```
/** C4.5.c: Gauss Elimination with Partial Pivoting **/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <pthread.h>

#define N 8
double A[N][N+1];
pthread_barrier_t barrier;

int getN()
{
  int a = 0;
  printf("input: %d", a);
  printf("How many threads?\n");
  scanf("%d", &a);
  return a;
}

int print_matrix()
{
  int i, j;
  printf("-----------------------------------\n");
  for(i=0; i<N; i++){
     for(j=0;j<N+1;j++)
       printf("%6.2f  ",  A[i][j]);
     printf("\n");
  }
}

void *ge(void *arg) // threads function: Gauss elimination
{
 int i, j, k, prow;
 int myid = (int)arg;
 double temp, factor;
 for(i=0; i<N-1; i++){
   if (i == myid){
     printf("partial pivoting by thread %d on row %d: ", myid, i);
     temp = 0.0; prow = i;
     for (j=i; j<=N; j++){
        if (fabs(A[j][i]) > temp){
           temp = fabs(A[j][i]);
                 prow = j;
               }
     }
     printf("pivot_row=%d  pivot=%6.2f\n", prow, A[prow][i]);
     if (prow != i){  // swap rows
               for (j=i; j<N+1; j++){
                 temp = A[i][j];
          A[i][j] = A[prow][j];
          A[prow][j] = temp;
        }
            }
   }
   // wait for partial pivoting done
   pthread_barrier_wait(&barrier);
   for(j=i+1; j<N; j++){
     if (j == myid){
```

```
        printf("thread %d do row %d\n", myid, j);
              factor = A[j][i]/A[i][i];
        for (k=i+1; k<=N; k++)
           A[j][k] -= A[i][k]*factor;
              A[j][i] = 0.0;
      }
    }
    // wait for current row reductions to finish
    pthread_barrier_wait(&barrier);
    if (i == myid)
      print_matrix();
  }
}


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

  int n = getN();
  pthread_t threads[n];

  printf("main: initialize matrix A[N][N+1] as [A|B]\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-i-1] = 1.0*n;
  for (i=0; i<n; i++){
     A[i][n] = (n*(n+1))/2 + (n-i)*(n - 1);
  }
  print_matrix();  // show initial matrix [A|B]

  pthread_barrier_init(&barrier, NULL, n); // set up barrier

  printf("main: create N=%d working threads\n", n);
  for (i=0; i<n; i++){
    pthread_create(&threads[i], NULL, ge, (void *)i);
  }
  printf("main: wait for all %d working threads to join\n", n);
  for (i=0; i<n; i++){
      pthread_join(threads[i], NULL);
  }
  printf("main: back substitution : ");
  for (i=n-1; i>=0; i--){
     sum = 0.0;
     for (j=i+1; j<n; j++)
        sum += A[i][j]*A[j][n];
     A[i][n] = (A[i][n]- sum)/A[i][i];
  }
  // print solution
  printf("The solution is :\n");
  for(i=0; i<n; i++){
              printf("%6.2f  ", A[i][n]);
  }
  printf("\n");
}
```