```c
// ECE 312 LEC A1
// Assignment 6, Fall 2014
// Author: Mohammad Sirajee
// Student ID: 1255986
// Percent Original: 100% but I received helpful advice and hints on how to
//                    approach the problem from the people listed in 'Other
//                    Contributors'
// Other Contributors: The TA (assignment marker), Joao Paulo Figueiredo
//                       (Student ID: 1417859)

#include <stdio.h> // For standard C file input/output
#define PAUSE -1 // Symbolize pause in user input

/* Implementation of the multi-tap interface of a classic Nokia handset.
   In this implementation, two static integer variables - previous_key and
   count - are used to keep track of the sequence of input keys and their
   count in order to determine when they change and thus, should be
   converted to the respective characters and sent to the output file
   stream.

   Inputs:
   fout - pointer to the file stream where the characters corresponding to
          the sequence of key taps should be sent as output
   key  - the current key press from the input file stream sent to the
          multiTap() function to be processed

   Output:
   Nothing is returned from the function. Only a character corresponding to
   the sequence of the same input key is sent as an output to the file
   stream fout
*/
void multiTap(FILE *fout, int key) {

    //The data given in the Technical Specifications of the assignment pdf
    //are encoded below in character arrays corresponding to the keypad key
    const char zero[] = {' ', '0'};
    const char one[] = { ',', '.', '\'', '?', '!', '"', '1', '-', '(', ')',
                        '/', ':', '_' };
    const char two[] = {'A', 'B', 'C', '2'};
    const char three[] = {'D', 'E', 'F', '3'};
    const char four[] = {'G', 'H', 'I', '4'};
    const char five[] = {'J', 'K', 'L', '5'};
    const char six[] = {'M', 'N', 'O', '6'};
    const char seven[] = {'P', 'Q', 'R', 'S', '7'};
    const char eight[] = {'T', 'U', 'V', '8'};
    const char nine[] = {'W', 'X', 'Y', 'Z', '9'};

    //Static integer variables to hold the value of the previous key and
    //its total count
    static int previous_key = key;
    static int count = 0;

    //If the previous key encountered was PAUSE, then make previous_key
    //equal to the current key input to the function
    if (previous_key == PAUSE) {
        previous_key = key;
    }

    //If the previous key equals the current key and if neither of the keys
    //represent PAUSE, increment the count of the number of times the same
    //key has been encountered/inputted
    if (previous_key == key && key != PAUSE) {
        count++;
    }
    //if the previous key does not equal the current key, determine what
    //character (from the total count of the previous key) the previous key
    //represents and output it to the file stream fout
    else if (previous_key != key) {
        char character;
        switch (previous_key)
        {
        case 0: character = zero[(count-1) % sizeof(zero)];
            break;
        case 1: character = one[(count - 1) % sizeof(one)];
            break;
        case 2: character = two[(count - 1) % sizeof(two)];
            break;
        case 3: character = three[(count - 1) % sizeof(three)];
            break;
        case 4: character = four[(count - 1) % sizeof(four)];
            break;
        case 5: character = five[(count - 1) % sizeof(five)];
            break;
        case 6: character = six[(count - 1) % sizeof(six)];
            break;
        case 7: character = seven[(count - 1) % sizeof(seven)];
            break;
        case 8: character = eight[(count - 1) % sizeof(eight)];
            break;
        case 9: character = nine[(count - 1) % sizeof(nine)];
            break;
        }
        fputc(character, fout);

        //if the current key is not PAUSE, make it equal to the previous
        //key and increment its count by 1 by re-initializing count to 1
        if (key != PAUSE) {
            previous_key = key;
            count = 1;
        }
        //if the current key is PAUSE, make it equal to the previous key
        //and re-initialize count to 0
        else {
            previous_key = PAUSE;
            count = 0;
        }
    }

    /*
    static int count = 0; // State variable
    if (key != PAUSE) fputc(key + '0', fout);
    else count++; // Count pauses
    */
}

```