```
/* To find the inverse of a matrix using LU decomposition */

/* standard Headers */
        #include<math.h>
        #include<stdio.h>
        main()
        {
/* Variable declarations */
        int i,j,n,m,an,am;
        float D[3][3],d[3],C[3][3];
        float x,s[3][3],y[3];
        void LU();
        FILE *FP,*fp1;
        an=3; am=3;

        n=2;
/* the matrix to be inverted */
        D[0][0]=2.0;
        D[0][1]=4.0;
        D[0][2]=6.0;
        D[1][0]=4.0;
        D[1][1]=9.0;
        D[1][2]=3.;
        D[2][0]=6.0;
        D[2][1]=3.0;
        D[2][2]=4.0;
/*
        D[0][0]=3.0;
        D[0][1]=2.0;
        D[0][2]=1.1;
        D[1][0]=6.0;
        D[1][1]=2.0;
        D[1][2]=1.;
        D[2][0]=1.0;
        D[2][1]=4.0;
        D[2][2]=2.0;
*/
/* Store the matrix value for camparison later.
this is just to check the results, we don't need this
array for the program to work */
        for(m=0;m<=2;m++)
        {
         for(j=0;j<=2;j++)
          {
             C[m][j]=D[m][j];
          }
        }

/* Call a sub-function to calculate the LU decomposed matrix. Note that
we pass the two dimensional array [D] to the function and get it back */
        LU(D,n);

        printf(" \n");
        printf("The matrix LU decomposed \n");
        for(m=0;m<=2;m++)
        {
        printf(" %f  %f   %f \n",D[m][0],D[m][1],D[m][2]);
        }

/*  TO FIND THE INVERSE */

/* to find the inverse we solve [D][y]=[d] with only one element in
the [d] array put equal to one at a time */
```

```
        for(m=0;m<=2;m++)
        {
        d[0]=0.0;d[1]=0.0;d[2]=0.0;
        d[m]=1.0;
        for(i=0;i<=n;i++)
        { x=0.0;
          for(j=0;j<=i-1;j++)
          { x=x+D[i][j]*y[j];}
          y[i]=(d[i]-x);
        }

        for(i=n;i>=0;i--)
        { x=0.0;
          for(j=i+1;j<=n;j++)
          { x=x+D[i][j]*s[j][m];}
          s[i][m]=(y[i]-x)/D[i][i];
        }
        }

/* Print the inverse matrix */
        printf("The Inverse Matrix\n");
        for(m=0;m<=2;m++)
        {
          printf(" %f %f %f \n", s[m][0],s[m][1],s[m][2]);
        }
/* check that  the product of the matrix with its iverse results
is  indeed a unit matrix  */
        printf("The product\n");
        for(m=0;m<=2;m++)
        {
         for(j=0;j<=2;j++)
          {
           x=0.0;
           for(i=0;i<=2;i++)
            { x=x+C[m][i]*s[i][j]; }
          printf(" %d    %d    %f \n", m,j,x );
        }
        }
   }

/* The function that calcualtes the LU deomposed matrix.
Note that it receives the matrix as a two dimensional array
of pointers. Any change made to [D] here will also change its
value in the main function. So there is no need of an explicit
"return" statement and the function is of type "void". */

    void LU(float(*D)[3][3],int n)
   {
        int i,j,k,m,an,am;
        float x;
        printf("The matrix \n");
        for(j=0;j<=2;j++)
        {
        printf(" %f  %f  %f \n",(*D)[j][0],(*D)[j][1],(*D)[j][2]);
        }
        for(k=0;k<=n-1;k++)
        {
          for(j=k+1;j<=n;j++)
          {
            x=(*D)[j][k]/(*D)[k][k];
            for(i=k;i<=n;i++)
             {
               (*D)[j][i]=(*D)[j][i]-x*(*D)[k][i];
             }
             (*D)[j][k]=x;
```

```
    }
  }

  }
```