Banarsidas Chandiwala Institute Of Information Technology

Data and File Structure
LAB File

Submittedto:-
Dr. Anu Taneja

Submittedby:-
Prachi Sharma
04911104422
MCA 2nd Semester

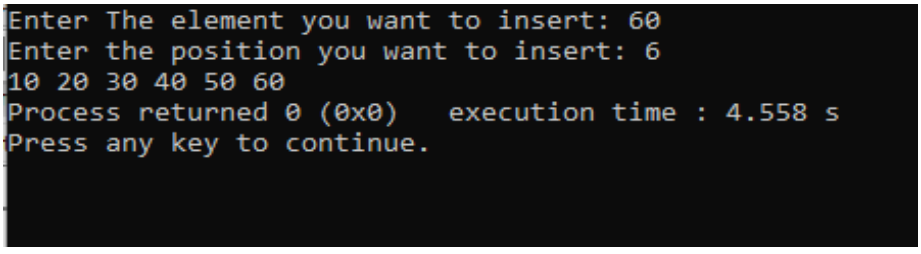
INDEX

	• insertion • deletion • traversal- inorder, preorder, postorder		
45	WAP to implement AVL tree • insertion • deletion • traversal- inorder, preorder, postorder	174	
46	WAP to store the graph information in form of adjacency matrix	182	
47	WAP to store the graph information in form of adjacency list	184	

1.Write a c program to insert an element into an array.

```
#include<stdio.h>
int main(){
   int arr[10]={10,20,30,40,50};
   int pos,value,i,size,j;
   printf("Enter The element you want to insert: ");
   scanf("%d",&value);
   printf("Enter the index at which you want to insert: ");
   scanf("%d",&pos);
   size=sizeof(arr)/sizeof(arr[0]);
for(i=size-2;i>=pos;i--){
    arr[i+1]=arr[i];
}
arr[pos]=value;
for(j=0;j<size;j++){
    if(arr[j]!=0){
        printf("%d ",arr[j]);
    }
}
return 0;
}
```

Output-

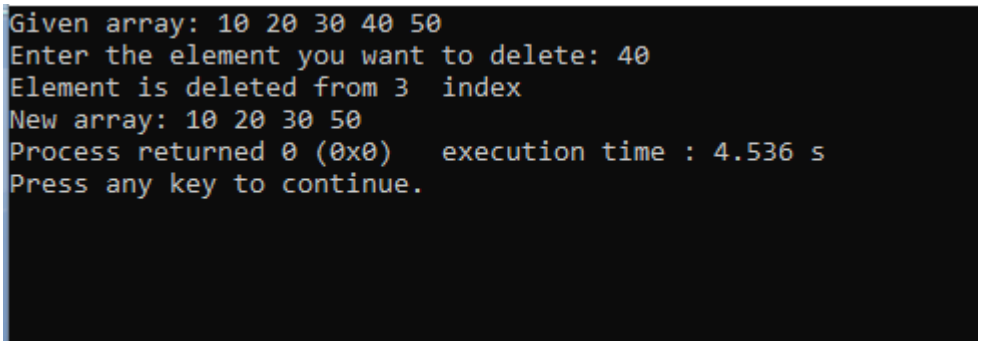


2.Write a program to perform deletion operation in the array.

```
#include<stdio.h>
int main()
{
   int arr[]={10,20,30,40,50};
   int size,del,flag=0;
   size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
   printf("\n");
   printf("Enter the element you want to delete: ");
   scanf("%d",&del);
   for(int i=0;i<size;i++){
      if(arr[i]==del){
         for(int j=i;j<size;j++){
            arr[j]=arr[j+1];
         }
         printf("Element is deleted from %d  index ",i);
         printf("\n");
         flag++;
      }
   }
   if(flag!=0){
```

```
        printf("New array: ");
        for(int k=0;k<size-1;k++){
            printf("%d ",arr[k]);
        }
    }
    else{
        printf("No such element exist in the array");
    }
    return 0;
}
```

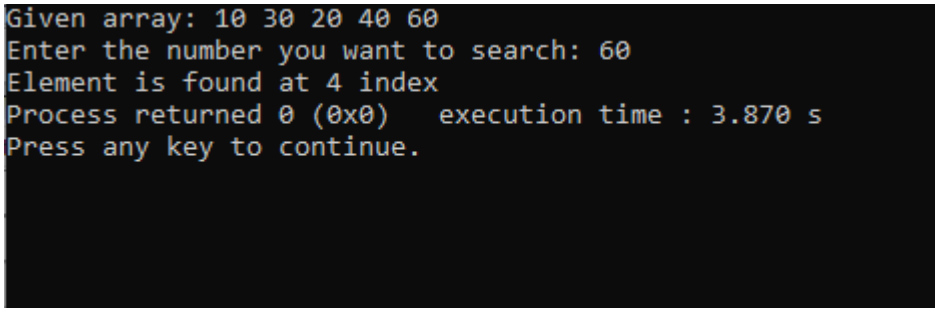
Output-



3(i).Write a program to implement linearsearch and print its first occurrence.

```
#include<stdio.h>
int main(){
   int arr[5]={10,30,20,40,60};
   int ser,flag=0;
   int size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
    printf("%d ",arr[i]);
   }
   printf("\n");
   printf("Enter the number you want to search: ");
   scanf("%d",&ser);
   for(int i=0;i<size;i++){
       if(arr[i]==ser){
           printf("Element is found at %d index ",i);
           flag++;
        break;
       }
}
if(flag==0){
    printf("No such element exist in the array");
}
return 0;
}
```

Output-

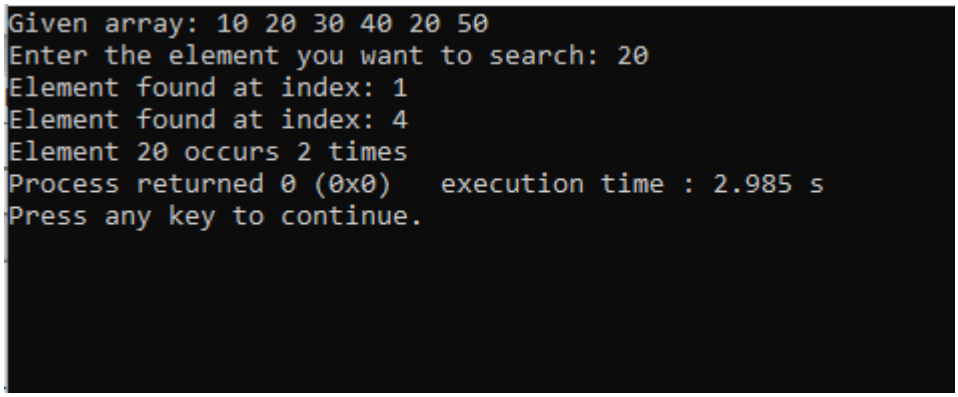

Given array: 10 30 20 40 60
Enter the number you want to search: 60
Element is found at 4 index
Process returned 0 (0x0)   execution time : 3.870 s
Press any key to continue.

3(ii). Write a program to implement linear search with count of multiple occurrence.

```
#include<stdio.h>
int main(){
  int arr[]={10,20,30,40,20,50};
  int size,ser,count=0,flag=0;
  size=sizeof(arr)/sizeof(arr[0]);
  printf("Given array: ");
  for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
  }
  printf("\n");
  printf("Enter the element you want to search: ");
  scanf("%d",&ser);
  for(int i=0;i<size;i++){
     if(arr[i]==ser){
          count++;
          printf("Element found at index: %d \n",i);
          flag=1;
     }
  }
  if(flag==0){
      printf("No such element exist");
  }
  else{
```

```
        printf("Element %d occurs %d times ",ser,count);
    }


    return 0;
}
```

Output-

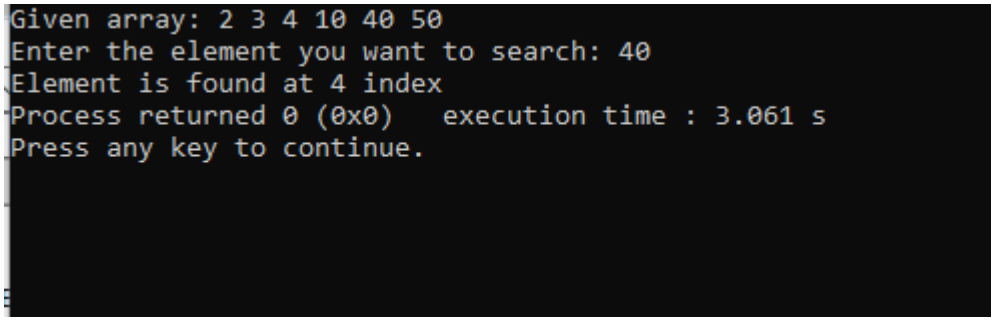


4.(i). Write a program to implement binary search without recursion

```
#include<stdio.h>
int main(){
  int arr[]={2,3,4,10,40,50};
  int ser,mid;
  int size=sizeof(arr)/sizeof(arr[0]);
  printf("Given array: ");
  for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
  }
  printf("\n");
  printf("Enter the element you want to search: ");
  scanf("%d",&ser);
  int lb=0,ub=size-1;
  mid=(lb+ub)/2;
  while(lb<=ub){
    if(arr[mid]==ser){
      printf("Element is found at %d index",mid);
      break;
    }
    else if(arr[mid]>ser){
       ub=mid-1;
    }
    else{
       lb=mid+1;
```

```
    }
    mid=(lb+ub)/2;
}
if(lb>ub){
   printf("Element is not present in the given array");
}
return 0;
}
```

Output-

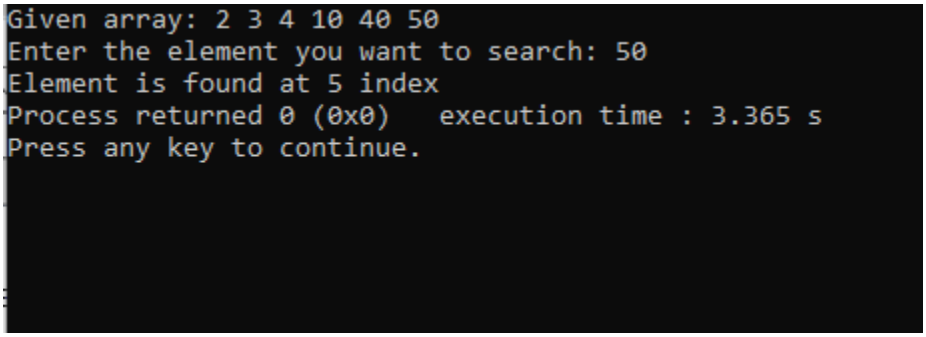


4.(ii).Write a program to implement binary search with recursion.

```
#include<stdio.h>
int binary_recursion(int arr[],int ser,int lb,int ub,int mid){
   if(lb>ub){
      return -1;
   }
   else{
      mid=(lb+ub)/2;
      if(arr[mid]==ser){
         return mid;
      }
      else if(arr[mid]>ser){
         return binary_recursion(arr,ser,lb,mid-1,mid);
      }
      else{
         return binary_recursion(arr,ser,mid+1,ub,mid);
      }
   }
}
int main(){
   int arr[]={2,3,4,10,40,50};
   int ser,mid;
   int size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
```

```
        printf("%d ",arr[i]);
    }
    printf("\n");
    printf("Enter the element you want to search: ");
    scanf("%d",&ser);
    int lb=0,ub=size-1;
    mid=(lb+ub)/2;
    int found_index=binary_recursion(arr,ser,lb,ub,mid);
    if(found_index<0){
        printf("Element is not exist in the array");
    }
    else{
        printf("Element is found at %d index ",found_index);
    }
    return 0;
}
```

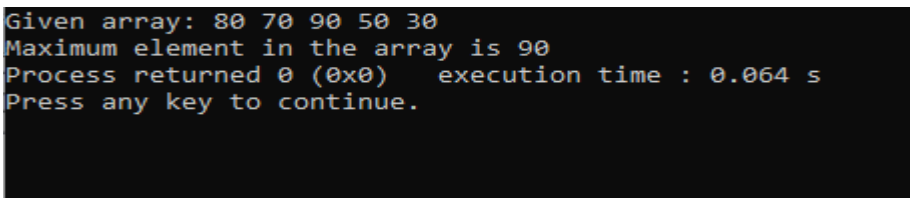
Output-



5.Write a program to find the largest element in the array.

```
#include<stdio.h>
int main()
{
   int arr[]={80,70,90,50,30};
   int size,max;
   printf("Given array: ");;
   size=sizeof(arr)/sizeof(arr[0]);
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
   printf("\n");
   max=arr[0];
   for(int i=1;i<size;i++){
      if(arr[i]>max){
         max=arr[i];
      }
   }
   printf("Maximum element in the array is %d",max);
   return 0;
}
```

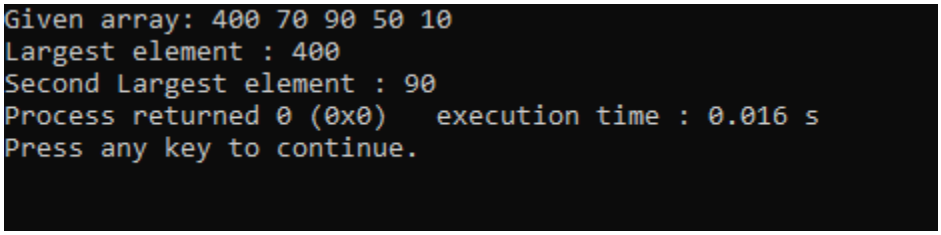
Output-



6.Write a program to find the second largest element in the array.

```
#include<stdio.h>
#include<limits.h>
int main()
{
   int arr[]={400,70,90,50,10};
   int size,max,smax;
   size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
   printf("\n");
   smax=max=INT_MIN;
   for(int i=0;i<size;i++){
      if(arr[i]>max){
         smax=max;
         max=arr[i];
      }
      else if(arr[i]>smax && arr[i]<max){
         smax=arr[i];
      }
   }
   printf("Largest element : %d \n",max);
   printf("Second Largest element : %d ",smax);
   return 0;
}
```

Output-

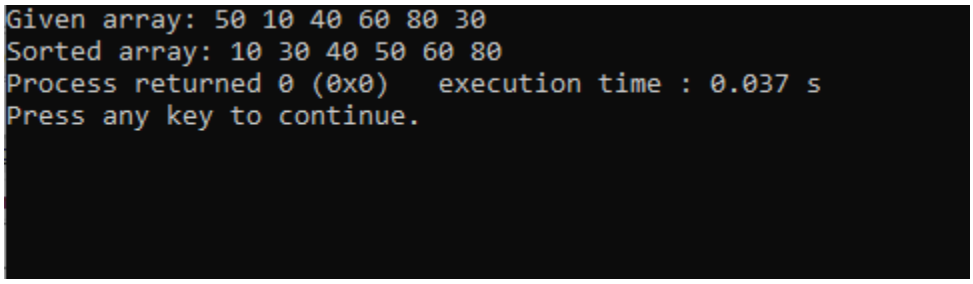


7. Write a program to implement bubble sort.

```
#include<stdio.h>
int main()
{
   int arr []={50,10,40,60,80,30};
   int size,temp;
   size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
   printf("\n");
   for(int i=0;i<size-1;i++){
      for(int 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;
         }
      }
   }
   printf("Sorted array: ");
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
```

```
    return 0;
}
```

Output-

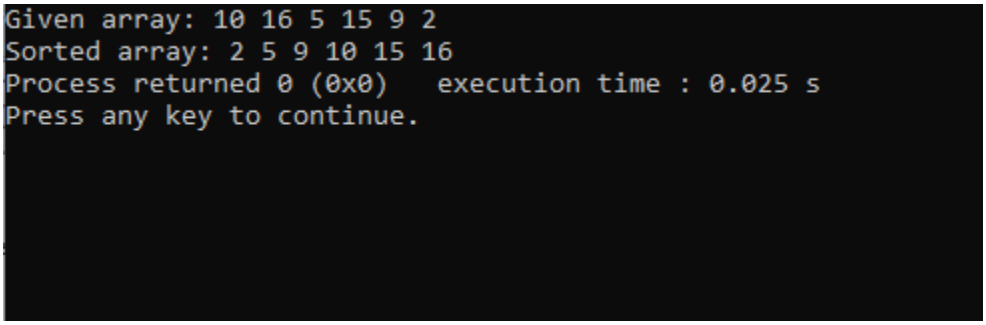


8.Write a program to implement insertion sort.

```
#include<stdio.h>
int main(){
    int arr[]={10,16,5,15,9,2};
    int size,j,temp;
    size=sizeof(arr)/sizeof(arr[0]);
    printf("Given array: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
    for(int i=1;i<size;i++){
        temp=arr[i];
        j=i-1;
        while(j>=0 && arr[j]>temp){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=temp;
    }
    printf("Sorted array: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    return 0;
```

}

Output-

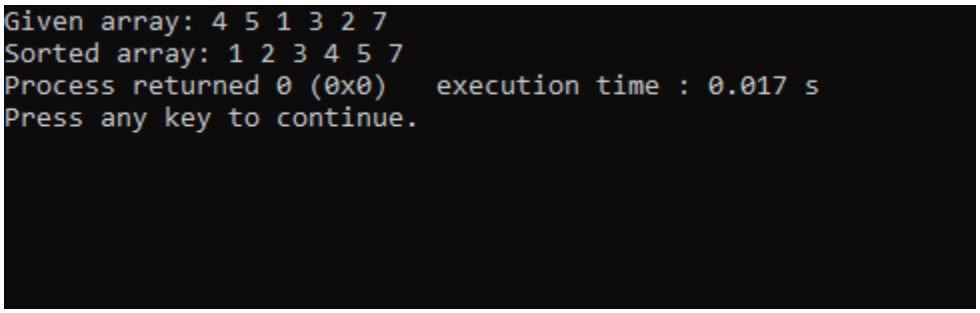

Given array: 10 16 5 15 9 2
Sorted array: 2 5 9 10 15 16
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.

9.Write a program to implement selection sort.

```
#include <stdio.h>
int main()
{
   int arr[]={4,5,1,3,2,7};
   int size,min,temp;
   size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
   }
   printf("\n");
   for(int i=0;i<size-1;i++){
      min=i;
      for(int j=i+1;j<size;j++){    //finding min element in the unsorted array
         if(arr[j]<arr[min]){
            min=j;
         }
      }
      //swapping the min element at beginning
      if(min!=i){
        temp=arr[i];
        arr[i]=arr[min];
        arr[min]=temp;
      }
```

```
    }
    printf("Sorted array: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    return 0;
}
```

Output-

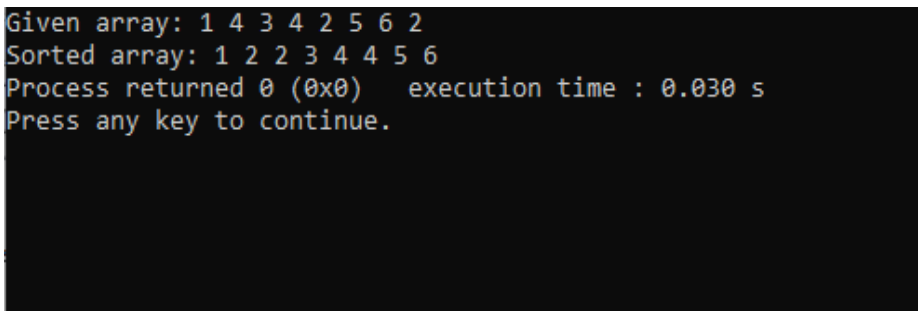


10. Write a program to implement counting sort.

```
#include<stdio.h>
int main(){
  int arr[]={1,4,3,4,2,5,6,2};
  int size,max;
  max=arr[0];
  size=sizeof(arr)/sizeof(arr[0]);
  printf("Given array: ");
  for(int i=0;i<size;i++){
    printf("%d ",arr[i]);
  }
  printf("\n");
  //Finding maximum element in the array
  for(int i=1;i<size;i++){
    if(arr[i]>max){
        max=arr[i];
    }
  }
  int count[max+1];
  int res [size];
  for(int i=0;i<=max;i++){
    count[i]=0;  //Initializing count array from 0
  }
  //Maintain the count of every element at its respective index
  for(int i=0;i<size;i++){
```

```
  count[arr[i]]++;
 }
 for(int i=1;i<=max;i++){
  count[i]+=count[i-1];
 }
 for(int i=size-1;i>=0;i--){
  res[count [arr[i]]-1]=arr[i];
  count[arr[i]]--;
 }
 for(int i=0;i<size;i++){
  arr[i]=res[i];
 }
 printf("Sorted array: ");
 for(int i=0;i<size;i++){
  printf("%d ",arr[i]);
 }
 return 0;
}
```

Output-



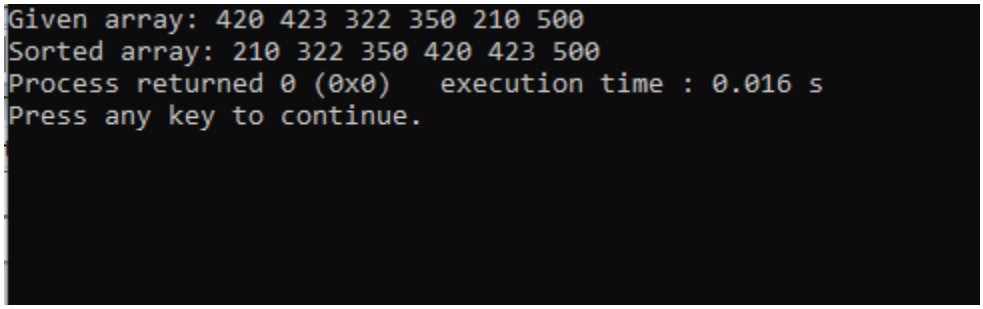
11. Write a program to implement Radix Sort.

```
#include<stdio.h>
int getMax(int arr[],int size){
    int max=arr[0];
    for(int i=1;i<size;i++){
        if(arr[i]>max){
            max=arr[i];
        }
    }
    return max;
}
void countingSort(int arr[],int size,int pos){
    int out [size+1];
    int count [10]={0};
    for(int i=0;i<size;i++){
        count[(arr[i]/pos)%10]++;
    }
    for(int i=1;i<10;i++){
        count[i]=count[i]+count[i-1];
    }
    for(int i=size-1;i>=0;i--){
        out[count[(arr[i]/pos)%10]-1] =arr[i];
        count[(arr[i]/pos)%10]--;
    }
```

```
    for(int i=0;i<size;i++){
        arr[i]=out[i];
    }
}
void printArray(int arr[],int size){
    printf("Sorted array: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
}
int main(){
    int arr []={420,423,322,350,210,500};
    int size,max;
    size=sizeof(arr)/sizeof(arr[0]);
    printf("Given array: ");
    for(int i=0;i<size;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
    max=getMax(arr,size);
    for(int pos=1;max/pos>0;pos*=10){
        countingSort(arr,size,pos);
    }
    printArray(arr,size);
    return 0;
```

}

Output-


Given array: 420 423 322 350 210 500
Sorted array: 210 322 350 420 423 500
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.

12. Write a program to menu-driven program which includes- Addition , Substraction and Multiplication of two matrices.

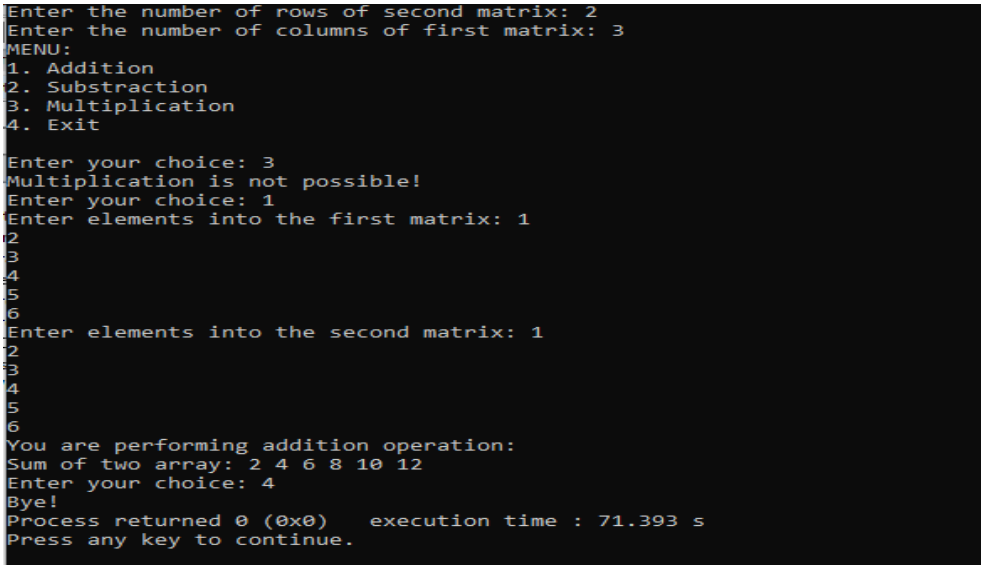
```
#include<stdio.h>
void insertElements(int arr1[50][50],int arr2[50][50],int row1,int col1,int row2,int col2){
   printf("Enter elements into the first matrix: ");
   for(int i=0;i<row1;i++){
       for(int j=0;j<col1;j++){
           scanf("%d",&arr1[i][j]);
       }
   }
   printf("Enter elements into the second matrix: ");
   for(int i=0;i<row2;i++){
       for(int j=0;j<col2;j++){
           scanf("%d",&arr2[i][j]);
       }
    }
}
void addition(int arr1[50][50],int arr2[50][50],int row,int col){
    printf("You are performing addition operation: \n");
    int sum[row][col];
    for(int i=0;i<row;i++){
       for(int j=0;j<col;j++){
           sum[i][j]=arr1[i][j]+arr2[i][j];
       }
    }
```

```
    printf("Sum of two array: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            printf("%d ",sum[i][j]);
        }
    }
}
void substraction(int arr1[50][50],int arr2[50][50],int row,int col){
    printf("You are performing substraction operation: \n");
    int sum[row][col];
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            sum[i][j]=arr1[i][j]-arr2[i][j];
        }
    }
    printf("Substraction of two array: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            printf("%d ",sum[i][j]);
        }
    }
}
void multiplication (int arr1[50][50],int arr2[50][50],int row1,int col1,int row2,int col2){
   printf("You are performing multiplication operation: \n");
   int mul[row1][col2],sum=0;
```

```
   for(int i=0;i<row1;i++){
      for(int j=0;j<col2;j++){
          for(int k=0;k<=col2;k++){
              sum=sum+(arr1[i][k]*arr2[k][j]);
          }
          mul[i][j]=sum;
          sum=0;
      }
      sum=0;
   }
    printf("Multiplication of two matrix: ");
    for(int i=0;i<row1;i++){
        for(int j=0;j<col2;j++){
            printf("%d ",mul[i][j]);
        }
    }
}
int main(){
    int arr1[50][50];
    int arr2[50][50];
    int row1,row2,col1,col2;
    printf("Enter the number of rows of first matrix: ");
    scanf("%d",&row1);
    printf("Enter the number of columns of first matrix: ");
    scanf("%d",&col1);
```

```
printf("Enter the number of rows of second matrix: ");
scanf("%d",&row2);
printf("Enter the number of columns of first matrix: ");
scanf("%d",&col2);
printf("MENU:\n1. Addition\n2. Substraction\n3. Multiplication\n4. Exit\n");
int flag=1;
int choice;
while(flag){
   printf("\nEnter your choice: ");
   scanf("%d",&choice);
   switch (choice){
     case 1:
        if(row1==row2 && col1==col2){
          insertElements(arr1,arr2,row1,col1,row2,col2);
          addition(arr1,arr2,row1,col1);
        }
        else{
          printf("Addition is not possible");
        }
        break;
     case 2:
        if(row1==row2 && col1==col2){
          insertElements(arr1,arr2,row1,col1,row2,col2);
          substraction(arr1,arr2,row1,col1);
        }
```

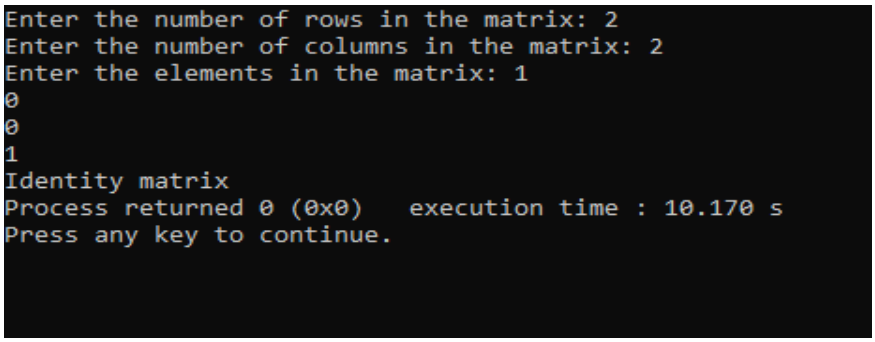
```
            else{
              printf("Substraction is not possible");
            }
            break;
        case 3:
            if(col1==row2){
              insertElements(arr1,arr2,row1,col1,row2,col2);
              multiplication(arr1,arr2,row1,col1,row2,col2);
            }
            else{
              printf("Multiplication is not possible! ");
            }
            break;
        case 4:
            printf("Bye!");
            flag=0;
            break;
        default:
            printf("Invalid input.....");
            flag=0;
            break;
      }
  }
  return 0;
}
```


Enter the number of rows of second matrix: 2
Enter the number of columns of first matrix: 3
MENU:
1. Addition
2. Substraction
3. Multiplication
4. Exit
Enter your choice: 3
Multiplication is not possible!
Enter your choice: 1
Enter elements into the first matrix: 1
2
3
4
5
6
Enter elements into the second matrix: 1
2
3
4
5
6
You are performing addition operation:
Sum of two array: 2 4 6 8 10 12
Enter your choice: 4
Bye!
Process returned 0 (0x0) execution time : 71.393 s
Press any key to continue.

13. Write a program to check whether the matrix is identity or not.

```
#include<stdio.h>
int main(){
   int arr[50][50];
   int row,col,flag=0;
   printf("Enter the number of rows in the matrix: ");
   scanf("%d",&row);
   printf("Enter the number of columns in the matrix: ");
   scanf("%d",&col);
   printf("Enter the elements in the matrix: ");
   for(int i=0;i<row;i++){
       for(int j=0;j<col;j++){
           scanf("%d",&arr[i][j]);
       }
   }
   if(row==col){
     for(int i=0;i<row;i++){
       for(int j=0;j<col;j++){
           if(i==j && arr[i][j]!=1){
               flag=1;
               break;
           }
           else if(i!=j && arr[i][j]!=0){
               flag=1;
               break;
```

```
        }
      }
    }
  }
  else{
      flag=1;
  }

  if(flag==0){
      printf("Identity matrix");
  }
  else{
      printf("Not an identity matrix");
  }
  return 0;
}
```



14. Write a program to check whether the two matrices are identical or not.

```
#include<stdio.h>
int main(){
    int arr1[50][50],arr2[50][50];
    int row1,row2,col1,col2,flag=0;
    printf("Enter the number of rows for first matrix: ");
    scanf("%d",&row1);
    printf("Enter the number of columns for first matrix: ");
    scanf("%d",&col1);
    printf("Enter the elements in the first matrix: ");
    for(int i=0;i<row1;i++){
        for(int j=0;j<col1;j++){
            scanf("%d",&arr1[i][j]);
        }
    }
    printf("Enter the number of rows for second matrix: ");
    scanf("%d",&row2);
    printf("Enter the number of columns for second matrix: ");
    scanf("%d",&col2);
    printf("Enter the elements in the second matrix: ");
    for(int i=0;i<row2;i++){
        for(int j=0;j<col2;j++){
            scanf("%d",&arr2[i][j]);
        }
    }
```

```
    if(row1==row2 && col1==col2){
        for(int i=0;i<row1;i++){
            for(int j=0;j<col1;j++){
                if(arr1[i][j]!=arr2[i][j]){
                    flag=1;
                }
            }
        }
    }
    else{
        flag=1;
    }
    if(flag==0){
        printf("Matrix is identical");
    }
    else{
        printf("Not an identical matrix");
    }
    return 0;
}
```

```
Enter the number of rows for first matrix: 2
Enter the number of columns for second matrix: 2
Enter the elements in the second matrix: 2
2
3
4
Enter the number of rows for second matrix: 2
Enter the number of columns for second matrix: 2
Enter the elements in the second matrix: 2
2
3
4
Matrix is identical
Process returned 0 (0x0)   execution time : 24.574 s
Press any key to continue.
```

15. Write a program to print all diagonal element.

```
#include<stdio.h>
int main(){
    int arr[50][50];
    int row,col;
    printf("Enter the number of rows: ");
    scanf("%d",&row);
    printf("Enter the number of columns: ");
    scanf("%d",&col);
    printf("Enter the elements in array: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            scanf("%d",&arr[i][j]);
        }
    }
    printf("Diagonal elements of the array: ");
    if(row==col){
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){
                if(i==j){
                    printf("%d ",arr[i][j]);
                }
            }
        }
    }
```

```
    else{
        printf("Can't find diagonal elements: ");
    }
    return 0;
}
```

```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements in array: 2
4
6
7
Diagonal elements of the array: 2 7
Process returned 0 (0x0)   execution time : 9.819 s
Press any key to continue.
```

16. Write a program to print sum of all rows of a matrix.

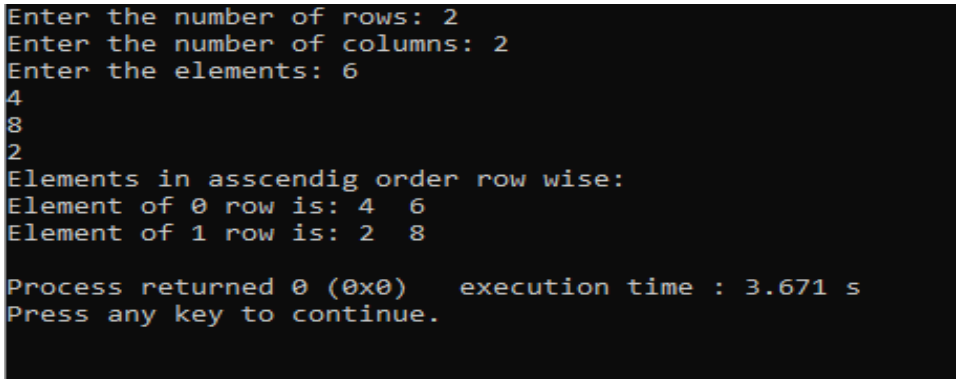
```
#include<stdio.h>
int main(){
   int arr[50][50];
   int row,col,sum=0;
   printf("Enter the number of rows: ");
   scanf("%d",&row);
   printf("Enter the number of columns: ");
   scanf("%d",&col);
   printf("Enter the elements in array: ");
   for(int i=0;i<row;i++){
      for(int j=0;j<col;j++){
         scanf("%d",&arr[i][j]);
      }
   }
   for(int i=0;i<row;i++){
      sum=0;
      for(int j=0;j<col;j++){
         sum+=arr[i][j];
      }
      printf("Sum of %d row is %d \n",i,sum);
   }
   return 0;
}
```

```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements in array: 1
2
3
4
Sum of 0 row is 3
Sum of 1 row is 7

Process returned 0 (0x0)   execution time : 5.482 s
Press any key to continue.
```

17. Write a program to arrange all rows of matrix in ascending order.

```
#include<stdio.h>
int main(){
    int arr[50][50];
    int row,col,temp;
    printf("Enter the number of rows: ");
    scanf("%d",&row);
    printf("Enter the number of columns: ");
    scanf("%d",&col);
    printf("Enter the elements: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            scanf("%d",&arr[i][j]);
        }
    }
    printf("Elements in asscendig order row wise: \n");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            for(int k=j+1;k<col;k++){
                if(arr[i][j]>arr[i][k]){
                    temp=arr[i][j];
                    arr[i][j]=arr[i][k];
                    arr[i][k]=temp;
                }
            }
```

```
        }
    }
    for(int i=0;i<row;i++){
        printf("Element of %d row is: ",i);
        for(int j=0;j<col;j++){
            printf("%d  ",arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



18. Write a program to check whether a matrix is sparse or not.

```
#include<stdio.h>
int main(){
    int arr[50][50];
    int row,col;
    printf("Enter the number of rows: ");
    scanf("%d",&row);
    printf("Enter the number of columns: ");
    scanf("%d",&col);
    int size=row*col,count=0;
    printf("Enter the element in the matrix: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            scanf("%d",&arr[i][j]);
        }
    }
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            if(arr[i][j]==0){
                count++;
            }
        }
    }
    if(count>size/2){
        printf("Sparse matrix");
```

```
    }
    else{
        printf("Not a sparse matrix");
    }
    return 0;
}
```

```
Enter the number of rows: 2
Enter the number of columns: 3
Enter the element in the matrix: 5
0
0
0
0
4
Sparse matrix
Process returned 0 (0x0)   execution time : 8.080 s
Press any key to continue.
```

19. Write a program to convert sparse matrix into row triplet form.

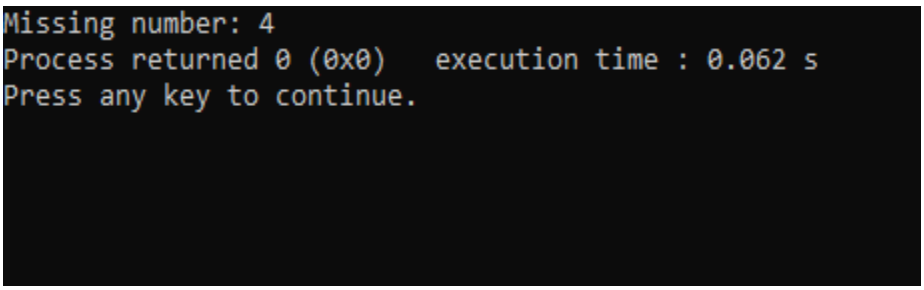
```
#include<stdio.h>
int main(){
    int arr[50][50];
    int row,col,size=0;
    printf("Enter the number of rows: ");
    scanf("%d",&row);
    printf("Enter the number of columns: ");
    scanf("%d",&col);
    printf("Enter the element in the matrix: ");
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            scanf("%d",&arr[i][j]);
        }
    }
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            if(arr[i][j]!=0){
                size++;
            }
        }
    }
    int new_matrix[3][size];
    int k=0;
    for(int i=0;i<3;i++){
```

```
        for(int j=0;j<size;j++){
            if(arr[i][j]!=0){
                new_matrix[0][k]=i;
                new_matrix[1][k]=j;
                new_matrix[2][k]=arr[i][j];
                k++;
            }
        }
    }
    printf("Row triplet of a matrix: ");
    for(int i=0;i<3;i++){
        for(int j=0;j<size;j++){
            printf("%d  ",new_matrix[i][j]);
        }
    }
    return 0;
}
```

```
Enter the number of rows: 3
Enter the number of columns: 2
Enter the element in the matrix: 5
0
0
0
0
9
Row triplet of a matrix: 0  2  0  1  5  9
Process returned 0 (0x0)   execution time : 7.816 s
Press any key to continue.
```

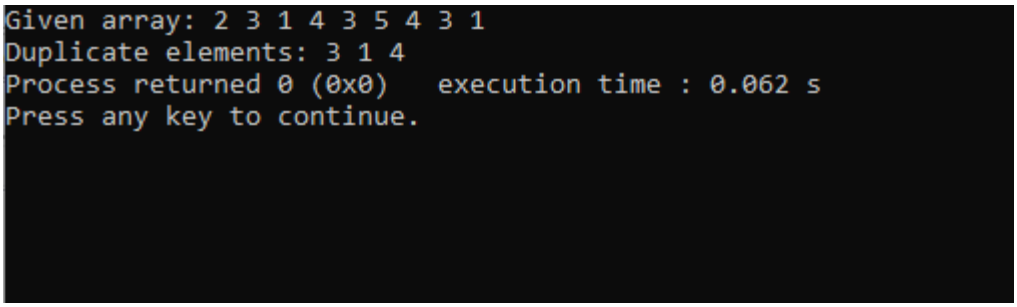
20. Write a program to find missing number in an array.

```
#include<stdio.h>
int main(){
   int arr[]={1,2,3,6,5};
   int size=sizeof(arr)/sizeof(arr[0]);
   int sum=0,natsum=0,res;
   for(int i=0;i<size;i++){
       sum+=arr[i];
   }
   for(int i=1;i<=size+1;i++){
       natsum+=i;
   }
   res=natsum-sum;
   printf("Missing number: %d",res);
   return 0;
}
```



21. Write a program to print all duplicate elements.

```
#include<stdio.h>
int main(){
   int arr[]={2,3,1,4,3,5,4,3,1},count;
   int size=sizeof(arr)/sizeof(arr[0]);
   printf("Given array: ");
   for(int i=0;i<size;i++){
       printf("%d ",arr[i]);
   }
   printf("\nRepeated elements: ");
   for(int i=0;i<size;i++){
       count=1;
       for(int j=i+1;j<size;j++){
          if(arr[i]==arr[j]){
              count++;
              arr[j]=-1;
          }
       }
       if(count>1 && arr[i]!=-1){
          printf("%d ",arr[i]);
       }
   }
   return 0;
}
```


Given array: 2 3 1 4 3 5 4 3 1
Duplicate elements: 3 1 4
Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.

22. Write a program to remove duplicate element from an array.

```
#include<stdio.h>
int main(){
  int arr[]={1,2,3,1,4,3,3,5};
  int size=sizeof(arr)/sizeof(arr[0]);
  printf("Given array: ");
  for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
  }
  for(int i=0;i<size;i++){
      for(int j=i+1;j<size;j++){
          if(arr[i]==arr[j]){
              for(int k=j;k<size-1;k++){
                 arr[k]=arr[k+1];
              }
              size--;
              j--;
          }
      }
  }
  printf("\nNew array: ");
  for(int i=0;i<size;i++){
      printf("%d ",arr[i]);
  }
  return 0;
```

}

```
Given array: 1 2 3 1 4 3 3 5
New array: 1 2 3 4 5
Process returned 0 (0x0)   execution time : 0.068 s
Press any key to continue.
```

23. Write a program to create a dynamic array and find sum of numbers entered using malloc(),calloc(),realloc() and free().

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

void malloc_fun(int n){

   int *p,sum=0;

   p=(int *)malloc(n*sizeof(int));

   printf("\nEnter the elements: ");

   for(int i=0;i<n;i++){

      scanf("%d",p+i);

   }

   for(int i=0;i<n;i++){

      sum+=*(p+i);

   }

   free(p);

   printf("\nSum of elements: %d\n",sum);

}

void calloc_fun(int n){

   int *p,sum=0;

   p=(int *)calloc(n,sizeof(int));

   printf("\nEnter the elements: ");

   for(int i=0;i<n;i++){

      scanf("%d",p+i);

   }

   for(int i=0;i<n;i++){

      sum+=*(p+i);
```

```
    }
    free(p);
    printf("Sum of elements: %d\n",sum);
}
void realloc_fun(int n){
    int *p,sum=0,size,i;
    p=(int *)malloc(n*sizeof(int));
    printf("\nEnter the elements: ");
    for(i=0;i<n;i++){
        scanf("%d",p+i);
    }
    for(int j=0;j<n;j++){
        sum+=*(p+j);
    }
    printf("Sum of elements before reallocation: %d",sum);
    printf("\nEnter the size you want to increase: ");
    scanf("%d",&size);
    p=(int *)realloc(p,size);
    printf("\nEnter new elements: ");
    for(int j=i;j<n+size;j++){
        scanf("%d",p+j);
    }
    printf("\nAll Elements: ");
    for(int j=0;j<n+size;j++){
        printf("%d ",*(p+j));
```

```
    }
    for(int j=i;j<n+size;j++){
        sum+=*(p+j);
    }
    free(p);
    printf("\nSum of elements after reallocation: %d\n",sum);
}
int main(){
    int n;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("MENU:\n1. Addition using malloc() \n2. Addition using calloc() \n3. Addition using realloc() \n4. Exit\n");
    int flag=1;
    int choice;
    while(flag){
        printf("\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                malloc_fun(n);
                break;
            case 2:
                calloc_fun(n);
                break;
```

```
            case 3:
                realloc_fun(n);
                break;
            case 4:
                printf("Bye!");
                flag=0;
                break;
            default:
                printf("Invalid input.....");
                break;
        }
    }
    return 0;
}
```

```
Enter the number of elements: 3
MENU:
1. Addition using malloc()
2. Addition using calloc()
3. Addition using realloc()
4. Exit

Enter your choice:1

Enter the elements: 10 20 30

Sum of elements: 60

Enter your choice:2

Enter the elements: 20 40 60
Sum of elements: 120

Enter your choice:3

Enter the elements: 40 50 10
Sum of elements before reallocation: 100
Enter the size you want to increase: 2

Enter new elements: 60 70

All Elements: 40 50 10 60 70
Sum of elements after reallocation: 230
```

24. Write a program to create an array of structure of 5 employee with following infor: eid(int),ename(char[]),esal(int). Find the employee with highest salary and display all its details.

```
#include<stdio.h>
struct employee{
    int id;
    char name[20];
    int sal;
};
int main(){
    struct employee emp[5];
    int count=0,max;
    for(int i=0;i<5;i++){
        printf("\nEnter the id,name and salary of %d employee: ",i+1);
        scanf("%d%s%d",&emp[i].id,&emp[i].name,&emp[i].sal);
    }
    max=emp[0].sal;
    for(int i=1;i<5;i++){
        if(max<emp[i].sal){
            max=emp[i].sal;
            count=i;
        }
    }
    printf("Details of Highest Paying Employee: \n");
    printf("Id: %d\n",emp[count].id);
```

```
    printf("Name: %s\n",emp[count].name);
    printf("Salary: %d",emp[count].sal);
    return 0;
}
```

```
Enter the id,name and salary of 1 employee: 101 sujit 2000

Enter the id,name and salary of 2 employee: 102 xyz 3000

Enter the id,name and salary of 3 employee: 103 tsd 5000

Enter the id,name and salary of 4 employee: 104 ijk 6000

Enter the id,name and salary of 5 employee: 105 ojf 2500
Details of Highest Paying Employee:
Id: 104
Name: ijk
Salary: 6000
Process returned 0 (0x0)   execution time : 72.559 s
Press any key to continue.
```

25. Write aprogram to perform following operation on Singly Linked List:-

a. Create
b.Traverse
c.Insertion
d. Deletion

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
struct node *begin_insert(struct node * head){
   struct node *new,*temp;
   new=(struct node*)malloc(sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&new->data);
   new->next=NULL;
   if(head==NULL){
      head=new;
      temp=new;
   }
   else{
      new->next=head;
      head=new;
   }
```

```
    return new;
}
void last_insert(struct node* head){
    struct node*new,*temp;
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    if(head==NULL){
        head=new;
        temp=new;
    }
    else{
        temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=new;
    }
}
void random_insert(struct node *head){
    struct node *new,*temp;
    int pos,i=1;
    printf("Enter position: ");
    scanf("%d",&pos);
```

```
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    temp=head;
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    new->next=temp->next;
    temp->next=new;
}
struct node *begin_delete(struct node *head){
    struct node *temp;
    temp=head;
    head=temp->next;
    free(temp);
    return head;
}
struct node *last_delete(struct node *head){
    struct node *temp,*prev;
    temp=head;
    while(temp->next!=NULL){
        prev=temp;
        temp=prev->next;
```

```
    }
    prev->next=NULL;
    free(temp);
    return head;
}
void random_delete(struct node *head){
    struct node *temp,*temp1;
    int i=1,pos;
    printf("Enter Position: ");
    scanf("%d",&pos);
    temp=head;
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    temp1=temp->next;
    temp->next=temp1->next;
    free(temp1);
}
void display(struct node *head){
    printf("\nNew Linked list: ");
    struct node *temp;
    temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
```

```
        temp=temp->next;
    }
}
int main(){
    int choice,flag=1;
    struct node*head=NULL,*second=NULL,*third=NULL,*temp=NULL;
    head = (struct Node*)malloc(sizeof(struct node));
    second = (struct Node*)malloc(sizeof(struct node));
    third = (struct Node*)malloc(sizeof(struct node));
    //Insert initial data
    head->data=10;
    second->data=20;
    third->data=30;
    head->next=second;
    second->next=third;
    third->next=NULL;
    printf("Initial linked list: ");
    printf("%d  %d  %d \n",head->data,second->data,third->data);
    printf("\nChoose one option from following: ");
    printf("\n1.Insert in beginning\n2.Insert at last\n3.Insert at any random position\n4.Delete
from beginning\n5.Delete from last\n6.Delete from specific position\n7.Exit");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
```

```
case 1:
    head=begin_insert(head);
    display(head);
    break;
case 2:
    last_insert(head);
    display(head);
    break;
case 3:
    random_insert(head);
    display(head);
    break;
case 4:
    head=begin_delete(head);
    display(head);
    break;
case 5:
    head=last_delete(head);
    display(head);
    break;
case 6:
    random_delete(head);
    display(head);
    break;
case 7:
```

```
            printf("\nBye!");
            flag=0;
            break;
        default:
            printf("Invalid input....");
            break;
    }
  }
  free(head);
  free(second);
  free(third);
  return 0;
}
```

```
Initial linked list: 10  20  30

Choose one option from following:
1.Insert in beginning
2.Insert at last
3.Insert at any random position
4.Delete from beginning
5.Delete from last
6.Delete from specific position
7.Exit
Enter your choice: 1
Enter data: 40

New Linked list: 40 10 20 30
Enter your choice: 2
Enter data: 50

New Linked list: 40 10 20 30 50
Enter your choice: 3
Enter position: 2
Enter data: 60

New Linked list: 40 60 10 20 30 50
Enter your choice: 4

New Linked list: 60 10 20 30 50
Enter your choice: 5

New Linked list: 60 10 20 30
Enter your choice: 6
Enter Position: 3

New Linked list: 60 10 30
Enter your choice: 7

Bye!
Process returned -1073740940 (0xC0000374)   execution time : 64.098 s
```

e. Find the length of the linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
int main(){
   struct node *new,*start=NULL,*temp=NULL;
   int choice=1,count=0;
   while(choice){
      new=(struct node *)malloc(sizeof(struct node));
      printf("Enter data: ");
      scanf("%d",&new->data);
      new->next=NULL;
      if(start==NULL){
         start=new;
         temp=start;
      }
      else{
         temp->next=new;
         temp=new;
      }
      printf("Do yu want to add more? If yes Enter 1 otherwise enter any number: ");
      scanf("%d",&choice);
```

```
    }
    //printing linked list
    printf("\nLinked list element: ");
    temp=start;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
        count++;
    }
    printf("\nNumber of elements in the linked list is: %d",count);
    free(start);
    free(new);
    free(temp);
    return 0;
}
```

```
Enter data: 10
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 20
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 30
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 40
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 50
Do yu want to add more? If yes Enter 1 otherwise enter any number: 0

Linked list element: 10 20 30 40 50
Number of elements in the linked list is: 5
Process returned 0 (0x0)   execution time : 16.068 s
Press any key to continue.
```

f. Search an element in the Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
int main(){
   struct node *new,*start=NULL,*temp=NULL;
   int choice=1,ser,flag=0;
   while(choice){
      new=(struct node *)malloc(sizeof(struct node));
      printf("Enter data: ");
      scanf("%d",&new->data);
      new->next=NULL;
      if(start==NULL){
         start=new;
         temp=start;
      }
      else{
         temp->next=new;
         temp=new;
      }
      printf("Do yu want to add more? If yes Enter 1 otherwise enter any number: ");
      scanf("%d",&choice);
```

```
}
//printing linked list
printf("\nLinked list element: ");
temp=start;
while(temp!=NULL){
   printf("%d ",temp->data);
   temp=temp->next;
}
//search element
printf("\nEnter the element you want to search: ");
scanf("%d",&ser);
temp=start;
while(temp!=NULL){
   if(ser==temp->data){
      flag=1;
      break;
   }
   temp=temp->next;
}
if(flag==1){
   printf("Element is present");
}
else{
   printf("Such element doesn't exist");
}
```

```
    free(start);
    free(new);
    free(temp);
    return 0;
}
```

```
Enter data: 10
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 20
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 30
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 40
Do yu want to add more? If yes Enter 1 otherwise enter any number: 1
Enter data: 20
Do yu want to add more? If yes Enter 1 otherwise enter any number: 0

Linked list element: 10 20 30 40 20
Enter the element you want to search: 40
Element is present
Process returned 0 (0x0)   execution time : 17.456 s
Press any key to continue.
```

g. Sort a Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
int main(){
    struct node *new,*temp=NULL,*start=NULL;
    int choice=1;
    while(choice==1){
        new=(struct node *)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d",&new->data);
        new->next=NULL;
        if(start==NULL){
            start=new;
            temp=new;
        }
        else{
            temp->next=new;
            temp=new;
        }
        printf("\nDo you want to add more? Enter 1 else enter anything: ");
        scanf("%d",&choice);
```

```
    }
    printf("\nPrinting linked list: ");
    temp=start;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
struct node *i=start,*j=start;
    while(i!=NULL){
        j=start;
        while(j!=NULL){
            if(j->data>i->data){
                int local=j->data;
                j->data=i->data;
                i->data=local;
            }
            j=j->next;
        }
        i=i->next;
    }
    printf("\nPrinting sorted linked list: ");
    temp=start;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
```

```
    }
    free(start);
    free(temp);
    free(new);
    free(i);
    free(j);
    return 0;
}
```

```
Enter data: 30

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 50

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 40

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 10

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 60

Do you want to add more? Enter 1 else enter anything: 0

Printing linked list: 30 50 40 10 60
Printing sorted linked list: 10 30 40 50 60
Process returned 0 (0x0)   execution time : 19.537 s
Press any key to continue.
```

h. Add a node in sorted Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
int main(){
   struct node *new,*temp=NULL,*start=NULL,*new_ele=NULL;
   int choice=1;
   printf("Entered data must be in sorted order: \n");
   while(choice==1){
      new=(struct node *)malloc (sizeof(struct node));
      printf("Enter data: ");
      scanf("%d",&new->data);
      new->next=NULL;
      if(start==NULL){
         start=new;
         temp=new;
      }
      else{
         temp->next=new;
         temp=new;
      }
      printf("\nDo you want to add more? Enter 1 else enter anything: ");
```

```
    scanf("%d",&choice);
}
printf("\nPrinting linked list: ");
temp=start;
while(temp!=NULL)
    {
    printf("%d ",temp->data);
    temp=temp->next;
}
printf("\nEnter the element you want to insert: ");
new_ele=(struct node *)malloc(sizeof(struct node));
scanf("%d",&new_ele->data);
new_ele->next=NULL;
temp=start;
while(temp!=NULL){
    //insert at beginning
    if(new_ele->data < temp->data){
        new_ele->next=temp;
        start=new_ele;
        temp=start;
        break;
    }
    //insert in between
    else if(new_ele->data < temp->next->data){
        new_ele->next = temp->next;
```

```
            temp->next=new_ele;
            break;
        }
        //insert at last
        if(temp->next->next==NULL){
            temp->next->next=new_ele;
            new_ele->next=NULL;
            break;
        }
        temp=temp->next;
    }
    printf("\nPrinting linked list after insertion : ");
    temp=start;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    free(start);
    free(new);
    free(temp);
    free(new_ele);
    return 0;
}
```

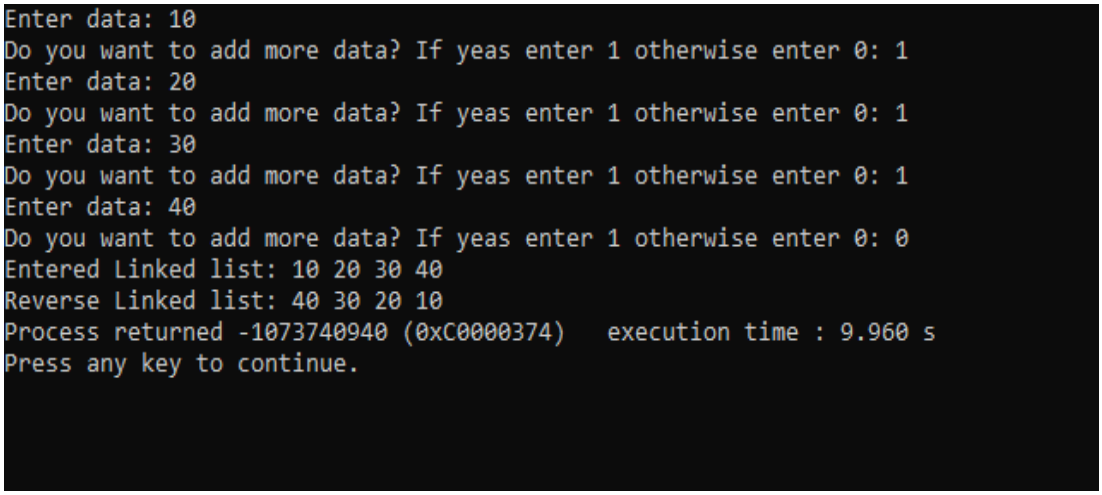
```
Entered data must be in sorted order:
Enter data: 10

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 20

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 30

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 40

Do you want to add more? Enter 1 else enter anything: 1
Enter data: 50

Do you want to add more? Enter 1 else enter anything: 0

Printing linked list: 10 20 30 40 50
Enter the element you want to insert: 35

Printing linked list after insertion : 10 20 30 35 40 50
Process returned 0 (0x0)   execution time : 11.388 s
Press any key to continue.
```

i. Reverse a Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *reverse(struct node *start){
    struct node *curr=NULL,*next_node=NULL,*prev=NULL;
    curr=start;
    next_node=start;
    while(next_node!=NULL){
        next_node=next_node->next;
        curr->next=prev;
        prev=curr;
        curr=next_node;
    }
    start=prev;
    return start;
}
void display(struct node *start){
    printf("\nReverse Linked list: ");
    struct node *temp;
    temp=start;
    while(temp!=NULL){
```

```
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
int main(){
    struct node *new,*start=NULL,*temp=NULL;
    int choice=1;
    while(choice==1){
        new=(struct node *)malloc (sizeof(struct node));
        printf("Enter data: ");
        scanf("%d",&new->data);
        new->next=NULL;
        if(start==NULL){
            start=new;
            temp=new;
        }
        else{
            temp->next=new;
            temp=new;
        }
        printf("Do you want to add more data? If yeas enter 1 otherwise enter 0: ");
        scanf("%d",&choice);
    }
    printf("Entered Linked list: ");
    temp=start;
```

```
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    start=reverse(start);
    display(start);
    free(start);
    free(new);
    free(temp);
    return 0;
}
```



j. Detect loop in Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
void floydcycle(struct node *start){
   int flag=0;
   if(start==NULL){
      printf("No value");
   }
   struct node *slow=start,*fast=start;
   while(slow!=NULL && fast!=NULL){
      fast=fast->next;
      if(fast!=NULL){
         fast=fast->next;
      }
      slow=slow->next;
      if(slow==fast){
         printf("loop  is present");
         flag=1;
         break;
      }
   }
```

```
    if(flag==0){
        printf("No loop");
    }
    free(slow);
    free(fast);
}
int main(){
    struct node *new,*start=NULL,*temp=NULL;
    int choice=1;
    while(choice==1){
        new=(struct node *)malloc (sizeof(struct node));
        printf("Enter data: ");
        scanf("%d",&new->data);
        new->next=NULL;
        if(start==NULL){
            start=new;
            temp=new;
        }
        else{
            temp->next=new;
            temp=new;
        }
        printf("Do you want to add more data? If yeas enter 1 otherwise enter 0: ");
        scanf("%d",&choice);
    }
```

```
    temp->next=start->next;
    floydcycle(start);
    free(start);
    free(temp);
    free(new);
    return 0;
}
```

```
Enter data: 10
Do you want to add more data? If yeas enter 1 otherwise enter 0: 1
Enter data: 20
Do you want to add more data? If yeas enter 1 otherwise enter 0: 1
Enter data: 30
Do you want to add more data? If yeas enter 1 otherwise enter 0: 1
Enter data: 40
Do you want to add more data? If yeas enter 1 otherwise enter 0: 0
loop  is present
Process returned -1073740940 (0xC0000374)   execution time : 9.251 s
Press any key to continue.
```

26.Write a menu-driven program to perform following operations on Doubly Linked List: create,traverse,insert.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *begin_insert(struct node * head){
   struct node *new,*temp;
   new=(struct node*)malloc(sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&new->data);
   new->next=NULL;
   new->prev=NULL;
   if(head==NULL){
      head=new;
      temp=new;
   }
   else{
      new->next=head;
      head->prev=new;
      head=new;
   }
```

```
    return new;
}
void last_insert(struct node* head){
    struct node*new,*temp;
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    new->prev=NULL;
    if(head==NULL){
        head=new;
        temp=new;
    }
    else{
        temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=new;
        new->prev=temp;
        temp=new;
    }
}
void random_insert(struct node *head){
    struct node *new,*temp;
```

```
    int pos,i=1;
    printf("Enter position: ");
    scanf("%d",&pos);
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    new->prev=NULL;
    temp=head;
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    new->prev=temp;
    new->next=temp->next;
    temp->next->prev=new;
    temp->next=new;
}
void display(struct node *head){
    printf("\nNew Doubly Linked list: ");
    struct node *temp;
    temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
```

```
    }
}
int main(){
    int choice,flag=1;
    struct node*head=NULL,*second=NULL,*third=NULL,*temp=NULL;
    head = (struct Node*)malloc(sizeof(struct node));
    second = (struct Node*)malloc(sizeof(struct node));
    third = (struct Node*)malloc(sizeof(struct node));
    //Insert initial data
    head->data=10;
    second->data=20;
    third->data=30;
    head->prev=NULL;
    head->next=second;
    second->prev=head;
    second->next=third;
    third->prev=second;
    third->next=NULL;
    printf("Initial Doubly Linked List: ");
    printf("%d  %d  %d \n",head->data,second->data,third->data);
    printf("\nChoose one option from following: ");
    printf("\n1.Insert in beginning\n2.Insert at last\n3.Insert at any random position\n4.Exit");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
```

```
    switch(choice){
        case 1:
            head=begin_insert(head);
            display(head);
            break;
        case 2:
            last_insert(head);
            display(head);
            break;
        case 3:
            random_insert(head);
            display(head);
            break;
        case 4:
            printf("\nBye!");
            flag=0;
            break;
        default:
            printf("Invalid input....");
            break;
    }
}
free(head);
free(second);
free(third);
```

```
    free(temp);
    return 0;
}
```

```
Initial Doubly Linked List: 10  20  30

Choose one option from following:
1.Insert in beginning
2.Insert at last
3.Insert at any random position
4.Exit
Enter your choice: 1
Enter data: 40

New Doubly Linked list: 40 10 20 30
Enter your choice: 2
Enter data: 50

New Doubly Linked list: 40 10 20 30 50
Enter your choice: 3
Enter position: 2
Enter data: 60

New Doubly Linked list: 40 60 10 20 30 50
Enter your choice: 4

Bye!
Process returned 0 (0x0)   execution time : 21.997 s
Press any key to continue.
```

27. Write a program to perform following operations on Singly Circular Linked List: creation, traversal, insertion, deletion.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
struct node *insert_begin(struct node *tail){
   struct node *new;
   new=(struct node *)malloc (sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&new->data);
   new->next=NULL;
   new->next=tail->next;
   tail->next=new;
   return tail;
}
void insert_random(struct node *tail){
   struct node *new,*temp;
   temp=tail->next;
   int pos,i=1;
   printf("Enter the position: ");
   scanf("%d",&pos);
   new=(struct node *)malloc (sizeof(struct node));
```

```
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    new->next=temp->next;
    temp->next=new;
}
struct node *insert_last(struct node *tail){
    struct node *new;
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    new->next=tail->next;
    tail->next=new;
    tail=new;
    return tail;
}
struct node *delete_begin(struct node *tail){
    struct node *temp;
    temp=tail->next;
    tail->next=temp->next;
```

```
    free(temp);
    return tail;
}
void delete_random(struct node *tail){
    struct node *temp,*temp1;
    int i=1,pos;
    printf("Enter Position: ");
    scanf("%d",&pos);
    temp=tail->next;
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    temp1=temp->next;
    temp->next=temp1->next;
    free(temp1);
}
struct node *delete_last(struct node *tail){
    struct node *temp=NULL,*prev=NULL;
    temp=tail->next;
    while(temp->next!=tail->next){
        prev=temp;
        temp=temp->next;
    }
    prev->next=temp->next;
```

```
    tail=prev;
    free(temp);
    return tail;
}
void display(struct node *tail){
    struct node *temp=tail->next;
    while(temp->next!=tail->next){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("%d ",temp->data);
}
int main(){
    struct node *first,*second,*third,*four,*tail;
    int choice,flag=1;
    first=(struct node *)malloc(sizeof(struct node));
    second=(struct node *)malloc(sizeof(struct node));
    third=(struct node *)malloc(sizeof(struct node));
    four=(struct node *)malloc(sizeof(struct node));
    first->data=10;
    first->next=second;
    second->data=20;
    second->next=third;
    third->data=30;
    third->next=four;
```

```
    four->data=40;
    four->next=first;
    tail=four;
    printf("Initial Singly Circular Linked List: ");
    display(tail);
    printf("\nChoose any one of them: \n");
    printf("1.Insert at beginning \n2.Insert at any random position \n3.Insert at last \n4.Delete from beginning \n5.Delete from specific position \n6.Delete from end \n7.Exit \n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                tail=insert_begin(tail);
                printf("Node inserted at the beginning: ");
                display(tail);
                break;
            case 2:
                insert_random(tail);
                printf("Node inserted at specific position: ");
                display(tail);
                break;
            case 3:
                tail=insert_last(tail);
                printf("Node inserted at last: ");
```

```
            display(tail);
            break;
        case 4:
            tail=delete_begin(tail);
            printf("Node deleted from begin: ");
            display(tail);
            break;
        case 5:
            delete_random(tail);
            printf("Node deleted from specific position: ");
            display(tail);
            break;
        case 6:
            tail=delete_last(tail);
            printf("Node deleted from last: ");
            display(tail);
            break;
        case 7:
            printf("Bye!");
            flag=0;
            break;
        default:
            printf("Invalid key...");
            break;
    }
```

```
    }
    free(first);
    free(second);
    free(third);
    free(four);
    free(tail);
    return 0;
}
```

```
Initial Singly Circular Linked List: 10 20 30 40
Choose any one of them:
1.Insert at beginning
2.Insert at any random position
3.Insert at last
4.Delete from beginning
5.Delete from specific position
6.Delete from end
7.Exit

Enter your choice: 1
Enter data: 50
Node inserted at the beginning: 50 10 20 30 40
Enter your choice: 2
Enter the position: 3
Enter data: 60
Node inserted at specific position: 50 10 60 20 30 40
Enter your choice: 3
Enter data: 70
Node inserted at last: 50 10 60 20 30 40 70
Enter your choice: 4
Node deleted from begin: 10 60 20 30 40 70
Enter your choice: 5
Enter Position: 2
Node deleted from specific position: 10 20 30 40 70
Enter your choice: 6
Node deleted from last: 10 20 30 40
Enter your choice: 8
Invalid key...
Enter your choice: 7
Bye!
Process returned -1073740940 (0xC0000374)   execution time : 46.347 s
Press any key to continue.
```

28. Write a program to perform following operation on Doubly Circular Linked List: create, traverse, insert, delete.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *insert_begin(struct node *tail){
   struct node *new,*temp=NULL;
   new=(struct node *)malloc (sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&new->data);
   new->next=NULL;
   new->prev=NULL;
   temp=tail->next;
   new->next=temp;
   temp->prev=new;
   new->prev=tail;
   tail->next=new;
   return tail;
}
void insert_random(struct node *tail){
   struct node *new,*temp=NULL,*temp1=NULL;
```

```
    int pos,i=1;
    printf("Enter the position: ");
    scanf("%d",&pos);
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
    new->prev=NULL;
    temp=tail->next;
    while(i<pos-1){
       temp=temp->next;
       i++;
    }
    temp1=temp->next;
    temp1->prev=new;
    new->next=temp1;
    new->prev=temp;
    temp->next=new;
}
struct node *insert_last(struct node *tail){
    struct node *new,*temp=NULL;
    new=(struct node *)malloc (sizeof(struct node));
    printf("Enter data: ");
    scanf("%d",&new->data);
    new->next=NULL;
```

```
    new->prev=NULL;
    temp=tail->next;
    tail->next=new;
    new->prev=tail;
    new->next=temp;
    temp->prev=new;
    tail=new;
    return tail;
}
struct node *delete_begin(struct node *tail){
    struct node *temp=NULL,*temp1=NULL;
    temp=tail->next;
    temp1=temp->next;
    tail->next=temp1;
    temp1->prev=tail;
    free(temp);
    return tail;
}
void delete_random(struct node *tail){
    struct node *temp=NULL,*temp1=NULL,*temp2;
    int i=1,pos;
    printf("Enter Position: ");
    scanf("%d",&pos);
    temp=tail->next;
    while(i<pos-1){
```

```
        temp=temp->next;
        i++;
    }
    temp1=temp->next;
    temp2=temp1->next;
    temp->next=temp2;
    temp2->prev=temp;
    free(temp1);
}
struct node *delete_last(struct node *tail){
    struct node *temp=NULL,*pre=NULL,*temp1=NULL;
    temp=tail->next;
    while(temp->next!=tail->next){
        pre=temp;
        temp=temp->next;
    }
    temp1=temp->next;
    temp1->prev=pre;
    pre->next=temp1;
    tail=pre;
    free(temp);
    return tail;
}
void display(struct node *tail){
    struct node *temp=tail->next;
```

```
    while(temp->next!=tail->next){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("%d ",temp->data);
}
int main(){
    struct node *first,*second,*third,*four,*tail;
    int choice,flag=1;
    first=(struct node *)malloc(sizeof(struct node));
    second=(struct node *)malloc(sizeof(struct node));
    third=(struct node *)malloc(sizeof(struct node));
    four=(struct node *)malloc(sizeof(struct node));
    first->data=10;
    first->next=second;
    first->prev=four;
    second->data=20;
    second->next=third;
    second->prev=first;
    third->data=30;
    third->next=four;
    third->prev=second;
    four->data=40;
    four->next=first;
    four->prev=third;
```

```
    tail=four;
    printf("Initial Doubly Circular Linked List: ");
    display(tail);
    printf("\nChoose any one of them: \n");
    printf("1.Insert at beginning \n2.Insert at any random position \n3.Insert at last \n4.Delete from beginning \n5.Delete from specific position \n6.Delete from end \n7.Exit \n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                tail=insert_begin(tail);
                printf("Node inserted at the beginning: ");
                display(tail);
                break;
            case 2:
                insert_random(tail);
                printf("Node inserted at specific position: ");
                display(tail);
                break;
            case 3:
                tail=insert_last(tail);
                printf("Node inserted at last: ");
                display(tail);
                break;
```

```
        case 4:
            tail=delete_begin(tail);
            printf("Node deleted from begin: ");
            display(tail);
            break;
        case 5:
            delete_random(tail);
            printf("Node deleted from specific position: ");
            display(tail);
            break;
        case 6:
            tail=delete_last(tail);
            printf("Node deleted from last: ");
            display(tail);
            break;
        case 7:
            printf("Bye!");
            flag=0;
            break;
        default:
            printf("Invalid key...");
            break;
    }
}
free(first);
```

```
    free(second);
    free(third);
    free(four);
    free(tail);
    return 0;
}
```

```
Initial Doubly Circular Linked List: 10 20 30 40
Choose any one of them:
1.Insert at beginning
2.Insert at any random position
3.Insert at last
4.Delete from beginning
5.Delete from specific position
6.Delete from end
7.Exit

Enter your choice: 1
Enter data: 50
Node inserted at the beginning: 50 10 20 30 40
Enter your choice: 2
Enter the position: 3
Enter data: 60
Node inserted at specific position: 50 10 60 20 30 40
Enter your choice: 3
Enter data: 70
Node inserted at last: 50 10 60 20 30 40 70
Enter your choice: 4
Node deleted from begin: 10 60 20 30 40 70
Enter your choice: 5
Enter Position: 2
Node deleted from specific position: 10 20 30 40 70
Enter your choice: 6
Node deleted from last: 10 20 30 40
Enter your choice: 7
Bye!
Process returned 0 (0x0)   execution time : 30.212 s
Press any key to continue.
```

29. Write a program to print the middle element of the Linked List.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
   struct node *prev;
};
void mid_element(struct node *tail,int size){
   struct node *temp=NULL,*temp1;
   int mid=size/2;
   int i=0;
   temp=tail->next;
   temp1=tail;
   if(size%2!=0){
      while(i!=mid){
         temp=temp->next;
         i++;
      }
      printf("\nMid element is: %d",temp->data);
   }
   else{
      while(i!=mid){
         temp=temp->next;
         temp1=temp1->next;
```

```
            i++;
        }
        printf("\nMid element are: %d and %d ",temp1->data,temp->data);
    }
    free(temp);
    free(temp1);
}
int main(){
    struct node *new,*start=NULL,*tail=NULL,*temp=NULL;
    int choice=1,size=1,mid;
    while(choice==1){
        new=(struct node *)malloc (sizeof(struct node));
        printf("Enter data: ");
        scanf("%d",&new->data);
        new->next=NULL;
        new->prev=NULL;
        if(start==NULL){
            start=new;
            tail=new;
            new->prev=start;
            new->next=start;
        }
        else{
            tail->next=new;
            new->prev=tail;
```

```
            new->next=start;
            start->prev=new;
            tail=new;
        }
        printf("Do you want to add more then enter 1 else enter 0: ");
        scanf("%d",&choice);
    }
    printf("\nEntered elements: ");
    temp=tail->next;
    while(temp->next!=tail->next){
        size+=1;
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("%d ",temp->data);
    mid_element(tail,size);
    free(start);
    free(tail);
    free(temp);
    free(new);
    return 0;
}
```

```
Enter data: 10
Do you want to add more then enter 1 else enter 0: 1
Enter data: 20
Do you want to add more then enter 1 else enter 0: 1
Enter data: 30
Do you want to add more then enter 1 else enter 0: 1
Enter data: 40
Do you want to add more then enter 1 else enter 0: 0

Entered elements: 10 20 30 40
Mid element are: 20 and 30
Process returned 0 (0x0)   execution time : 21.282 s
Press any key to continue.
```

30. Write a program to check the linked list is palindrome or not.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
    struct node *prev;
};
void palindrome(struct node *temp,int size){
   struct node *i=NULL,*j=NULL,*start=NULL;
   int i_index,j_index,flag=0;
   start=temp->next;
   i=temp->next;
   j=temp;
   i_index=0;
   j_index=size;
   while(i_index<=j_index){
        if(i->data==j->data){
            i_index++;
            j_index--;
            i=i->next;
            j=j->prev;
        }
        else{
            flag=1;
```

```
            break;
        }
    }
    if(flag==0){
        printf("\nPalindrome");
    }
    else{
        printf("\nNot palindrome");
    }
}
int main(){
    struct node *new,*start=NULL,*tail=NULL,*temp=NULL;
    int choice=1,size=1;
    while(choice==1){
        new=(struct node *)malloc (sizeof(struct node));
        printf("Enter data: ");
        scanf("%d",&new->data);
        new->next=NULL;
        new->prev=NULL;
        if(start==NULL){
            start=new;
            tail=new;
            new->prev=start;
            new->next=start;
        }
```

```
        else{
            tail->next=new;
            new->prev=tail;
            new->next=start;
            start->prev=new;
            tail=new;
        }
        printf("Do you want to add more then enter 1 else enter 0: ");
        scanf("%d",&choice);
    }
    printf("\nEntered element: ");
    temp=tail->next;
    while(temp->next!=tail->next){
        size+=1;
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("%d ",temp->data);
    palindrome(temp,size);
    free(start);
    free(tail);
    free(temp);
    free(new);
    return 0;
}
```

```
Enter data: 1
Do you want to add more then enter 1 else enter 0: 1
Enter data: 3
Do you want to add more then enter 1 else enter 0: 1
Enter data: 3
Do you want to add more then enter 1 else enter 0: 1
Enter data: 1
Do you want to add more then enter 1 else enter 0: 0

Entered element: 1 3 3 1
Palindrome
Process returned -1073740940 (0xC0000374)   execution time : 14.868 s
Press any key to continue.
```

31. Write a program to add ,substract and multiply two polynomials.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int coeff;
   int expo;
   struct node *next;
};
struct node *insert(struct node *head,int co,int ex){
   struct node *temp,*temp1;
   struct node *new=malloc(sizeof(struct node));
   new->coeff=co;
   new->expo=ex;
   new->next=NULL;
   if(head==NULL || ex > head->expo){
      new->next=head;
      head=new;
   }
   else{
      temp=head;
      while(temp->next!=NULL && temp->next->expo > ex){
         temp=temp->next;
      }
      new->next=temp->next;
      temp->next=new;
```

```
    }
    return head;
}
struct node *create(struct node *head){
    int n,coeff,expo,i=0;
    printf("Enter the number of terms: ");
    scanf("%d",&n);
    while(i<n){
        printf("Enter the coefficient for term %d: ",i+1);
        scanf("%d",&coeff);
        printf("Enter the exponent for term %d: ",i+1);
        scanf("%d",&expo);
        i++;
        head=insert(head,coeff,expo);
    }
    return head;
}
void addPolynomial(struct node *head1,struct node *head2){
    struct node *temp1=head1;
    struct node *temp2=head2;
    struct node *head3=NULL;
    while(temp1!=NULL && temp2!=NULL){
        if(temp1->expo==temp2->expo){
            head3=insert(head3,(temp1->coeff + temp2->coeff),temp1->expo);
            temp1=temp1->next;
```

```
            temp2=temp2->next;
        }
        else if(temp1->expo > temp2->expo){
            head3=insert(head3,temp1->coeff,temp1->expo);
            temp1=temp1->next;
        }
        else if(temp1->expo < temp2->expo){
            head3=insert(head3,temp2->coeff,temp2->expo);
            temp2=temp2->next;
        }
    }
    while(temp1!=NULL){
        head3=insert(head3,temp1->coeff,temp1->expo);
        temp1=temp1->next;
    }
    while(temp2!=NULL){
        head3=insert(head3,temp2->coeff,temp2->expo);
        temp2=temp2->next;
    }
    printf("\nAddition of two polynomial: ");
    display(head3);
}
void subPolynomial(struct node *head1,struct node *head2){
    struct node *temp1=head1;
    struct node *temp2=head2;
```

```
struct node *head3=NULL;
while(temp1!=NULL && temp2!=NULL){
    if(temp1->expo==temp2->expo){
        head3=insert(head3,(temp1->coeff - temp2->coeff),temp1->expo);
        temp1=temp1->next;
        temp2=temp2->next;
    }
    else if(temp1->expo > temp2->expo){
        head3=insert(head3,temp1->coeff,temp1->expo);
        temp1=temp1->next;
    }
    else if(temp1->expo < temp2->expo){
        head3=insert(head3,temp2->coeff,temp2->expo);
        temp2=temp2->next;
    }
}
while(temp1!=NULL){
    head3=insert(head3,temp1->coeff,temp1->expo);
    temp1=temp1->next;
}
while(temp2!=NULL){
    head3=insert(head3,temp2->coeff,temp2->expo);
    temp2=temp2->next;
}
printf("\nSubstraction of two polynomial: ");
```

```
    display(head3);
}
void mulPolynomial(struct node *head1, struct node *head2) {
    struct node *temp1 = head1, *temp2 = head2, *head3 = NULL;
    while (temp1 != NULL) {
        temp2 = head2;
        while (temp2 != NULL) {
            int coeff = temp1->coeff * temp2->coeff;
            int expo = temp1->expo + temp2->expo;
            // Check if a node with the same exponent already exists in head3
            struct node *temp3 = head3;
            while (temp3 != NULL && temp3->expo != expo) {
                temp3 = temp3->next;
            }
            if (temp3 != NULL) {
                // Node with the same exponent exists, add the coefficients
                temp3->coeff += coeff;
            } else {
                // Create a new node and insert it into head3
                head3 = insert(head3, coeff, expo);
            }
            temp2 = temp2->next;
        }
        temp1 = temp1->next;
    }
```

```
    printf("\nMultiplication of two polynomials: ");
    display(head3);
}
void display(struct node *head){
    struct node *temp=head;
    if(head==NULL){
        printf("No polynomial");
    }
    while(temp!=NULL){
        printf("(%dx^%d)",temp->coeff,temp->expo);
        temp=temp->next;
        if(temp!=NULL){
            printf(" + ");
        }
    }
}
int main(){
    struct node *head1=NULL;
    struct node *head2=NULL;
    printf("Enter the first polynomial:\n");
    head1=create(head1);
    printf("\nFirst polynomial is: ");
    display(head1);
    printf("\nEnter the second polynomial:\n");
    head2=create(head2);
```

```
    printf("\nSecond polynomial is: ");
    display(head2);
    addPolynomial(head1,head2);
    subPolynomial(head1,head2);
    mulPolynomial(head1,head2);
    free(head1);
    free(head2);
    return 0;
}
```

```
Enter the first polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1: 2
Enter the exponent for term 1: 2
Enter the coefficient for term 2: 3
Enter the exponent for term 2: 1
Enter the coefficient for term 3: 5
Enter the exponent for term 3: 0

First polynomial is: (2x^2) + (3x^1) + (5x^0)
Enter the second polynomial:
Enter the number of terms: 4
Enter the coefficient for term 1: 5
Enter the exponent for term 1: 3
Enter the coefficient for term 2: 2
Enter the exponent for term 2: 2
Enter the coefficient for term 3: 3
Enter the exponent for term 3: 0
Enter the coefficient for term 4: 2
Enter the exponent for term 4: 1

Second polynomial is: (5x^3) + (2x^2) + (2x^1) + (3x^0)
Addition of two polynomial: (5x^3) + (4x^2) + (5x^1) + (8x^0)
Substraction of two polynomial: (5x^3) + (0x^2) + (1x^1) + (2x^0)
Multiplication of two polynomials: (10x^5) + (19x^4) + (35x^3) + (22x^2) + (19x^1) + (15x^0)
Process returned 0 (0x0)   execution time : 35.361 s
Press any key to continue.
```

32. Write a program to implement stack using array.

```
#include<stdio.h>
int n,top=-1;
int stack[50];
void push(){
   int val;
   if(top==(n-1)){
      printf("Overflow\n");
   }
   else{
      printf("Enter the value: ");
      scanf("%d",&val);
      top++;
      stack[top]=val;
      printf("Inserted in stack....\n");
   }
}
void pop(){
   if(top==-1){
      printf("Underflow\n");
   }
   else{
      int val=stack[top];
      top--;
      printf("Deleted item is: %d\n",val);
```

```
    }
}
void peek(){
    if(top==-1){
        printf("Stack is empty....\n");
    }else{
        printf("Topmost element is: %d\n",stack[top]);
    }
}
void display(){
    if(top==-1){
        printf("stack is empty...\n");
    }
    for(int i=top;i>=0;i--){
        printf("%d ",stack[i]);
    }
}
int main(){
    int choice,flag=1;
    printf("Enter the number of elements in the stack: ");
    scanf("%d",&n);
    printf("Choose one from the below options...\n");
    printf("1.Push\n2.Pop\n3.Peek\n4.Show\n5.Exit\n");
    while(flag){
        printf("\nEnter your choice: ");
```

```
        scanf("%d",&choice);
        switch(choice){
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
            case 5:
                printf("Bye....");
                flag=0;
                break;
            default:
                printf("Enter valid choice...");
                break;
        }
    }
    return 0;
}
```

```
Enter the number of elements in the stack: 3
Choose one from the below options...
1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter your choice: 1
Enter the value: 10
Inserted in stack....

Enter your choice: 1
Enter the value: 20
Inserted in stack....

Enter your choice: 1
Enter the value: 30
Inserted in stack....

Enter your choice: 2
Deleted item is: 30

Enter your choice: 3
Topmost element is: 20

Enter your choice: 4
20 10
Enter your choice: 5
Bye....
```

33. Write a program to implement stack using Linked List.

```
#include<stdio.h>
struct node{
   int data;
   struct node *next;
}*top=NULL;
void push(){
   struct node *new=(struct node *)malloc(sizeof(struct node));
   if(new==NULL){
      printf("Overflow....\n");
   }
   else{
      int val;
      printf("Enter the value: ");
      scanf("%d",&val);
      new->data=val;
      new->next=top;
      top=new;
   }
}
void pop(){
   struct node *temp;
   int item;
   if(top==NULL){
      printf("Under flow....\n");
```

```
    }
    else{
        item=top->data;
        temp=top;
        top=top->next;
        free(temp);
        printf("Deleted item is: %d \n",item);
    }
}
void peek(){
    if(top==NULL){
        printf("Stack is empty....\n");
    }
    else{
        printf("Topmost element is: %d\n",top->data);
    }
}
void display(){
    struct node *temp=top;
    if(temp==NULL){
        printf("Stack is empty...\n");
    }
    else{
       while(temp!=NULL){
           printf("%d ",temp->data);
```

```
            temp=temp->next;
        }
        printf("\n");
    }
}
int main(){
    int flag=1,choice;
    printf("Choose one from the below options...\n");
    printf("1.Push\n2.Pop\n3.Peek\n4.Display\n5.Exit\n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
```

```
        case 5:
            printf("Bye....");
            flag=0;
            break;
        default:
            printf("Enter valid choice...\n");
            break;
      }
   }
   return 0;
}
```

```
Choose one from the below options...
1.Push
2.Pop
3.Peek
4.Display
5.Exit

Enter your choice: 1
Enter the value: 10

Enter your choice: 1
Enter the value: 20

Enter your choice: 1
Enter the value: 30

Enter your choice: 2
Deleted item is: 30

Enter your choice: 4
20 10

Enter your choice: 3
Topmost element is: 20

Enter your choice: 5
Bye....
Process returned 0 (0x0)   execution time : 19.419 s
Press any key to continue.
```

34. Write a program to reverse string using stack.

```
#include<stdio.h>
#include<string.h>
#define MAX 20
char stack[MAX];
char str[MAX];
int top=-1;
void push(char ch){
   if(top==(MAX-1)){
      printf("Overflow");
   }
   else{
     stack[++top]=ch;
   }
}
char pop(){
   if(top==-1){
      printf("Underflow\n");
      return '\0';
   }
   else{
      return stack[top--];
   }
}
int main(){
```

```
    int i=0;
    printf("Enter a string: ");
    gets(str);
    while(str[i]!='\0'){
        push(str[i]);
        if(top==MAX-1){
            break;
        }
        i++;
    }
    for(int i=0;i<strlen(str);i++){
        str[i]=pop();
    }
   printf("\nReverse String is: ");
   puts(str);
   return 0;
}
```

```
Enter a string: Varun thapliyal

Reverse String is: layilpaht nuraV
```

35. Write a program to convert infix to postfix.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100
int top=-1;
char stack[MAX],postfix[MAX],infix[MAX];
int precedence(char symbol){
   switch(symbol){
      case '^':
         return 3;
      case '/':
      case '*':
         return 2;
      case '+':
      case '-':
         return 1;
      default:
         return 0;
   }
}
void print(){
   int i=0;
   printf("The equivalent postfix expression is: ");
   while(postfix[i]){
```

```
        printf("%c",postfix[i++]);
    }
    printf("\n");
}
void push(char c){
    if(top==MAX-1){
        printf("Overflow");
        return;
    }
    top++;
    stack[top]=c;
}
char pop(){
    char c;
    if(top==-1){
        printf("Underflow");
        exit(1);
    }
    c=stack[top];
    top-=1;
    return c;
}
int isEmpty(){
    if(top==-1){
        return 1;
```

```
    }
    else{
        return 0;
    }
}
void infixToPostfix(){
    int i,j=0;
    char symbol,next;
    for(int i=0;i<strlen(infix);i++){
        symbol=infix[i];
        switch(symbol){
            case '(':
                push(symbol);
                break;
            case ')':
                while((next=pop())!='('){
                    postfix[j++]=next;
                }
                break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
                while(!isEmpty() && precedence(stack[top])>= precedence(symbol)){
```

```
                postfix[j++]=pop();
            }
            push(symbol);
            break;
        default:
            postfix[j++]=symbol;
            break;
    }
  }
  while(!isEmpty()){
    postfix[j++]=pop();
  }
  postfix[j]='\0';
}
int main(){
  printf("Enter the infix expression: ");
  gets(infix);
  infixToPostfix();
  print();
  return 0;
}
```

```
Enter the infix expression: (a+b*(c-d))/e
The equivalent postfix expression is: abcd-*+e/

Process returned 0 (0x0)   execution time : 25.956 s
Press any key to continue.
```

36. Write a program to convert infix to prefix.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100
int top=-1;
char stack[MAX], prefix[MAX], infix[MAX];
int precedence(char symbol){
   switch(symbol){
      case '^':
         return 3;
      case '/':
      case '*':
         return 2;
      case '+':
      case '-':
         return 1;
      default:
         return 0;
   }
}
void print(){
   int i = strlen(prefix) - 1;
   printf("The equivalent prefix expression is: ");
   while(i >= 0){
```

```
        printf("%c", prefix[i--]);
    }
    printf("\n");
}
void push(char c){
    if(top==MAX-1){
        printf("Overflow");
        return;
    }
    top++;
    stack[top]=c;
}
char pop(){
    char c;
    if(top==-1){
        printf("Underflow");
        exit(1);
    }
    c=stack[top];
    top-=1;
    return c;
}
int isEmpty(){
    if(top==-1){
        return 1;
```

```
    }
    else{
        return 0;
    }
}
void reverseString(char str[]){
    int i = 0, j = strlen(str) - 1;
    while(i < j){
        char temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++;
        j--;
    }
}
void infixToPrefix(){
    int i, j = 0;
    char symbol, next;
    reverseString(infix); // Reverse the infix expression
    for(i = 0; i < strlen(infix); i++){
        symbol = infix[i];
        switch(symbol){
            case '(':
                push(symbol);
                break;
```

```
        case ')':
            while((next = pop()) != '('){
                prefix[j++] = next;
            }
            break;
        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
            while(!isEmpty() && precedence(stack[top]) >= precedence(symbol)){
                prefix[j++] = pop();
            }
            push(symbol);
            break;
        default:
            prefix[j++] = symbol;
          break;
    }
}
while(!isEmpty()){
    prefix[j++] = pop();
}
prefix[j] = '\0';
reverseString(prefix); // Reverse the prefix expression to get the correct order
```

```
}
int main(){
   printf("Enter the infix expression: ");
   gets(infix);
   infixToPrefix();
   print();
   return 0;
}
```

```
Enter the infix expression: a+b*c-d/e
The equivalent prefix expression is: ed/cb*-a+

Process returned 0 (0x0)   execution time : 19.624 s
Press any key to continue.
```

37. Write a program to implement queue using array.

```
#include<stdio.h>
#define N 5
int queue[N];
int rear=-1;
int front=-1;
void enqueue(){
   int num;
   printf("Enter the number: ");
   scanf("%d",&num);
   if(rear==N-1){
      printf("Overflow!\n");
   }
   else if(rear==-1 && front==-1){
      printf("Element inserted!\n");
      rear=front=0;
      queue[rear]=num;
   }
   else{
      printf("Element inserted!\n");
      rear++;
      queue[rear]=num;
   }
}
void dequeue(){
```

```
    if(rear==-1 && front==-1){
        printf("Underflow!\n");
    }
    else{
        printf("Deleted item: %d\n",queue[front]);
        front++;
    }
}
void peek(){
    printf("Topmost element: %d\n",queue[front]);
}
void display(){
    printf("Displaying Queue: ");
    for(int i=front;i<=rear;i++){
        printf("%d ",queue[i]);
    }
    printf("\n");
}
int main(){
    int choice,flag=1;
    printf("Following Operations: \n1.Enqueue Operation \n2.Dequeue Operation \n3.Peek
Operation \n4.Display Queue\n5.Exit\n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
```

```
        switch(choice){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
            case 5:
                printf("Exit....!");
                flag=0;
                break;
            default :
                printf("Enter the correct option");
                break;
        }
    }
    return 0;
}
```

```
Following Operations:
1.Enqueue Operation
2.Dequeue Operation
3.Peek Operation
4.Display Queue
5.Exit

Enter your choice: 1
Enter the number: 10
Element inserted!

Enter your choice: 1
Enter the number: 20
Element inserted!

Enter your choice: 1
Enter the number: 30
Element inserted!

Enter your choice: 4
Displaying Queue: 10 20 30

Enter your choice: 3
Topmost element: 10

Enter your choice: 4
Displaying Queue: 10 20 30

Enter your choice: 5
Exit....!
Process returned 0 (0x0)   execution time : 17.659 s
Press any key to continue.
```

38. Write a program to implement queue using Linked List.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
}*rear=NULL,*front=NULL;
void enqueue(){
    struct node *newnode= (struct node *)malloc(sizeof(struct node));
    printf("Enter the data: ");
    scanf("%d",&newnode->data);
    newnode->next=NULL;
    if(rear==NULL && front==NULL){
        printf("Inserted....\n");
        front=newnode;
        rear=newnode;
    }
    else{
        printf("Inserted....\n");
        rear->next=newnode;
        rear=newnode;
    }
}
void dequeue(){
    struct node *temp;
```

```
    temp=front;
    if(front!=NULL){
        printf("Deleted item is: %d\n",temp->data);
        front=front->next;
        free(temp);
    }
    else{
        front=NULL;
        rear=NULL;
        printf("Underflow\n");
    }
}
void peek(){
    if(rear==NULL && front==NULL){
        printf("No elements...\n");
    }
    else{
        printf("Topmost element is: %d\n",front->data);
    }
}
void display(){
    struct node *temp;
    temp=front;
    printf("Element is: ");
    while(temp!=NULL){
```

```
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
int main(){
    int choice,flag=1;
    printf("Following Operations: \n1.Enqueue Operation \n2.Dequeue Operation \n3.Peek Operation \n4.Display Queue\n5.Exit\n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
```

```
        case 5:
            printf("Exit....!");
            flag=0;
            break;
        default :
            printf("Enter the correct option");
            break;
      }
   }
   return 0;
}
```

```
Following Operations:
1.Enqueue Operation
2.Dequeue Operation
3.Peek Operation
4.Display Queue
5.Exit

Enter your choice: 1
Enter the data: 10
Inserted....

Enter your choice: 1
Enter the data: 20
Inserted....

Enter your choice: 1
Enter the data: 30
Inserted....

Enter your choice: 2
Deleted item is: 10

Enter your choice: 3
Topmost element is: 20

Enter your choice: 4
Element is: 20 30

Enter your choice: 5
Exit....!
```

39. Write a program to implement circular queue using array.

```
#include<stdio.h>
#define N 5
int queue[N];
int rear=-1,front=-1;
void enqueue(){
   int num;
   printf("Enter the number: ");
   scanf("%d",&num);
   if(front==-1 && rear==-1){
      front=rear=0;
      queue[rear]=num;
   }
   else if((rear+1)%N == front){
      printf("Overflow\n");
   }
   else{
      rear=(rear+1)%N;
      queue[rear]=num;
   }
}
void dequeue(){
   if(front==-1 && rear==-1){
      printf("Underflow\n");
   }
```

```
    else if(front==rear){
        printf("Deleted element is: %d\n",queue[front]);
        front=rear=-1;
    }
    else {
        printf("Deleted element is: %d \n",queue[front]);
        front=(front+1)%N;
    }
}
void peek(){
    printf("Front element is %d \n",queue[front]);
}
void display(){
    int i=front;
    while(i!=rear){
        printf("%d ",queue[i]);
        i=(i+1)%N;
    }
    printf("%d \n",queue[i]);
}
int main(){
    int choice,flag=1;
    printf("Following Operations: \n1.Enqueue Operation \n2.Dequeue Operation \n3.Peek
Operation \n4.Display Queue\n5.Exit\n");
    while(flag){
```

```
    printf("\nEnter your choice: ");
    scanf("%d",&choice);
    switch(choice){
        case 1:
            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:
            peek();
            break;
        case 4:
            display();
            break;
        case 5:
            printf("Exit....!");
            flag=0;
            break;
        default :
            printf("Enter the correct option");
            break;
    }
}
return 0;
```

}

```
Following Operations:
1.Enqueue Operation
2.Dequeue Operation
3.Peek Operation
4.Display Queue
5.Exit

Enter your choice: 1
Enter the number: 10

Enter your choice: 1
Enter the number: 20

Enter your choice: 1
Enter the number: 30

Enter your choice: 3
Front element is 10

Enter your choice: 2
Deleted element is: 10

Enter your choice: 4
20 30

Enter your choice: 5
Exit....!
Process returned 0 (0x0)   execution time : 18.899 s
Press any key to continue.
```

40. Write a program to implement circular queue using Linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
}*front=NULL,*rear=NULL,*newnode;

void enqueue(){
   newnode=(struct node *)malloc(sizeof(struct node));
   printf("Enter data: ");
   scanf("%d",&newnode->data);
   newnode->next=NULL;
   if(front==NULL && rear==NULL){
      front=newnode;
      rear=newnode;
      rear->next=front;
   }
   else{
      rear->next=newnode;
      rear=newnode;
      rear->next=front;
   }
}
```

```
void dequeue(){
    struct node *temp;
    if(front==NULL && rear==NULL){
        printf("Underflow\n");
    }
    else if(front==rear){
        temp=front;
        printf("Deleted element is : %d \n",temp->data);
        front=NULL;
        rear=NULL;
        free(temp);
    }
    else{
        temp=front;
        printf("Deleted element is : %d \n",temp->data);
        front=front->next;
        free(temp);
    }
}

void display(){
    struct node *temp;
    temp = front;
    if (temp == NULL) {
        printf("Queue is empty.\n");
```

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



void peek(){
    struct node *temp;
    temp=front;
    printf("Topmost element is : %d \n",temp->data);
}


int main(){
    int choice,flag=1;
    printf("Following Operations: \n1.Enqueue Operation \n2.Dequeue Operation \n3.Peek
Operation \n4.Display Queue\n5.Exit\n");
    while(flag){
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
```

```
            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:
            peek();
            break;
        case 4:
            display();
            break;
        case 5:
            printf("Exit....!");
            flag=0;
            break;
        default :
            printf("Enter the correct option \n");
            break;
      }
   }
   return 0;
}
```

```
Following Operations:
1.Enqueue Operation
2.Dequeue Operation
3.Peek Operation
4.Display Queue
5.Exit

Enter your choice: 1
Enter data: 10

Enter your choice: 1
Enter data: 20

Enter your choice: 1
Enter data: 30

Enter your choice: 3
Topmost element is : 10

Enter your choice: 2
Deleted element is : 10

Enter your choice: 4
20 30

Enter your choice: 5
Exit....!
Process returned 0 (0x0)   execution time : 11.446 s
Press any key to continue.
```

41. Write a program to implement stack using queue.

```
#include <stdio.h>
#define N 5
int stack1[N];
int top1 = -1;
int count = 0;
void push(){
   int num;
   printf("Enter the number: ");
   scanf("%d",&num);
   if(top1==N-1) {
      printf("Overflow\n");
   }
   else{
      top1++;
      stack1[top1]=num;
      count++;
   }
}
void pop(){
   if(top1==-1) {
      printf("Stack is empty\n");
      return -1;
   }
   else{
```

```
        int item=stack1[top1];
        top1--;
        count--;
        printf("Popped element is: %d\n",item);
    }
}
void display(){
    if(top1==-1) {
        printf("Stack is empty\n");
    }
    else{
        printf("Stack: ");
        for (int i=0;i<=top1;i++) {
            printf("%d ",stack1[i]);
        }
        printf("\n");
    }
}
void top(){
    if(top1==-1) {
        printf("Stack is empty\n");
    }
    else{
        printf("Top element: %d\n",stack1[top1]);
    }
```

```
}
int main() {
   int choice, flag = 1;
   printf("Following Operations:\n1. Push Operation\n2. Pop Operation\n3. Display Stack\n4. Top Element\n5. Exit\n");
   while (flag) {
      printf("\nEnter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            push();
            break;
         case 2:
            pop();
            break;
         case 3:
            display();
            break;
         case 4:
            top();
            break;
         case 5:
            printf("Exit....!\n");
            flag=0;
            break;
```

```
        default:
            printf("Enter the correct option\n");
            break;
      }
   }
   return 0;
}
```

```
Following Operations:
1. Push Operation
2. Pop Operation
3. Display Stack
4. Top Element
5. Exit

Enter your choice: 1
Enter the number: 10

Enter your choice: 1
Enter the number: 20

Enter your choice: 1
Enter the number: 30

Enter your choice: 1
Enter the number: 40

Enter your choice: 1
Enter the number: 50

Enter your choice: 1
Enter the number: 60
Overflow

Enter your choice: 3
Stack: 10 20 30 40 50

Enter your choice: 2
Popped element is: 50

Enter your choice: 3
Stack: 10 20 30 40

Enter your choice: 4
Top element: 40
```

42. Write a program to implement queue using stack.

```
#include<stdio.h>
#define N 5
int stack1[N],stack2[N];
int top1=-1,top2=-1,count=0;
void push1(int num){
   if(top1==N-1){
      printf("Overflow\n");
   }
   else{
      top1++;
      stack1[top1]=num;
   }
}
int pop1(){
   return (stack1[top1--]);
}
int pop2(){
   return (stack2[top2--]);
}
void push2(int num){
   if(top2==N-1){
      printf("Overflow\n");
   }
   else{
```

```
        top2++;
        stack2[top2]=num;
    }
}
void enqueue(){
    int num;
    printf("Enter the number: ");
    scanf("%d",&num);
    push1(num);
    count++;
}
void dequeue(){
    if(top1==-1 && top2==-1){
        printf("Queue is empty \n");
    }
    else{
        for(int i=0;i<count;i++){
            push2(pop1());
        }
        printf("Deleted element is: %d \n", pop2());
        count--;
        for(int i=0;i<count;i++){
            push1(pop2());
        }
    }
```

```
}
void peek(){
   if(top1==-1){
      printf("No element\n");
   }
   else{
      printf("Topmost element is: %d \n",stack1[0]);
   }
}
void display(){
   if(top1 == -1){
      printf("No element\n");
   }
   else{
      for(int i=0;i<=top1;i++){
         printf("%d ",stack1[i]);
      }
   }
}
int main(){
   int choice,flag=1;
   printf("Following Operations: \n1.Enqueue Operation \n2.Dequeue Operation \n3.Peek
Operation \n4.Display Queue\n5.Exit\n");
   while(flag){
      printf("\nEnter your choice: ");
```

```
        scanf("%d",&choice);
        switch(choice){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
            case 5:
                printf("Exit....!");
                flag=0;
                break;
            default :
                printf("Enter the correct option");
                break;
        }
    }
    return 0;
}
```

```
Following Operations:
1.Enqueue Operation
2.Dequeue Operation
3.Peek Operation
4.Display Queue
5.Exit

Enter your choice: 1
Enter the number: 10

Enter your choice: 1
Enter the number: 20

Enter your choice: 1
Enter the number: 30

Enter your choice: 4
10 20 30
Enter your choice: 2
Deleted element is: 10

Enter your choice: 3
Topmost element is: 20

Enter your choice: 5
Exit....!
Process returned 0 (0x0)   execution time : 21.673 s
Press any key to continue.
```

43. Write a program to implement Binnary Tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *left,*right;
};
struct node *create(){
   struct node *newnode;
   int x;
   newnode=(struct node *) malloc (sizeof(struct node));
   printf("Enter data (Press -1 for no node): ");
   scanf("%d",&x);
   if(x==-1){
      return NULL;
   }
   newnode->data=x;
   printf("Enter the left child: \n");
   newnode->left=create();
   printf("Enter the right child: \n");
   newnode->right=create();
   return newnode;
}
void preorder(struct node *root){
   if(root==NULL){
```

```
        return;
    }
    printf("%d ",root->data);
    preorder(root->left);
    preorder(root->right);
}
void inorder(struct node *root){
    if(root==NULL){
        return;
    }
    inorder(root->left);
    printf("%d ",root->data);
    inorder(root->right);
}
void postorder(struct node *root){
    if(root==NULL){
        return;
    }
    postorder(root->left);
    postorder(root->right);
    printf("%d ",root->data);
}
int main(){
    struct node *root;
    root=create();
```

```
    printf("\nPreorder is: ");
    preorder(root);
    printf("\nInorder is: ");
    inorder(root);
    printf("\nPostorder is: ");
    postorder(root);
    return 0;
}
```

```
Enter data (Press -1 for no node): 10
Enter the left child:
Enter data (Press -1 for no node): 30
Enter the left child:
Enter data (Press -1 for no node): 50
Enter the left child:
Enter data (Press -1 for no node): 40
Enter the left child:
Enter data (Press -1 for no node): 60
Enter the left child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): 90
Enter the left child:
Enter data (Press -1 for no node): 70
Enter the left child:
Enter data (Press -1 for no node): 11
Enter the left child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1
Enter the right child:
Enter data (Press -1 for no node): -1

Preorder is: 10 30 50 40 60 90 70 11
Inorder is: 60 40 50 30 11 70 90 10
Postorder is: 60 40 50 11 70 90 30 10
Process returned 0 (0x0)   execution time : 41.218 s
Press any key to continue.
```

44. Write a program to implement Binary Search Tree.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node* left,*right;
};
struct node* createNode(int data) {
   struct node* newNode = (struct node*)malloc(sizeof(struct node));
   newNode->data = data;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}
struct node* insert(struct node* root, int data) {
   if(root==NULL){
      return createNode(data);
   }
   if (data<root->data) {
      root->left = insert(root->left,data);
   } else {
      root->right=insert(root->right,data);
   }
   return root;
}
```

```
void preorder(struct node* root) {
    if (root==NULL) {
        return;
    }
    printf("%d ",root->data);
    preorder(root->left);
    preorder(root->right);
}
void inorder(struct node* root) {
    if (root==NULL) {
        return;
    }
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}
void postorder(struct node* root) {
    if (root == NULL) {
        return;
    }
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
struct node* findMinNode(struct node* node) {
```

```
    struct node* current = node;
    while (current && current->left != NULL) {
        current=current->left;
    }
    return current;
}
struct node* deleteNode(struct node* root, int data) {
    if (root == NULL) {
        return root;
    }
    if (data < root->data) {
        root->left = deleteNode(root->left, data);
    } else if (data > root->data) {
        root->right = deleteNode(root->right, data);
    } else {
        // Node to be deleted found
        // Case 1: No child or only one child
        if (root->left==NULL) {
            struct node* temp=root->right;
            free(root);
            return temp;
        } else if (root->right==NULL) {
            struct node* temp=root->left;
            free(root);
            return temp;
```

```
        }
        // Case 2: Two children
        struct node* temp=findMinNode(root->right);
        root->data=temp->data;
        root->right=deleteNode(root->right, temp->data);
    }
    return root;
}
int main() {
    struct node* root = NULL;
    int n, data;
    printf("Enter the number of nodes in the binary search tree: ");
    scanf("%d", &n);
    printf("Enter the data for each node:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d",&data);
        root=insert(root, data);
    }
    printf("Preorder traversal: ");
    preorder(root);
    printf("\nInorder traversal: ");
    inorder(root);
    printf("\nPostorder traversal: ");
    postorder(root);
    int key;
```

```
    printf("\nEnter the data to delete: ");
    scanf("%d",&key);
    root=deleteNode(root, key);
    printf("\nInorder traversal after deletion: ");
    inorder(root);
    printf("\n");
    return 0;
}
```

```
Enter the number of nodes in the binary search tree: 5
Enter the data for each node:
10
30
20
8
6
Preorder traversal: 10 8 6 30 20
Inorder traversal: 6 8 10 20 30
Postorder traversal: 6 8 20 30 10
Enter the data to delete: 20

Inorder traversal after deletion: 6 8 10 30

Process returned 0 (0x0)   execution time : 33.820 s
Press any key to continue.
```

45. Write a program to implement AVL Tree.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int key;
    struct Node* left;
    struct Node* right;
    int height;
};
int max(int a, int b) {
    return (a > b) ? a : b;
}
int height(struct Node* node) {
    if (node == NULL) {
        return 0;
    }
    return node->height;
}
int getBalanceFactor(struct Node* node) {
    if (node == NULL) {
        return 0;
    }
    return height(node->left) - height(node->right);
}
struct Node* createNode(int key) {
```

```
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->key = key;
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->height = 1;
    return newNode;
}
struct Node* rotateRight(struct Node* y) {
    struct Node* x = y->left;
    struct Node* T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}
struct Node* rotateLeft(struct Node* x) {
    struct Node* y = x->right;
    struct Node* T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
}
```

```
struct Node* insert(struct Node* node, int key) {
    if (node == NULL) {
        return createNode(key);
    }
    if (key < node->key) {
        node->left = insert(node->left, key);
    } else if (key > node->key) {
        node->right = insert(node->right, key);
    } else {
        // Duplicate keys are not allowed in AVL tree
        return node;
    }
    node->height = 1 + max(height(node->left), height(node->right));
    int balanceFactor = getBalanceFactor(node);
    if (balanceFactor > 1) {
        // Left Left case
        if (key < node->left->key) {
            return rotateRight(node);
        }
        // Left Right case
        else if (key > node->left->key) {
            node->left = rotateLeft(node->left);
            return rotateRight(node);
        }
    }
```

```
    if (balanceFactor < -1) {
        // Right Right case
        if (key > node->right->key) {
            return rotateLeft(node);
        }
        // Right Left case
        else if (key < node->right->key) {
            node->right = rotateRight(node->right);
            return rotateLeft(node);
        }
    }
    return node;
}
struct Node* findMinValueNode(struct Node* node) {
    struct Node* current = node;
    while (current->left != NULL) {
        current = current->left;
    }
    return current;
}
struct Node* deleteNode(struct Node* root, int key) {
    if (root == NULL)
        return root;
    if (key < root->key) {
        root->left=deleteNode(root->left, key);
```

```
} else if (key>root->key) {
    root->right=deleteNode(root->right, key);
}
else{
    if(root->left == NULL || root->right == NULL){
        struct Node* temp=root->left ? root->left : root->right;
        if (temp==NULL){
            temp=root;
            root=NULL;
        } else
            *root=*temp;
        free(temp);
    } else{
        struct Node* temp=findMinValueNode(root->right);
        root->key=temp->key;
        root->right=deleteNode(root->right,temp->key);
    }
}
if(root==NULL){
    return root;
}
root->height=1+max(height(root->left),height(root->right));
int balanceFactor=getBalanceFactor(root);
if (balanceFactor>1) {
    if (getBalanceFactor(root->left)>=0) {
```

```
            return rotateRight(root);
        } else {
            root->left=rotateLeft(root->left);
            return rotateRight(root);
        }
    }
    if(balanceFactor < -1){
        if(getBalanceFactor(root->right)<=0) {
            return rotateLeft(root);
        }
        else{
            root->right=rotateRight(root->right);
            return rotateLeft(root);
        }
    }
    return root;
}
void preorderTraversal(struct Node* root) {
    if(root==NULL){
        return;
    }
    printf("%d ",root->key);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}
```

```
void inorderTraversal(struct Node* root) {
    if(root==NULL){
        return;
    }
    inorderTraversal(root->left);
    printf("%d ",root->key);
    inorderTraversal(root->right);
}
void postorderTraversal(struct Node* root) {
    if(root==NULL){
        return;
    }
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ",root->key);
}
int main() {
    struct Node* root = NULL;
    int choice,key;
    printf("Enter the elements to be inserted into the AVL tree (-1 to stop):\n");
    while(1){
        printf("Enter element: ");
        scanf("%d",&key);
        if(key==-1) {
            break;
```

```
        }
        root=insert(root,key);
    }
    printf("\nPreorder traversal of the AVL tree: ");
    preorderTraversal(root);
    printf("\nInorder traversal of the AVL tree: ");
    inorderTraversal(root);
    printf("\nPostorder traversal of the AVL tree: ");
    postorderTraversal(root);
    printf("\nEnter the element to be deleted from the AVL tree: ");
    scanf("%d", &key);
    root = deleteNode(root, key);
    printf("\nInorder traversal of the AVL tree after deletion: ");
    inorderTraversal(root);
    printf("\n");
    return 0;
}
```

```
Enter the elements to be inserted into the AVL tree (-1 to stop):
Enter element: 10
Enter element: 20
Enter element: 50
Enter element: 11
Enter element: 5
Enter element: 35
Enter element: 26
Enter element: -1

Preorder traversal of the AVL tree: 20 10 5 11 35 26 50
Inorder traversal of the AVL tree: 5 10 11 20 26 35 50
Postorder traversal of the AVL tree: 5 11 10 26 50 35 20

Enter the element to be deleted from the AVL tree: 26

Inorder traversal of the AVL tree after deletion: 5 10 11 20 35 50

Process returned 0 (0x0)   execution time : 24.372 s
Press any key to continue.
```

Q46 WAP to store graph information in the form of adjacency matrix.

```
#include <stdio.h>
int main(){
int e,u,v,vertices,i,j;
printf("Enter no. of Vertices : ");
scanf("%d",&vertices);
printf("Enter no. of Edges : ");
scanf("%d",&e);
int g[vertices][vertices];
for(i=0;i<vertices;i++){
for(j=0;j<vertices;j++){
g[i][j] =0;
}
}
printf("Enter Connected Edges (u,v): \n");
for(i = 0;i<e;i++){
scanf("%d %d",&u,&v);
g[u][v] = 1;
g[v][u] = 1;
}
printf("\nAdjacency Matrix : \n");
for (i=0;i<vertices;i++){
for (j=0;j<vertices;j++){
```

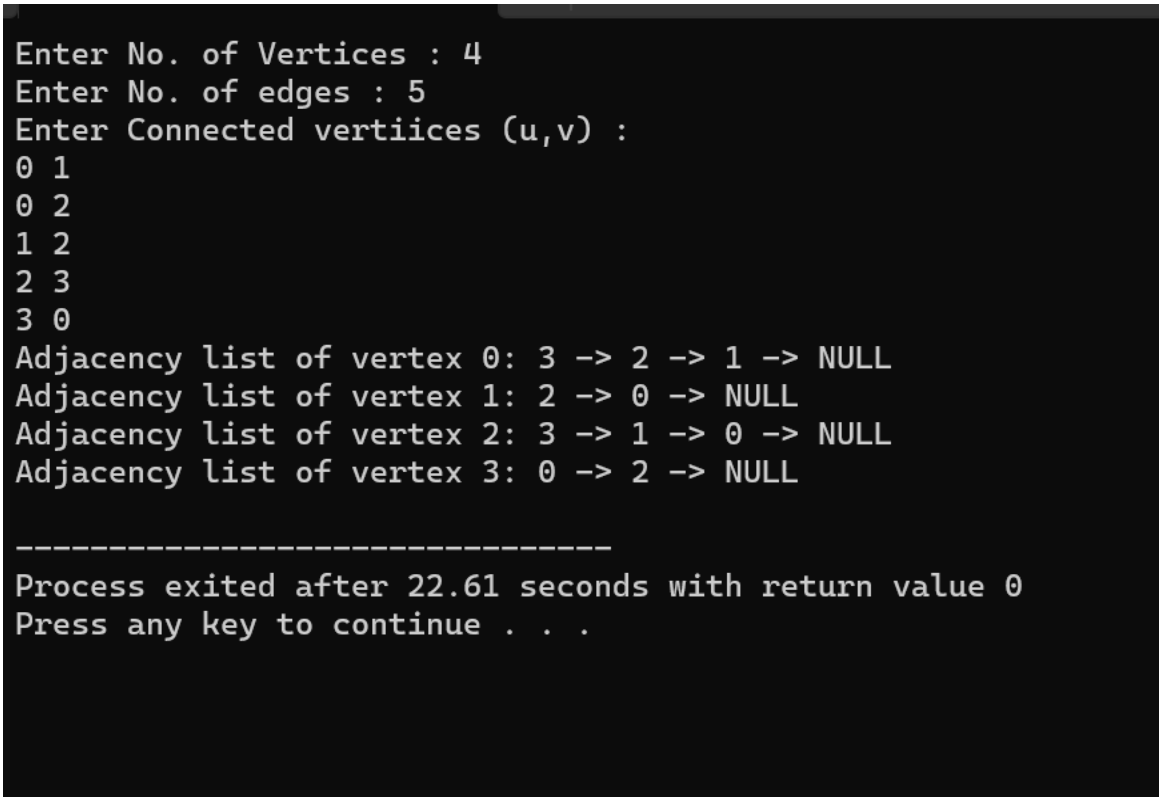
```
if(g[i][j]!=1){
printf("0 ");
}
else
printf("%d ",g[i][j]);
}
printf("\n");
}
return 0;
}
```

```
Enter no. of Vertices : 4
Enter no. of Edges : 5
Enter Connected Edges (u,v):
0 1
0 2
0 3
1 2
2 3

Adjacency Matrix :
0 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0

--------------------------------
Process exited after 53.86 seconds with return value 0
Press any key to continue . . .
```

Q47 WAP to store graph information in the form of adjacency List.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int vertex;
struct node *next;
};
struct adjlist{
struct node *head;
};
int main(){
int vertices,e,u,v,i;
printf("Enter No. of Vertices : ");
scanf("%d",&vertices);
struct adjlist *adjlst = (struct adjlist *)malloc(vertices*sizeof(struct adjlist));
for (i = 0; i < vertices; i++) {
adjlst[i].head = NULL;
}
printf("Enter No. of edges : ");
scanf("%d",&e);
printf("Enter Connected vertiices (u,v) : \n");
for(i=0;i<e;i++){
scanf("%d %d",&u,&v);
```

```
struct node *newnode = (struct node *)malloc(sizeof(struct node));
newnode->vertex = v;
newnode->next = adjlst[u].head;
adjlst[u].head = newnode;
struct node *newnode1 = (struct node *)malloc(sizeof(struct node));
newnode1->vertex = u;
newnode1->next = adjlst[v].head;
adjlst[v].head = newnode1;
}
for (i = 0; i < vertices; i++) {
struct node *temp = adjlst[i].head;
printf("Adjacency list of vertex %d: ", i);
while (temp) {
printf("%d -> ", temp->vertex);
temp = temp->next;
}
printf("NULL\n");
}
return 0;
}
```


Enter No. of Vertices : 4
Enter No. of edges : 5
Enter Connected vertiices (u,v) :
0 1
0 2
1 2
2 3
3 0
Adjacency list of vertex 0: 3 -> 2 -> 1 -> NULL
Adjacency list of vertex 1: 2 -> 0 -> NULL
Adjacency list of vertex 2: 3 -> 1 -> 0 -> NULL
Adjacency list of vertex 3: 0 -> 2 -> NULL

Process exited after 22.61 seconds with return value 0
Press any key to continue . . .