NAME PUSHPENDRA PATEL

EN 21115113

BATCH P6

```
#include<iostream>

using namespace std;
void multi(  float result [][6] , float p[][6] ,float q[][6] ,int n, int m ,int o  );
void showmatrix(float p[][6] ,int n, int m);
void sum(  float result [][6] , float p[][6] ,float q[][6] ,int n, int m   );
void transpose(float r[][6] , float p[][6] ,int n, int m);
void difference(  float result [][6] , float p[][6] ,float q[][6] ,int n, int m   );
bool inverse(float result[][6],float p[][6],int n);
void getCofactor(float A[][6], float temp[][6], int p, int q, int n);
float determinant(float A[][6], int n);
void adjoint(int A[][6],int adj[][6],int N);
int main()
{
        int node , ele ;
        cout<<"Enter number of element "<<endl;
        cin>>ele;
        cout<<"Enter number of node "<<endl;
        cin>>node;
        float input[ele][5];
        for(int i=0 ;i<ele;i++)
        {
                for(int j=0;j<5;j++)
                {
                 switch(j)
                   {
                        case 0:
                                cout<<"enter starting node of element "<<i+1<<endl;
```

```
                cin>>input[i][j];
                break;
            case 1:
                cout<<"enter ending node of element "<<i+1<<endl;
                cin>>input[i][j];
                break;
            case 2:
                cout<<"enter souce current of element "<<i+1<<endl;
                cin>>input[i][j];
                break;
            case 3:
                cout<<"enter souce voltage of element "<<i+1<<endl;
                cin>>input[i][j];
                break;
            case 4:
                cout<<"enter conductance of element "<<i+1<<endl;
                cin>>input[i][j];
                break;

          }
        }
}
float ise[ele][6] , vse[ele][6] , conductance[ele][6];
 for(int i = 0 ; i<ele;i++)
 {
        ise[i][0]=input[i][2];
        vse[i][0] =input[i][3];

 }
for(int i=0;i<ele;i++)
{
```

```
		for(int j=0; j<ele; j++)
		{
			if(i==j)
			{
				conductance[i][i]=input[i][4];
			}
			else{
				conductance[i][j] = 0;
			}
		}
}
float A[node-1][6];
for(int i=0;i<ele;i++)
{
		for(int j=1;j<node;j++)
		{
			if(j==input[i][0])
			{
				A[j-1][i]= 1;
			}else if(j==input[i][1])
			{
				A[j-1][i] =-1;
			}else{
				A[j-1][i] =0;
			}
		}
}



float at[ele][6];
```

```
transpose(at,A,ele,node-1);
float Y1[node-1][6];
multi(Y1,A,conductance,node-1,ele,ele);
float Y[node-1][6];
multi(Y,Y1,at,node-1,node-1,ele);

float isn11[node-1][6];
multi(isn11,A,conductance,node-1,ele,ele);
        float isn1[node-1][6];
        multi(isn1,isn11,vse,node-1,1,ele);
        float isn2[node-1][6];
        multi(isn2,A,ise,node-1,1,ele);
        float isn[node-1][6];
        difference(isn,isn1,isn2,node-1,1);
        cout<<endl;

        float Yin[node-1][6];
        inverse(Yin,Y,node-1);
        float Vn[node-1][6];
        multi(Vn,Yin,isn,node-1,1,node-1);

        float Ve[ele][6];
        multi(Ve,at,Vn,ele,1,node-1);
        cout<<"matrix ve = "<<endl;
        showmatrix(Ve,ele,1);
        float ie1[ele][6];
        multi(ie1,conductance,Ve,ele,1,ele);
        float ie2[ele][6];
        multi(ie2,conductance,vse,ele,1,ele);
        float ie3[ele][6];
        difference(ie3,ie1,ie2,ele,1);
```

```
	float ie[ele][6];
	sum(ie,ie3,ise,ele,1);
	cout<<"matrix ie = "<<endl;
	showmatrix(ie,ele,1);
	return 0;
}
*/
void multi(  float r [][6] , float p[][6] ,float q[][6] ,int n,int m ,int o )
 {
	for( int i=0;i<n;i++)
	{
		for( int j =0 ;j<m;j++)
		{  float sum =0;
			for(int k=0;k<o;k++)
			{
			r[i][j] += p[i][k]*q[k][j];
			 }
		 }
	 }
 }
void showmatrix(float p[][6] ,int n,  int m)
{
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<m;j++)
		{
			cout<<p[i][j]<<"\t";
		}
		cout<<endl;
	}
}
```

```
void sum(  float r [][6] , float p[][6] ,float q[][6] ,int n,int m  )
 {
          for( int i=0;i<n;i++)
          {
                    for( int j =0 ;j<m;j++)
                    {
                              r[i][j]=p[i][j]+q[i][j];
                     }
           }
 }
void transpose(float r[][6],float p[][6] ,int n,  int m)
{
          for(int i=0;i<n;i++)
          {
                    for(int j=0;j<m;j++)
                    {
                              r[i][j] = p[j][i];
                    }
                    cout<<endl;
          }
}
void difference(  float r [][6] , float p[][6] ,float q[][6] ,int n,int m  )
 {
          for( int i=0;i<n;i++)
          {
                    for( int j =0 ;j<m;j++)
                    {
                              r[i][j]=p[i][j]-q[i][j];
                     }
}
}
```

```
void getCofactor(float A[][6], float temp[][6], int p, int q, int n)
{
   int i = 0, j = 0;



   for (int row = 0; row < n; row++)
   {
      for (int col = 0; col < n; col++)
      {



         if (row != p && col != q)
         {
            temp[i][j++] = A[row][col];


            if (j == n - 1)
            {
               j = 0;
               i++;
            }
         }
      }
   }
}



float determinant(float A[][6], int n)
{
   float D = 0;
```

```
    if (n == 1)
        return A[0][0];

    float temp[n][6];

    int sign = 1;


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

        getCofactor(A, temp, 0, f, n);
        D += sign * A[0][f] * determinant(temp, n - 1);


        sign = -sign;
    }

    return D;
}


void adjoint(float A[][6],float adj[][6],int N)
{
    if (N == 1)
    {
        adj[0][0] = 1;
        return;
    }
```

```
    int sign = 1;
            float temp[N][6];

    for (int i=0; i<N; i++)
    {
       for (int j=0; j<N; j++)
       {
          getCofactor(A, temp, i, j, N);

          sign = ((i+j)%2==0)? 1: -1;

          adj[j][i] = (sign)*(determinant(temp, N-1));
       }
    }
}

bool inverse( float inverse[][6], float A[][6],int N)
{

    float det = determinant(A, N);
    if (det == 0)
    {
       cout << "Singular matrix, can't find its inverse";
       return false;
    }


    float adj[N][6];
    adjoint(A, adj,  N);

    // Find Inverse using formula "inverse(A) = adj(A)/det(A)"
```

```
    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
            inverse[i][j] = adj[i][j]/float(det);

    return true;
}
```