Assignment 7, 2020-11-05,

12 minutes

Question 1 ("Folding HashCodes" for Strings).
You have 3 different string keys (representing for example car registration numbers): `K`*abc*, `K`*bca*, `K`*bc*, and you want to compute a hash function using the "mod two folding" (also known as XOR), and will insert your results in a hash table H with 7 slots: $H[0], \ldots, H[6]$.
(Here a, b, c denote the last 3 digits from your Student ID. For example, if $abc = 789$, then the strings to encode are `K789`, `K897`, `K89`. They all start with the same ASCII letter `K`.
(A) Compute the (uncompressed) hashcode values for all the 3 string values. Your hashcode function $h_1(s)$ is defined as follows:

$$h_1(s) = \bigoplus_{i=0}^{L-1} \text{ord}(c[i]) = c[0] \oplus \ldots \oplus c[L-1].$$

Here L is the length of the string s. By $c[i]$ we denote the ith character of the input string s ($i = 0, 1, \ldots, L-1$). All the hash values computed by this function are 1 byte long, they are integers in $[0; 255]$.
Note. By $\text{ord}(c)$ we denote the ASCII code of some character `c`; these ASCII codes are also integers in $[0; 255]$. (See the ASCII codes: `http://www.asciitable.com/`.)

(B) Represent the bits of all four characters in `K`*abc* (one capital letter "K" and the three digits from your ID). Check the computation of $h_1($`K`*abc*$)$ by writing these bits aligned in columns and add them up modulo 2. (To see how a character is represented as a sequence of bits, use ASCII code in hexadecimal. For example, character "A" has hex code `0x41`, i.e. it is represented by these eight bits: `01000001` (since hex "4" converts into `0100`, but "1" converts into `0001`).

(C) Compute the compressed hash values for the same 3 strings modulo 7. Namely, the compressed hash value is

$$h_2(h_1(s)) = h_1(s) \bmod 7.$$

(D) Draw the four string objects in a hashtable H with 7 cells $(H[0], \ldots, H[6])$. Are there any collisions?

Note. Here is the pseudocode of the abovementioned string hashing function $h_1(s)$ (in Python):

```
def h1(s):
    h = 0
    for c in s:
        h = h ^ ord(c)
    return h
```