Help

```
#include <stdlib.h>
#include"solversyslin.h"

void multiplytridiag(double **M, double *u, double *r,  int n)
{
  int i;

  r[0] = M[0][0] * u[0] + M[0][1] * u[1];

  for (i = 1; i < n - 1; i++)
    {
      r[i] = M[i][i] * u[i] + M[i][i - 1] * u[i - 1] + M[i][i + 1] * u[i + 1];
    }

  r[n - 1] = M[n - 1][n - 1] * u[n - 1] + M[n - 1][n - 2] * u[n - 2];

}

void tridiagsolve(double **M, double *u, double *r,  int n)
{
  int i, j;
  double *diagd;
  double *diagm;
  double *diagu;

  double *a;
  double *b;
  double *c;

  /* double bet, *gam; */
  diagd = malloc((n - 1) * sizeof(double));
  diagu = malloc((n - 1) * sizeof(double));
  diagm = malloc((n) * sizeof(double));

  a = malloc((n - 1) * sizeof(double));
  b = malloc((n) * sizeof(double));
  c = malloc((n - 1) * sizeof(double));

  diagm[0] = M[0][0];
```

```
for (j = 1; j < n; j++)
  {
    diagm[j] = M[j][j];
    diagd[j - 1] = M[j][j - 1];
    diagu[j - 1] = M[j - 1][j];
  }

a[0] = diagd[0] / diagm[0];
b[0] = diagm[0];
c[0] = diagu[0];
if (b[0] == 0)
  {
    printf("FATALE ERREUR, DIVION PAR ZERO LORS DE L'INVERSION DE LA MATRICE P
  }

for (i = 1; i < n - 1; i++)
  {
    b[i] = diagm[i] - a[i - 1] * c[i - 1];
    a[i] = diagd[i] / b[i];
    c[i] = diagu[i];
    if (b[i] == 0)
      {
        printf("FATALE ERREUR, DIVION PAR ZERO LORS DE L'INVERSION DE LA MATRI
      }

  }

b[n - 1] = diagm[n - 1] - a[n - 2] * c[n - 2];
if (b[n - 1] == 0)
  {
    printf("FATALE ERREUR, DIVION PAR ZERO LORS DE L'INVERSION DE LA MATRICE P
  }

u[0] = r[0];
for (i = 1; i < n; i++)
  {
    u[i] = r[i] - a[i - 1] * u[i - 1];
  }
u[n - 1] = u[n - 1] / b[n - 1];
for (i = n - 2; i >= 0; i--)
```

```
    {
      u[i] = (u[i] - c[i] * u[i + 1]) / b[i];
    }

  free(diagd);
  free(diagm);
  free(diagu);
  free(a);
  free(b);
  free(c);

}
```