```
/** lu.c: LU desompostion with Partial Pivoting **/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <pthread.h>

#define N 8
int n;

double A[N][N], L[N][N], U[N][N];
double B[N], b[N];
int    P[N];
double Y[N], X[N];

int print(char c, double x[N][N])
{
  int i, j;
  printf("------------- %c ----------------------\n", c);
  for(i=0; i<n; i++){
     for(j=0; j<n;j++)
        printf("%6.2f  ",  x[i][j]);
     printf("\n");
  }
}
int printV(char c, double x[N])
{
 int i;
 printf("--------- %c vector-----------\n",c);
 for (i=0; i<n; i++)
    printf("%6.2f ", x[i]);
 printf("\n");
}
int printP()
{
 int i;
 printf("--------- P vector-----------\n");
 for (i=0; i<n; i++)
    printf("%d ", P[i]);
 printf("\n");
}

// LU decomposition function
int lu()
{
```

```
int i, j, k, m, itemp;
double max;
double temp;
double si;

// LU decomposition loop
for (k=0; k<n; k++){
  max = 0;
  printf("partial pivoting by row %d\n", k);
  for (i=k; i<n; i++){
    if (max < fabs(A[i][k])){ // partial pivoting
        max = fabs(A[i][k]);
            j = i;
    }
  }
  if (max==0){
     printf("zero pivot: singular A matrix\n");
     exit(1);
  }

  // swap P[k] and P[j];
  itemp = P[k]; P[k] = P[j]; P[j] = itemp;

  // swap row A[k] and row A[j]
  printf("swap row %d and row %d of A\n", k, j);
  for (m=0; m<n; m++){
     temp = A[k][m]; A[k][m] = A[j][m]; A[j][m] = temp;
  }
  print('A', A);
  getchar();

  //swap L[k][0,k-2] and L[j][0,k-2]
  for (m=0; m<k-2; m++){
     temp = L[k][m]; L[k][m] = L[j][m]; L[j][m] = temp;
  }

  // compute L U entries
  U[k][k] = A[k][k];
  for (i=k+1; i<n; i++){
     L[i][k] = A[i][k] / U[k][k];
     U[k][i] = A[k][i];
  }

  // row reductions on A
```

```
   printf("row reducations of A by row %d\n", k);
   for (i=k+1; i<n; i++){
     for (m=k+1; m<n; m++){
        A[i][m] -=  L[i][k]*U[k][m];
     }
   }
   print('A', A); print('L', L); print('U', U); printP();
   getchar();
  }
}

int main(int argc, char *argv[])
{
  int i, j, k;
  double si;

  n = N;

  printf("main: initialize matrix A[N][N], B[N], L, U and P\n");
  for (i=0; i<n; i++)
    for (j=0; j<n; j++)
       A[i][j] = 1.0;

  for (i=0; i<n; i++)
    A[i][N-1-i] = 1.0*n;

  for (i=0; i<n; i++){
    B[i] = (n)*(n+1)/2 + (n-i)*(n-1);
  }

  for (i=0; i<n; i++){
    for (j=0; j<n; j++){
      U[i][j] = 0.0;
      L[i][j] = 0.0;
      if (i==j)
        L[i][j] = 1.0;
    }
  }

  for (i=0; i<n; i++){
    P[i] = i;
  }

  print('A', A);  print('L', L);  print('U', U);
```

```
  printV('B', B); printP();

  lu();

  // P L U are all computed; solve P*U*L*X = P*B

  // apply P to B to get b[ ]
  printV('B', B);
  for (i=0; i<N; i++){
    b[i] = B[ P[i] ];
  }
  printV('b', b);

  // solve L*Y = PB = b
  for (i = 0; i<n; i++){  // forwar substitution
     Y[i] = b[i];
     for (j=0; j<i; j++){
       Y[i] -= L[i][j]*Y[j];
     }
  }
  printV('Y', Y);

  // solve U*X=Y
  for (i = n-1; i >= 0; i--){ // backward substitution
     si = 0.0;
     for (j=i+1; j<n; j++)
        si += U[i][j]*X[j];
     X[i] = (Y[i] - si) / U[i][i];
  }
  printV('X', X);
}
```