Introduction to Computing Technology

Microproject – 2

Deadline: 23 Oct 2020, 2355 hrs

Assume you are starting a website with a login/password signup system. You need to make sure that passwords are strong and valid. Unix systems allowed only passwords of 8 characters or less until the 90s. You, however, decide that valid passwords should have <#> number of characters. A strong valid password should follow certain conditions. Refer to attached appendix for specific conditions assigned to your team.

Everything users enter from a keyboard is stored as an ASCII character. Please read up about ASCII character and understand the character table. For each valid character, you will find its decimal value, hexadecimal (or base 16) value, and its binary value. [Needless to say, you can convert each to the other!] Each ASCII character is 8 bits long as can be seen in the table. Let us refer to the bits from the left as a, b, c, d, e, f, g, and h. (That is, bit a is the left-most bit and bit h is the rightmost for each character.)

You have to design a *valid password circuit* that takes the password of specific number of characters as assigned to your team as input and gives out a 1 if the password is a valid string and 0 if it is not.

Hint 1: Say your team has been assigned a 5-character length. Design 5 independent **detectors** for the 4 conditions given (you make digital circuits to **recognize** characters like capital letters, small letters, numerals, special character, and a valid character). Combine the outputs of these detectors logically to give the final output that indicates if the given password is valid or not.

Use AND/OR/NOT/NAND/NOR gates only to design each of these detectors. You are free to use gates with 8 or fewer inputs (that is, you can use 5-input NAND or 7-input OR and others!)

Hint 2: You can follow modular design if you like. Design the circuit that takes one character as input and givens all necessary outputs for that character. You may then treat this as a larger **PasswordCharacter** block with clearly indicated inputs and outputs. You can use multiple such blocks and combine all of their outputs to give the final **PasswordValid** logic output.

Hint 3: There is one more hint inside one of the hints. See if you can find it.