ASSIGNMENT NO:11 **DATE:27/03/2015**

PROGRAM STATEMENT: **Solve the system of equations, by Gauss-Seidel method,**

$$x_1 + x_2 + 4x_3 = 9$$
$$8x_1 - 3x_2 + 2x_3 = 20$$
$$4x_1 + 11x_2 - x_3 = 33$$

THEORY:

In certain cases, one can solve the system of equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad (1 \le i \le n)$$

by iteration. In Jacobi iteration, the kth approximation $x_j^{(k)}$ to the solution x_j of this equation is calculated as

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1,\, j\ne i}^{n} a_{ij}x_j^{(k)}}{a_{ii}} \qquad (1 \le i \le n),$$

and in Gauss-Seidel iteration one takes

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}} \qquad (1 \le i \le n);$$

that is, in Gauss-Seidel iteration, one makes use of $x_j^{(k+1)}$ as soon as it is available. One can use any starting values with these iterations; for example, $x_i^{(0)} = 0$ for each i is a reasonable choice.

Call an $n \times n$ matrix $A = (a_{ij})$ *row-diagonally dominant* if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\ne i}}^{n} |a_{ij}|$$

for each i with $1 \le i \le n$.

PROGRAM CODE:

```
//C Program to Solve a System of Equations by Gauss-Seidel Method
#include <stdio.h>
#include <math.h>
double mod(double x)
{
      if(x<0)
            return -x;
      else
            return x;
}
double error(int a)
{
      return 5*pow(10,-a-1);
}
int main()
{
      int n,i,j,k=0,pos;
      printf("Enter the number of Equations and the number of variables::");
      scanf("%d",&n);
      printf("Enter the coefficients of the equations\n");
```

```
double a[n][n+1],x[n],e,c,f;
for(i=0;i<n;i++)
{
      printf("Equation %d\n",i+1);
      x[i]=0;
      for(j=0;j<=n;j++)
      {
            scanf("%lf",&a[i][j]);
      }
}
printf("You need the answer correct upto how many decimal places? ::");
scanf("%lf",&e);
e=error(e);
for(i=0;i<n;)
{
      c=0;
      for(j=0;j<n;j++)
      {
            if(i!=j)
                  c=c+mod(a[i][j]);
      }
      if(mod(a[i][i])>c)
            i=i+1;
      else
      {
            f=mod(a[i][0]);
            for(j=0;j<n;j++)
                  if(mod(a[i][j])>f)
                  {
                        f=mod(a[i][j]);
                        pos=j;
                  }
            for(j=0;j<=n;j++)
            {
                  f=a[i][j];
                  a[i][j]=a[pos][j];
                  a[pos][j]=f;
            }
      }
}
printf("The rearranged eqns are::");
for(i=0;i<n;i++)
{
      printf("\nEquation %d\n",i+1);
      for(j=0;j<=n;j++)
      {
            printf("%lf\t",a[i][j]);
      }
}
printf("\nk\t");
for(i=0;i<n;i++)
      printf("x%d\t\t",i+1);
printf("\n%d\t",k++);
for(i=0;i<n;i++)
{
```

```
            printf("%lf\t",x[i]);
        }
        do
        {
            c=0;
            for(i=0;i<n;i++)
            {
                f=x[i];
                x[i]=0;
                for(j=0;j<n;j++)
                {
                    if(i!=j)
                        x[i]=x[i]-(x[j]*a[i][j]/a[i][i]);
                    else
                        x[i]=x[i]+(a[i][n]/a[i][i]);
                }
                if(mod(x[i]-f)<e)
                    c++;
            }
            printf("\n%d\t",k++);
            for(i=0;i<n;i++)
            {
                printf("%lf\t",x[i]);
            }
        }
        while(c!=n);
        printf("\nThe Solutions are::\n");
        for(i=0;i<n;i++)
            printf("x%d=%.2lf\n",i+1,x[i]);
        return 0;
}
```

OUTPUT:

```
Enter the number of Equations and the number of variables::3
Enter the coefficients of the equations
Equation 1
1 1 4 9
Equation 2
8 -3 2 20
Equation 3
4 11 -1 33
You need the answer correct upto how many decimal places? ::3
The rearranged eqns are::
Equation 1
8.000000   -3.000000  2.000000   20.000000
Equation 2
4.000000   11.000000  -1.000000  33.000000
Equation 3
1.000000   1.000000   4.000000   9.000000
k     x1          x2          x3
0     0.000000    0.000000    0.000000
1     2.500000    2.090909    1.102273
2     3.008523    2.006198    0.996320
3     3.003244    1.998486    0.999567
4     2.999540    2.000128    1.000083
```

```
5    3.000027   1.999998   0.999994
The Solutions are::
x1=3.00
x2=2.00
x3=1.00
```