```c
/*
 * ECE220 Spring 2018 (update from Fall 2005)
 *
 * Program name: unique_count.c, a unique line counting program
 *
 * Description: This program reads lines from stdin, merges identical
 *              lines, and prints each line with a number prefix
 *              indicating how many times the same line appeared
 *              consecutively in the input.
 */

#include <stdint.h>       /* Include C's standard integer header file. */
#include <stdio.h>        /* Include C's standard I/O header file.  */
#include <string.h>       /* Include C's string library.               */

static const int32_t max_word_len = 500;  /* limit on word length */

/* My favorite exit condition definitions. */
enum {
    EXIT_SUCCEED  = 0,
    EXIT_FAIL     = 1,
    EXIT_BAD_ARGS = 2,
    EXIT_PANIC    = 3
};

/*
 * Function: main
 * Description: read lines from stdin, merge duplicate consecutive lines,
 *              and print lines prefixed by their multiplicities in the
 *              input (consecutive counts only; appearance elsewhere is
 *              ignored)
 * Parameters: argc -- the number of arguments, including the executable name
 *             argv -- an array of strings containing each argument
 *             argc must equal 1; no additional argument are allowed
 * Return Value: EXIT_SUCCEED for success
 *               EXIT_FAIL if the input contains no lines
 *               EXIT_BAD_ARGS if the wrong number of arguments are given
 */

int
main (int argc, char* argv[])
{
    char    buf1[max_word_len + 1];     /* a line                    */
    char    buf2[max_word_len + 1];     /* a second line             */
    char*   last_line;                  /* points to last line       */
    char*   cur_line;                   /* points to current line    */
    char*   tmp;                        /* a temporary for swapping  */
    int32_t count;                      /* multiplicity of last_line */

    /* Program must receive exactly one argument. */
    if (1 != argc) {
        /* Print an error message.  argv[0] is the executable name. */
        fprintf (stderr, "syntax: %s\n", argv[0]);
        return EXIT_BAD_ARGS;
    }

    /* Read the first line. */
    if (NULL == fgets (buf1, max_word_len + 1, stdin)) {
        fputs ("Could not read any lines!\n", stderr);
        return EXIT_FAIL;
    }

    /* Initialize the double buffering scheme based on the first line's
       residing in buf1. */
    last_line = buf1;
    count     = 1;
    cur_line  = buf2;

    /* Read lines until we find the end of the input. */
    while (NULL != fgets (cur_line, max_word_len + 1, stdin)) {

        /* Check for duplication. */
        if (0 == strcmp (cur_line, last_line)) {
            count++;
            continue;
        }

        /* Print last line (it already includes a carriage return). */
        printf ("%5d %s", count, last_line);

        /* Switch buffering for lines, and reset count. */
        tmp = cur_line;
        cur_line = last_line;
        last_line = tmp;
        count = 1;
    }

    /* Print final line (it already includes a carriage return). */
    printf ("%5d %s", count, last_line);

    /* Program finished successfully. */
    return EXIT_SUCCEED;
}
```