```
//Gauss Elimination with partial pivoting    J. M. Hill  3-16-12
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<iomanip.h> //for setw
#include<math.h> // for fabs()
void write_matrix(double a[][9],double b[],int n);
void get_matrix(double a[][9],double b[],int& n);
void write_vector(double x[9], int n);
int gauss_elim(double a[][9],double b[9],double x[],int n);
void solve(double a[][9],double b[9],double x[],int n);
void elim(double a[][9],double b[9],int i,int n);
void main()
{
      double a[9][9],b[9],x[9];
      int n,error_code;
      get_matrix(a,b,n);
      write_matrix(a,b,n);
      error_code = gauss_elim(a,b,x,n);
      if(error_code == 0)
            write_vector(x,n);
      else
            cout << "Answers are probably wrong";
}

int gauss_elim(double a[][9],double b[9],double x[],int n) //Gauss
elimination
{
      int k,i,ibig;
      double big,temp,aba;
      for(i=1; i<n; i++)
      {
            big = fabs(a[i][i]);
            ibig = i;
            for(k=i+1; k<=n; k++)          //locate largest pivot
            {
                  aba = fabs(a[k][i]);
                  if(aba > big)
                  {
                        big = aba;
                        ibig = k;
                  }
            }
            if(big < 1.e-30)               //check for zero pivot
            {
                  cout << "Matrix is singular" << endl;
                  return 1;
            }
            if(ibig != i)                  //check for necessity of row
switch
            {
                  for(k=i; k<=n; k++) //switch row with biggest pivot
into ith row
                  {
                        temp = a[i][k];
```

```
                        a[i][k] = a[ibig][k];
                        a[ibig][k] = temp;
                  }
                  temp = b[i];
                  b[i] = b[ibig];
                  b[ibig] = temp;
            }
            elim( a, b, i, n);
            write_matrix(a,b,n);
      }
      if(fabs(a[n][n]) < 1.e-30)          //check for singular matrix
      {
            cout << " matrix is singular " <<endl;
            return 1;
      }
    solve( a, b, x, n);
      return 0;
}



void solve(double a[][9],double b[9],double x[],int n)
{
      int k,j,np1,nmk;
      double q;
      np1 = n + 1;
      x[n] = b[n]/a[n][n];
      for(k=1; k<n; k++)
      {
            q = 0;
            nmk = n - k;
            for(j=1; j<=k; j++)
                  q = q + a[nmk][np1-j]*x[np1-j];
            x[nmk] = (b[nmk] - q)/a[nmk][nmk];
      }
}
void elim(double a[][9],double b[9],int i,int n)
{
      int k,j;
      double q;
      for(k=i+1; k<=n; k++)
      {
            q = -a[k][i]/a[i][i];
            a[k][i] = 0;
            b[k] = q*b[i] + b[k];
            for(j=i+1; j<=n; j++)
                  a[k][j] = q*a[i][j] + a[k][j];
      }
}
void write_vector(double x[9], int n)
{
      cout << " Solution vector\n";
      for(int i=1; i<=n; i++)
            cout << " x[" << i <<"] = " << x[i] << endl;
}
void write_matrix(double a[][9],double b[],int n)
```

```
{
	for(int i=1; i<=n; i++)
	{
		for(int j=1;j<=n;j++)
			cout << setw(9) <<a[i][j];
		cout << setw(9) << b[i] << endl;
	}
	cout << endl;
}
void get_matrix(double a[][9],double b[],int& n)
{

	ifstream in_file;
	char in_file_name[16];
	cout <<"Enter input file name (max of 15 characters): ";
	cin >> in_file_name;
	in_file.open(in_file_name,ios::nocreate);
	if(in_file.fail())
	{
		cout << "\nInput file opening failed\n";
		exit(1);
	}
	in_file >> n;
	for(int i=1; i<=n; i++)
	{
		for(int j=1; j<=n; j++)
			in_file >> a[i][j];
		in_file >> b[i];
	}
}
```