```c
/* ==================================================================
 *
 * CS 566 - Assignment 03
 * Camillo Lugaresi, Cosmin Stroe
 *
 * Header file for common.c
 *
 * ================================================================== */

/* CELL(matrix,r,c) - for matrices in row,column order */
#define CELL(m,r,c) (((m)->data)[((m)->n)*(r) + (c)])

/* BLK_CELL */
#define BLK_CELL(data,blksz,b,r,c) ((data)[(blksz*blksz)*(b) + (blksz)*(r) + (c)])

struct matrix {
        int n;
        int *data;
};

struct matrix2 {
        int n;
        int stride;             // width of "mother" matrix
        int *data;
};

/* CELL2(matrix2,r,c) - for submatrices */
#define CELL2(m2,r,c) (((m2)->data)[((m2)->stride)*(r) + (c)])

struct input_params {
        int print;
        int lu2d;
        int strassen;

        int mode;
        union {
                double prob[2];
                int pattern[4];
        } u;
};

void alloc_matrix(struct matrix *m, int n);
void naive_matrix_mult_add(struct matrix *C, struct matrix *A, struct matrix *B);
int parse_args(int argc, char *argv[], int *n, int *k, struct input_params *m_in);
void fill_matrix(struct matrix *m, struct input_params *m_in);
void print_matrix(struct matrix *m);
void make_matrix2(struct matrix *m, struct matrix2 *m2);
void make_submatrix(struct matrix2 *m, struct matrix2 *sub, int r, int c, int blksz);
void copy_block(int blksz, struct matrix *sm, int sr, int sc,
                                                   struct matrix *dm, int dr, int dc);
void matrix_to_blocks(struct matrix *m, int *blockdata, int blksz, int column_first);
void blocks_to_matrix(struct matrix *m, int *blockdata, int blksz, int column_first);
void matrix_to_rowblocks_cyclic(struct matrix *m, int *blockdata, int blksz);
int intlog2( int n );
int intpow2( int power );
int count_swaps(int *reorder_all, int n);
```