The Luhn algorithm will detect any single-digit error, as well as almost all transpositions of adjacent digits. It will not, however, detect transposition of the two-digit sequence 09 to 90 (or vice versa). Other, more complex check-digit algorithms (such as the Verhoeff algorithm) can detect more transcription errors. The Luhn mod N algorithm is an extension that supports non-numerical strings.

The formula verifies a number against its included check digit, which is usually appended to a partial account number to generate the full account number. This account number must pass the following test:

1. Starting with the rightmost digit (which is the check digit) and moving left, double the value of every second digit. For any digits that thus become 10 or more, add their digits together as if casting out nines. For example, 1111 becomes 2121, while 8763 becomes 7733 (from 2×6=12 • 1+2=3 and 2×8=16 • 1+6=7).

2. Add all these digits together. For example, if 1111 becomes 2121, then 2+1+2+1 is 6; and 8763 becomes 7733, so 7+7+3+3 is 20.

3. If the total ends in 0 (put another way, if the total modulus 10 is congruent to 0), then the number is valid according to the Luhn formula; else it is not valid. So, 1111 is not valid (as shown above, it comes out to 6), while 8763 is valid (as shown above, it comes out to 20).
//LUHN algorithm to generate a check digit for a string of numbers that are randomly selected for demonstration. The array returned is the random string and the check digit in the last location.

int[] CreateNumber(int length) {
    Random random = new Random();
    int[] digits = new int[length];
    for (int i = 0; i < length - 1; i++) {
        digits[i] = random.Next(10);
    }
    int sum = 0;
    bool alt = true;
    for (int i = length - 2; i >= 0; i--) {
        if (alt) {
            int temp = digits[i];
            temp *= 2;
            if (temp > 9) {
                temp -= 9;
            }
            sum += temp;
        } else {
            sum += digits[i];
        }
        alt = !alt;
    }
    int modulo = sum % 10;
    if (modulo > 0) {
        digits[length - 1] = 10 - modulo;
    }
    return digits;
}
/LUHN algorithm to verify a check digit for a string
//of numbers that are submitted to the function. returns
//true is it is a valid LUHN sequence
//
//446-667-651 is a valid number
//
bool CheckNumber(int[] digits){
    int sum = 0;
    bool alt = false;
    for(int i = digits.Length - 1; i >= 0; i--)
    {
        if(alt)
        {
            digits[i] *= 2;
            if(digits[i] > 9)
            {
                digits[i] -= 9; // equivalent to adding the digits of value
            }
        }
        sum += digits[i];
        alt = !alt;
    }
    return sum % 10 == 0;
}