

```

/*
 * symmetric_group.c
 *
 * At the moment this file contains only the function
 *
 * Boolean is_group_S5(SymmetryGroup *the_group);
 *
 * which checks whether the_group is the symmetric group S5.
 * Eventually I will write a more general set of functions
 * to test for other symmetric and alternating groups.
 */

#include "kernel.h"

Boolean is_group_S5(
    SymmetryGroup *the_group)
{
    /*
     * Table 5 on page 137 of Coxeter & Moser's book "Generators and
     * relations for discrete groups" provides the following presentation
     * for the symmetric group S5, which they in turn credit to page 125 of
     * W. Burnside's article "Note on the symmetric group", Proc. London
     * Math. Soc. (1), vol. 28 (1897) 119-129.
     *
     *  $\{A, B \mid A^2 = B^5 = (AB)^4 = (A(B^{-1})AB)^3 = 1\}$ 
     *
     * where  $A = \begin{pmatrix} 1 & 2 \end{pmatrix}$  and  $B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \end{pmatrix}$ .
     *
     * Coxeter & Moser adopt the convention of reading products of
     * group elements left-to-right, whereas we read them right-to-left,
     * so we should translate the above presentation as
     *
     *  $\{A, B \mid A^2 = B^5 = (BA)^4 = (BA(B^{-1})A)^3 = 1\}$ 
     *
     * In this case, though, it doesn't matter, because the relations
     * in the latter presentation are conjugate to the corresponding
     * relations in the former.
     */

    int a,
        b,
        ab,
        aB,
        aBab,
        possible_generators[2];

    /*
     * If the order of the_group isn't 120, it can't possibly be S5.
     */
    if (the_group->order != 120)
        return FALSE;

    /*
     * Consider all possible images of the generator a in the_group.
     */
    for (a = 0; a < the_group->order; a++)
    {
        /*
         * If a does not have order 2, ignore it and move on.
         */
        if (the_group->order_of_element[a] != 2)
            continue;

        /*
         * Consider all possible images of the generator b in the_group.
         */
        for (b = 0; b < the_group->order; b++)
        {
            /*
             * If b does not have order 5, ignore it and move on.
             */
            if (the_group->order_of_element[b] != 5)
                continue;

```

```
/*
 * Compute  $ab$  and  $a(b^{-1})ab$ .
 */
ab      = the_group->product[a][b];
aB      = the_group->product[a][the_group->inverse[b]];
aBab    = the_group->product[aB][ab];

/*
 * If  $ab$  does not have order 4, give up and move on.
 */
if (the_group->order_of_element[ab] != 4)
    continue;

/*
 * If  $a(b^{-1})ab$  does not have order 3, give up and move on.
 */
if (the_group->order_of_element[aBab] != 3)
    continue;

/*
 * At this point we know  $a^2 = b^5 = (ab)^4 = (a(b^{-1})ab)^3 = 1$ .
 * We have a homomorphism from  $S_5$  to the_group. It will be an
 * isomorphism iff  $a$  and  $b$  generate the_group.
 */

/*
 * Write  $a$  and  $b$  into an array . . .
 */
possible_generators[0] = a;
possible_generators[1] = b;

/*
 * . . . and pass the array to the function which checks
 * whether they generate the group.
 */
if (elements_generate_group(the_group, 2, possible_generators) == TRUE)
    return TRUE;

/*
 * If  $a$  and  $b$  failed to generate the_group, we continue on
 * with the hope that some other choice of  $a$  and  $b$  will work.
 */
}

return FALSE;
}
```