
{
   int i;
   printf("Your Queue is :\n");
   for (i = front; i <= rear; i++)
   {
      printf("%d ", que[i]);
   }
void Peek()
{
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
   }
   else if (front == rear)
   {
      printf("Front Element is %d ", que[front]);
   }
   else
   {
      printf("Front Element is %d ", que[front]);
   }

void Search()
{
   int ser, flag, i;
   flag = 0;
   printf("\nEnter the element which you want to serach from the Queue :\n");
   scanf("%d ", &ser);
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty");
   }
   else
   {
      for (i = front; i < rear; i++)
      {
         if (que[i] == ser)
         {
            printf("Element %d is found at index %d", ser, i);
            flag = 1;
         }
      }
      if (flag == 0)
      {
         printf("\nElement Not Found ");
```

```
    }
  }
void Maximum()
{
  int max, i;
  max = que[front];
  if (front == -1 && rear == -1)
  {
    printf("Queue is Empty!!");
  }
  else
  {
    for (i = front; i < rear; i++)
    {
      if (max < que[i])
      {
        max = que[i];
      }
    }
    printf("\nMaximum element in Queue is %d ", max);
  }
void Minimum()
{
  int min, i;
  min = que[front];
  if (front == -1 && rear == -1)
  {
    printf("Queue is Empty!!");
  }
  else
  {
    for (i = front; i < rear; i++)
    {
      if (min > que[i])
      {
        min = que[i];
      }
    }
    printf("\nMinimum element in Queue is %d ", min);
  }
void Count_Ele()
{
  int count;
  count = 0;
  if (front == -1 && rear == -1)
```

```
    {
        printf("Queue is Empty!!");
    }
    else if (front == rear)
    {
        count++;
        printf("There is only %d element in Queue.\n", count);
    }
    else
    {
        printf("Total Number of Elements in Queue is %d ", rear + 1);
    }
void EXIT()
{
    printf("You have Successfully Executed the program.");
    exit(0);
}
int main()
{
    int size, choice;
    char ch;
    do
    {
        printf("***ENTER YOUR CHOICE***\n");
        printf("1. Enqueue Elements into Queue\n");
        printf("2. Dequeue the Elements form the Queue\n");
        printf("3. Display the Queue\n");
        printf("4. Peek Element of the Queue\n");
        printf("5. Search an Element from the Queue\n");
        printf("6. Find Maximum Element\n");
        printf("7. Find Minimum Element\n");
        printf("8. Count the Elements of the Queue\n");
        printf("9. Exit");
        printf("\nEnter the Choice From Above Points....");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int i;
            printf("\nEnter the size of Queue.\n");
            scanf("%d", &size);
            do
            {
                Enqueue(size);
                printf("\nDo You want to more data in Queue?,If yes Enter 1..");
```

```
            scanf("%d", &i);
        } while (i == 1);
        break;
    case 2:
        int j;
        do
        {
            Dequeue();
            printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
            scanf("%d", &j);
        } while (j == 1);
        break;
    case 3:
        Display();
        break;
    case 4:
        Peek();
        break;
    case 5:
        Search();
        break;
    case 6:
        Maximum();
        break;
    case 7:
        Minimum();
        break;
    case 8:
        Count_Ele();
        break;
    case 9:
        EXIT();
        break;
    default:
        printf("\nPlease Enter the Valid choice from above points\n");
        break;
    }
    printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
    scanf(" %c", &ch);
  } while (ch == 'Y' || ch == 'y');
  return 0;
}
```

OUTPUT:

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....3
Your Queue is :
12 34 56 78 34
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....2
Deleted Element from front is 12
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....4
Front Element is 34
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....6

Maximum element in Queue is 78
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....7

Minimum element in Queue is 34
```

Ques 19. Write a program to implement Linear queue using Linked List and perform the following operation on it.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *front = NULL;
struct node *rear = NULL;
void Enqueue()
{
   struct node *newnode;
   newnode = (struct node *)malloc(sizeof(struct node));
   printf("\nEnter the data into Queue :\n");
   scanf("%d", &newnode->data);
   newnode->next = NULL;
   if (front == NULL && rear == NULL)
   {
      front = newnode;
      rear = newnode;
      front->next = NULL;
      rear->next = NULL;
   }
   else
   {
      rear->next = newnode;
      rear = rear->next;
      rear->next = NULL;
   }
void Dequeue()
{
   struct node *temp;
   temp = front;
   if (front == NULL && rear == NULL)
   {
      printf("\nQueue is Empty!!");
   }
   else if (front == rear)
   {
      printf("\nQueue has only 1 element.");
      printf("\nDeleted Element is %d :", temp->data);
   }
```

```
    else
    {
        front = front->next;
        printf("\nDeleted Element is %d :", temp->data);
        free(temp);
    }
void Display()
{
    struct node *temp;
    if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty!!UNDERFLOW");
    }
    else if (front == rear)
    {
        printf("\nQueue has only 1 element.");
        printf("%d ", front->data);
    }
    else
    {
        temp = front;
        printf("\nQueue is. ");
        while (temp != NULL)
        {
            printf("%d ", temp->data);
            temp = temp->next;
        }
void Peek()
{
    if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty!!UNDERFLOW");
    }
    else if (front == rear)
    {
        printf("\nQueue has only 1 element.");
        printf("%d ", front->data);
    }
    else
    {
        printf("\nFront element in Queue is.%d ", front->data);
    }
void Search()
{
    struct node *temp;
```

```
    int ser, flag, i;
    flag = 0;
    i = 1;
    if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty!!UNDERFLOW");
    }
    else
    {
        printf("\nEnter the element which you want to search..");
        scanf("%d",&ser);
        temp = front;
        while (temp != NULL)
        {
            if (ser == temp->data)
            {
                printf("\nElement Found at index %d ", i);
                flag = 1;
            }
            i++;
            temp=temp->next;
        }
        if (flag == 0)
        {
            printf("\nElement NOT FOUND ");
        }
    }
void Maximum()
{
    struct node *temp;
    if (front == rear)
    {
        printf("\nThere is only one element in Queue so the Maximum element is :%d", front->data);
    }
    else if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty.");
    }
    else
    {
        int max;
        temp = front;
        max = front->data;
        while (temp != NULL)
        {
```

```
        if (max < temp->data)
        {
            max = temp->data;
        }
        temp = temp->next;
    }
    printf("\nMaximum element in Queue is %d", max);
}
void Minimum()
{
    struct node *temp;
    if (front == rear)
    {
        printf("\nThere is only one element in Queue so the Minimum element is :%d", front->data);
    }
    else if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty.");
    }
    else
    {
        int min;
        temp = front;
        min = front->data;
        while (temp != NULL)
        {
            if (min > temp->data)
            {
                min = temp->data;
            }
            temp = temp->next;
        }
        printf("\nMinimum element in Queue is %d", min);
    }
int Count_Ele()
{
    int count;
    struct node *temp;
    count = 0;
    if (front == NULL && rear == NULL)
    {
        printf("Queue is empty,Elements in Queue is %d ", count);
    }
    else
    {
```

```
    temp = front;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    printf("The numbers of elements in Queue is %d", count);
}
void EXIT()
{
  printf("You have Successfully Executed the program.");
  exit(0);
}
int main()
{
  int choice;
  char ch;
  do
  {
    printf("***ENTER YOUR CHOICE***\n");
    printf("1. Enqueue Elements into Queue\n");
    printf("2. Dequeue the Elements form the Queue\n");
    printf("3. Display the Queue\n");
    printf("4. Peek Element of the Queue\n");
    printf("5. Search an Element from the Queue\n");
    printf("6. Find Maximum Element\n");
    printf("7. Find Minimum Element\n");
    printf("8. Count the Elements of the Queue\n");
    printf("9. Exit");
    printf("\nEnter the Choice From Above Points....");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      int i;
      do
      {
        Enqueue();
        printf("\nDo You want to more data in Queue?,If yes Enter 1..");
        scanf("%d", &i);
      } while (i == 1);
      break;
    case 2:
      int j;
      do
```

```
        {
            Dequeue();
            printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
            scanf("%d", &j);
        } while (j == 1);
        break;
    case 3:
        Display();
        break;
    case 4:
        Peek();
        break;
    case 5:
        Search();
        break;
    case 6:
        Maximum();
        break;
    case 7:
        Minimum();
        break;
    case 8:
        Count_Ele();
        break;
    case 9:
        EXIT();
        break;
    default:
        printf("\nPlease Enter the Valid choice from above points\n");
        break;
    }
    printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
    scanf(" %c", &ch);

  } while (ch == 'Y' || ch == 'y');
  return 0;
}
```

OUTPUT:

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....3

Queue is. 12 51 678 345 7 45
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....2

Deleted Element is 12 :
```

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....5

Enter the element which you want to search..44

Element NOT FOUND
```

```
Element NOT FOUND
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....8
The numbers of elements in Queue is 5
```

Ques 20. Write a program to implement Circular queue using array and perform the following operation on it.

```
#include <stdio.h>
int front, rear;
front = -1;
rear = -1;
int que[10];
void Enqueue(int size)
{
   int data;
   printf("Enter the data :\n");
   scanf("%d", &data);
   if ((rear+1)%size == front)
   {
      printf("\nQueue is OverFlow!!");
   }
   else if (front == -1 && rear == -1)
   {
      front = 0;
      rear = 0;
      que[rear] = data;
   }
   else
   {
      rear = (rear + 1)%size;
      que[rear] = data;
   }
void Dequeue(int size)
{
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
   }
   else if (front == rear)
   {
      printf("Deleted Element is %d ", que[front]);
      front = -1;
      rear = -1;
   }
   else
   {
      printf("Deleted Element from front is %d ", que[front]);
      front = (front + 1)%size;
   }
```

```
void Display(int size)
{
   int i;
   printf("Your Queue is :\n");
   for (i = front; i != rear; i=(i+1)%size)
   {
      printf("%d ", que[i]);
   }
      printf("%d ", que[i]);
}
void Peek()
{
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
   }
   else if (front == rear)
   {
      printf("Front Element is %d ", que[front]);
   }
   else
   {
      printf("Front Element is %d ", que[front]);
   }
void Search(int size)
{
   int ser, flag, i;
   flag = 0;
   printf("\nEnter the element which you want to serach from the Queue :\n");
   scanf("%d ", &ser);
   if (front == -1 && rear == -1)
   {
      printf("\nQueue is Empty");
   }
   else
   {
      for (i = front; i != rear; i=(i+1)%size)
      {
         if (que[i] == ser)
         {
            printf("Element %d is found at index %d", ser, i);
            flag = 1;
         }
      }
      if (flag == 0)
```

```
        {
            printf("\nElement Not Found ");
        }
void Maximum(int size)
{
    int max, i;
    max = que[front];
    if (front == -1 && rear == -1)
    {
        printf("Queue is Empty!!");
    }
    else
    {
        for (i = front; i != rear; i=(i+1)%size)
        {
            if (max < que[i])
            {
                max = que[i];
            }
        }
        printf("\nMaximum element in Queue is %d ", max);
    }
void Minimum(int size)
{
    int min, i;
    min = que[front];
    if (front == -1 && rear == -1)
    {
        printf("Queue is Empty!!");
    }
    else
    {
        for (i = front; i != rear; i=(i+1)%size)
        {
            if (min > que[i])
            {
                min = que[i];
            }
        }
        printf("\nMinimum element in Queue is %d ", min);
    }
void Count_Ele()
{
    int count;
    count = 0;
```

```
    if (front == -1 && rear == -1)
    {
        printf("Queue is Empty!!");
    }
    else if (front == rear)
    {
        count++;
        printf("There is only %d element in Queue.\n", count);
    }
    else
    {
        printf("Total Number of Elements in Queue is %d ", rear + 1);
    }
void EXIT()
{
    printf("You have Successfully Executed the program.");
    exit(0);
}
int main()
{
    int size, choice;
    char ch;
    printf("\nEnter the size of Queue.\n");
    scanf("%d", &size);
    do
    {
        printf("***ENTER YOUR CHOICE***\n");
        printf("1. Enqueue Elements into Queue\n");
        printf("2. Dequeue the Elements form the Queue\n");
        printf("3. Display the Queue\n");
        printf("4. Peek Element of the Queue\n");
        printf("5. Search an Element from the Queue\n");
        printf("6. Find Maximum Element\n");
        printf("7. Find Minimum Element\n");
        printf("8. Count the Elements of the Queue\n");
        printf("9. Exit");
        printf("\nEnter the Choice From Above Points....");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int i=0;
            do
            {
                Enqueue(size);
```

```
            i++;
        } while (i <size);
        break;
    case 2:
        int j;
        do
        {
            Dequeue(size);
            printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
            scanf("%d", &j);
        } while (j == 1);
        break;
    case 3:
        Display(size);
        break;
    case 4:
        Peek();
        break;
    case 5:
        Search(size);
        break;
    case 6:
        Maximum(size);
        break;
    case 7:
        Minimum(size);
        break;
    case 8:
        Count_Ele();
        break;
    case 9:
        EXIT();
        break;
    default:
        printf("\nPlease Enter the Valid choice from above points\n");
        break;
    }
    printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
    scanf(" %c", &ch);
  } while (ch == 'Y' || ch == 'y');
  return 0;
}
```

OUTPUT:

```
Enter the size of Queue.
6
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....1
Enter the data :
23
Enter the data :
45
Enter the data :
78
Enter the data :
99
Enter the data :
```

```
Do You want to perform more Operations?,If yes
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....3
Your Queue is :
23 45 78 99 45 22
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....4
Front Element is 23
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....2
Deleted Element from front is 23
Do You want to dequeue more data from Queue?,If yes Enter 1..
1
Deleted Element from front is 45
Do You want to dequeue more data from Queue?,If yes Enter 1..22

Do You want to perform more Operations?,If yes Enter y/Y..y
```

```
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....1
Enter the data :
33
Enter the data :
22
Enter the data :
222
```

Ques 21. Write a program to implement Circular queue using Linked List and perform the following operation on it.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *front = NULL;
struct node *rear = NULL;
void Enqueue()
{
   struct node *newnode;
   newnode = (struct node *)malloc(sizeof(struct node));
   printf("\nEnter the data into Queue :\n");
   scanf("%d", &newnode->data);
   newnode->next = NULL;

   if (front == NULL && rear == NULL)
   {
      front = newnode;
      rear = newnode;
      rear->next = NULL;
   }
   else
   {
      rear->next = newnode;
      rear = newnode;
      rear->next = front;
   }
void Dequeue()
{
   struct node *temp;
   temp=front;
   if (front == NULL && rear == NULL)
   {
      printf("\nQueue is Empty!!");
   }
   else if (front == rear)
   {
      printf("\nQueue has only 1 element.");
      printf("\nDeleted Element is %d :", temp->data);
      front=NULL;
```

```
        rear=NULL;
    }
    else
    {
        front = front->next;
        rear->next=front;
        printf("\nDeleted Element is %d :", temp->data);
        free(temp);
    }
void Display()
{
    struct node *temp;
    if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty!!UNDERFLOW");
    }
    else if (front == rear)
    {
        printf("\nQueue has only 1 element.");
        printf("%d ", front->data);
    }
    else
    {
        temp = front;
        printf("\nQueue is. ");
        while (temp->next != front)
        {
            printf("%d ", temp->data);
            temp = temp->next;
        }
            printf("%d ", temp->data);
    }
void Peek()
{
    if (front == NULL && rear == NULL)
    {
        printf("\nQueue is Empty!!UNDERFLOW");
    }
    else if (front == rear)
    {
        printf("\nQueue has only 1 element.");
        printf("%d ", front->data);
    }
    else
    {
```

```
    printf("\nFront element in Queue is.%d ", front->data);
  }
void Search()
{
  struct node *temp;
  int ser, flag, i;
  flag = 0;
  i = 1;
  if (front == NULL && rear == NULL)
  {
    printf("\nQueue is Empty!!UNDERFLOW");
  }
  else
  {
    printf("\nEnter the element which you want to search..");
    scanf("%d",&ser);
    temp = front;
    while (temp->next != front)
    {
      if (ser == temp->data)
      {
        printf("\nElement Found at index %d ", i);
        flag = 1;
      }
      i++;
      temp=temp->next;
    }
    if (flag == 0)
    {
      printf("\nElement NOT FOUND ");
    }
  }
void Maximum()
{
  struct node *temp;
  if (front == rear)
  {
    printf("\nThere is only one element in Queue so the Maximum element is :%d", front->data);
  }
  else if (front == NULL && rear == NULL)
  {
    printf("\nQueue is Empty.");
  }
  else
  {
```

```
    int max;
    temp = front;
    max = front->data;
    while (temp->next != front)
    {
      if (max < temp->data)
      {
        max = temp->data;
      }
      temp = temp->next;
    }
    printf("\nMaximum element in Queue is %d", max);
  }
void Minimum()
{
  struct node *temp;
  if (front == rear)
  {
    printf("\nThere is only one element in Queue so the Minimum element is :%d", front->data);
  }
  else if (front == NULL && rear == NULL)
  {
    printf("\nQueue is Empty.");
  }
  else
  {
    int min;
    temp = front;
    min = front->data;
    while (temp->next != front)
    {
      if (min > temp->data)
      {
        min = temp->data;
      }
      temp = temp->next;
    }
    printf("\nMinimum element in Queue is %d", min);
  }
int Count_Ele()
{
  int count;
  struct node *temp;
  count = 0;
  if (front == NULL && rear == NULL)
```

```
    {
        printf("Queue is empty,Elements in Queue is %d ", count);
    }
    else
    {
        temp = front;
        while (temp->next != front)
        {
            count++;
            temp = temp->next;
        }
        printf("The numbers of elements in Queue is %d", count);
    }
void EXIT()
{
    printf("You have Successfully Executed the program.");
    exit(0);
}
int main()
{
    int choice;
    char ch;
    do
    {
        printf("***ENTER YOUR CHOICE***\n");
        printf("1. Enqueue Elements into Queue\n");
        printf("2. Dequeue the Elements form the Queue\n");
        printf("3. Display the Queue\n");
        printf("4. Peek Element of the Queue\n");
        printf("5. Search an Element from the Queue\n");
        printf("6. Find Maximum Element\n");
        printf("7. Find Minimum Element\n");
        printf("8. Count the Elements of the Queue\n");
        printf("9. Exit");
        printf("\nEnter the Choice From Above Points....");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int i;
            do
            {
                Enqueue();
                printf("\nDo You want to more data in Queue?,If yes Enter 1..");
                scanf("%d", &i);
```

```
        } while (i == 1);
        break;
    case 2:
        int j;
        do
        {
            Dequeue();
            printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
            scanf("%d", &j);
        } while (j == 1);
        break;
    case 3:
        Display();
        break;
    case 4:
        Peek();
        break;
    case 5:
        Search();
        break;
    case 6:
        Maximum();
        break;
    case 7:
        Minimum();
        break;
    case 8:
        Count_Ele();
        break;
    case 9:
        EXIT();
        break;

    default:
        printf("\nPlease Enter the Valid choice from above points\n");
        break;
    }
    printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
    scanf(" %c", &ch);

  } while (ch == 'Y' || ch == 'y');
  return 0;
}
```

OUTPUT:

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....3

Queue is. 12 56 78 65 90
```

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....4

Front element in Queue is.12
```

```
Do You want to perform more Operations?,If yes Enter y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....6

Maximum element in Queue is 78
```

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue
4. Peek Element of the Queue
5. Search an Element from the Queue
6. Find Maximum Element
7. Find Minimum Element
8. Count the Elements of the Queue
9. Exit
Enter the Choice From Above Points....8
The numbers of elements in Queue is 4
Do You want to perform more Operations?,If yes Enter y/Y
```

Ques 22. Write a program to implement concept of Muti-Stack and Multi-Queue

22(a). Multi-Stack

```
#include <stdio.h>
const int N = 8;
int count;
int top1;
int top2;
count = 0;
top1 = -1;
top2 = -1;
int stk1[10], stk2[10];
void Push1(int data)
{
   if (top1==N-1)
   {
      printf("\nStack is Overflow!!");
   }
   else{
      top1++;
      stk1[top1]=data;
   }
void Enqueue()
{
   int data;
   printf("\nEnter the data in the Queue..");
   scanf("%d",&data);
   Push1(data);
   count++;
}
int Pop1()
{
   if(top1==-1)
   {
      printf("Stack is empty,Underflow!!");
   }
   else{
      return stk1[top1--];
   }
}
void Push2(int data)
{
   if(top2==N-1)
   {
      printf("OverFlow");
```

```
    }
    else{
        top2++;
        stk2[top2]=data;
    }
}
int Pop2()
{
    if(top2==-1)
    {
        printf("Stack is empty,Underflow!!");
    }
    else{
        return stk2[top2--];
    }
}
void Dequeue()
{
    int i,a,b;
    for(i=0;i<count;i++)
    {
        a=Pop1();
        Push2(a);
    }
    b=Pop2();
    printf("\nDeleted Element is %d ",b);
    count--;

    for(i=0;i<count;i++)
    {
        a=Pop2();
        Push1(a);
    }
}
void Display()
{
    int i;
    printf("\nElements in Queue is :");
    for(i=0;i<=top1;i++)
    {
        printf("%d ",stk1[i]);
    }
int main()
{
    int choice;
```

```
    char ch;
    do
    {
        printf("***ENTER YOUR CHOICE***\n");
        printf("1. Enqueue Elements into Queue\n");
        printf("2. Dequeue the Elements form the Queue\n");
        printf("3. Display the Queue\n");
        printf("\nEnter the Choice From Above Points....");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            int i;
            do
            {
                Enqueue();
                printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
                scanf("%d", &i);
            } while (i == 1);
            break;
        case 2:
            int j;
            do
            {
                Dequeue();
                printf("\nDo You want to dequeue more data from Queue?,If yes Enter 1..");
                scanf("%d", &j);
            } while (j == 1);
            break;
        case 3:
            Display();
            break;
        default:
            printf("\nPlease Enter the Valid choice from above points\n");
            break;
        }
        printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
        scanf(" %c", &ch);

    } while (ch == 'Y' || ch == 'y');
    return 0;
}
```

OUTPUT:

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue

Enter the Choice From Above Points....3

Elements in Queue is :34 56 34 89 333
```

```
Do You want to perform more Operations?,If yes Enter y/Y..y
***ENTER YOUR CHOICE***
1. Enqueue Elements into Queue
2. Dequeue the Elements form the Queue
3. Display the Queue

Enter the Choice From Above Points....2

Deleted Element is 34
Do You want to dequeue more data from Queue?,If yes Enter 1..
```

22(b). Multi-Queue

```
#include <stdio.h>
const int N = 8;
int Q1[10], Q2[10];
int front1, rear1;
int front2, rear2;
void Enqueue1(int data)
{
   if (rear1 == N - 1)
   {
      printf("\nQueue is OverFlow!!");
   }
   else if (front1 == -1 && rear1 == -1)
   {
      front1 = 0;
      rear1 = 0;
      Q1[rear1] = data;
   }
   else
   {
      rear1 = rear1 + 1;
      Q1[rear1] = data;
   }
```

```
void Push()
{
   int data;
   printf("\nEnter the data in Stack..");
   scanf("%d", &data);
   Enqueue1(data);
}
int Dequeue1()
{
   if (front1 == -1 && rear1 == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
      return -1;
   }
   else if (front1 == rear1)
   {
      int val=Q1[front1];
      front1 = -1;
      rear1 = -1;
      return val;
   }
   else
   {
      return Q1[front1++];
   }
int Dequeue2()
{
   if (front2 == -1 && rear2 == -1)
   {
      printf("\nQueue is Empty,UNDERFLOW!!\n");
      return -1;
   }
   else if (front2 == rear2)
   {
      int val=Q2[front2];
      front2 = -1;
      rear2 = -1;
      return val;
   }
   else
   {
      return Q2[front2++];
   }
}
void Enqueue2(int data)
```

```
{
   if (rear2 == N - 1)
   {
      printf("\nQueue is OverFlow!!");
   }
   else if (front2 == -1 && rear2 == -1)
   {
      front2 = 0;
      rear2 = 0;
      Q2[rear2] = data;
   }
   else
   {
      rear2 = rear2 + 1;
      Q2[rear2] = data;
   }
}
void Pop()
{
   int i, a, b;
   while (front1 != rear1)
   {
      a = Dequeue1();
      Enqueue2(a);
   }
   int poppedele = Dequeue1();
   if (poppedele != -1) {
      printf("\nPopped Element is %d", poppedele);
   }
   for (i = front2; i <= rear2; i++)
   {
      b = Dequeue2();
      Enqueue1(b);
   }
}
void Display()
{
   int i;
   printf("Your Stack is :\n");
   for (i = rear1; i >= front1; i--)
   {
      printf("%d ", Q1[i]);
   }
}
Void main()
```

```
{
   front1 = -1;
   rear1 = -1;
   front2 = -1;
   rear2 = -1;
   char ch;
   int choice;
   do
   {
      printf("***ENTER YOUR CHOICE***\n");
      printf("1. Push the Elements into Stack\n");
      printf("2. Pop the Elements form the Stack\n");
      printf("3. Display the Stack\n");
      printf("\nEnter the Choice From Above Points....");
      scanf("%d", &choice);
      switch (choice)
      {
      case 1:
         int i;
         do
         {
            Push();
            printf("\nDo You want to PUSH more data in stack?,If yes Enter 1..");
            scanf("%d", &i);
         } while (i == 1);
         break;
      case 2:
         int j;
         do
         {
            Pop();
            printf("\nDo You want to POP more data from stack?,If yes Enter 1..");
            scanf("%d", &j);
         } while (j == 1);
         break;
      case 3:
         Display();
         break;
      default:
         printf("\nPlease Enter the Valid choice from above points\n");
         break;
      }
      printf("\nDo You want to perform more Operations?,If yes Enter y/Y..");
      scanf(" %c", &ch);
   } while (ch == 'Y' || ch == 'y');}
```

OUTPUT:

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack

Enter the Choice From Above Points....3
Your Stack is :
89 454 67 34
Do You want to perform more Operations?,If yes Enter y/Y..
```

```
***ENTER YOUR CHOICE***
1. Push the Elements into Stack
2. Pop the Elements form the Stack
3. Display the Stack

Enter the Choice From Above Points....2

Popped Element is 89
Do You want to POP more data from stack?,If yes Enter 1..0
```

Ques 23. Write a program to implement Merge Sort.

```
#include<stdio.h>

int arr[10];
void Display(int n){
   int i;
   for(i=0;i<=n;i++)
   {
      printf("%d ",arr[i]);
   }
}
void Merge(int lb,int mid,int ub)
{
   int i,j,k;
   int b[10];
   i=lb;j=mid+1;k=lb;
   while(i<=mid && j<=ub)
   {
      if(arr[i] < arr[j]){
         b[k]=arr[i];
         i++;
      }
      else{
         b[k]=arr[j];
         j++;
      }
      k++;
   }
   while(i<=mid)
   {
      b[k]=arr[i];
      i++;k++;
   }
   while(j<=ub)
   {
      b[k]=arr[j];
      j++;k++;
   }
   for(k=lb;k<=ub;k++){
      arr[k]=b[k];
   }
}
void Merge_Sort(int lb,int ub)
{
   int mid;
```

```
    if(lb<ub)
    {
        mid = (lb + ub)/2;
        Merge_Sort(lb,mid);
        Merge_Sort(mid+1,ub);
        Merge(lb,mid,ub);
    }
}
int main()
{
    int n,i,lb,ub;
    printf("\nEnter the size of Array :");
    scanf("%d",&n);
    lb=0;
    ub=n-1;
    printf("\nEnter the elements of an Array :");
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("\n The Array is:");
    Display(ub);
    Merge_Sort(lb,ub);
    printf("\nAfter Sorting the Array is :");
    Display(ub);
    return 0;
}
```

OUTPUT:-

```
Enter the size of Array :5

Enter the elements of an Array :3
56
897
12
34

 The Array is:3 56 897 12 34
After Sorting the Array is :3 12 34 56 897
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 24. Write a program to implement Quick Sort.

```
#include<stdio.h>
int arr[10];
int size;
void display(int arr[10], int low, int high)
{
   int i;
   for(i=low ; i<=high ; i++)
   {
      printf("%d ",arr[i]);
   }
   printf("\n");
}
int partition(int arr[10],int low,int high)
{
   int pivot, start, end;
   pivot=arr[low];
   start=low;
   end=high;
   while(start < end)
   {
      while(arr[start] <= pivot)
      {
         start++;
      }
      while(arr[end] > pivot)
      {
         end--;
      }
      if(start < end)
      {
         int temp = arr[start];
         arr[start] = arr[end];
         arr[end] = temp;
      }
   }
   int temp=arr[end];
   arr[end]=arr[low];
   arr[low]=temp;
   return end;
}
void QuickSort(int arr[10] , int low,int high)
{
   int piv;
```

```
    if(low<high){
        piv=partition(arr,low,high);
        QuickSort(arr,low,piv-1);
        QuickSort(arr,piv+1,high);
    }}
int main()
{
    int i;
    printf("Enter the Size of the Array :\n");
    scanf("%d",&size);
    printf("Enter the Elements of an Array :\n");
    for(i=0;i<size;i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("The ARRAY is: \n");
    display(arr,0,size-1);
    QuickSort(arr,0,size-1);
    printf("\nAfter Sorting,The ARRAY is: \n");
    display(arr,0,size-1);

    return 0;
}
```

OUTPUT:

```
Enter the Size of the Array :
7
Enter the Elements of an Array :
435
123
7
4345
890
12
3
The ARRAY is:
435 123 7 4345 890 12 3

After Sorting,The ARRAY is:
3 7 12 123 435 890 4345
PS C:\Users\Dell\Desktop\DFSPrac>
```

Ques 25. Write a program to implement Shell Sort.

```
#include<stdio.h>
void Display(int arr[10],int size)
{
   int i;
   for(i=0;i<size;i++)
   {
      printf("%d ",arr[i]);
   }}
void Shell_Sort(int arr[10],int size){
   int i,j,gap,temp;
   for(gap=size/2;gap>=1;gap=gap/2){
      for(j=gap;j<size;j++){
         for(i=j-gap;i>=0;i=i-gap)
         {
            if(arr[i+gap] > arr[i])
               break;
            else{
               temp = arr[i+gap];
               arr[i+gap] = arr[i];
               arr[i] = temp;}
            }}}}
   printf("\nAfter Sorting, The Array is:\n");
   Display(arr,size);
}
void main()
{
   int arr[10];
   int size,i;
   printf("Enter the size of Array :");
   scanf("%d",&size);
   printf("\nEnter the Elements of an Array :");
   for(i=0;i<size;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("The Array is:\n");
   Display(arr,size);
   Shell_Sort(arr,size);

}
```

OUTPUT:-

```
Enter the Elements of an Array :67
567
45
34
890
12
33
The Array is:
67 567 45 34 890 12 33
After Sorting, The Array is:
12 33 34 45 67 567 890
PS C:\Users\Dell\Desktop\DFSPrac>
```