Jose Lorenzo M. Cansana
CSALGCM S15

Problem B: Digit Sequence (Hard Edition)
Status: Not Submitted
Difficulty Rating: 3
Solution Sketch/Write-Up/Narrative:

The problem asks what digit exists given an N-th index in the array of increasing numbers when one index in the array can only occupy 1 digit. To illustrate, the array starts [1, 2, … , 1, 0, 1, 1, 1, 2, …] and so on.

Therefore to create our algorithm, we must consider the number of digits and where they start in the array. The formula to get the starting index is:

$$Number\ of\ N-digit\ Numbers\ *\ N\ +\ N-1\ Start\ Index$$

where N is the number of digits

Given an array (0...n), we can tabulate the start index of each unique number of digits

NUMBER OF DIGITS	INDEX
1	0
2	10
3	188
...	...

Once we calculate the starting index of the unique number of digits, given the input we can identify in which 2 starting indices our input lies between. From this we know how many digits the desired number that exists in the inputted index. Now to find the specific digit within the number, we play with division and modulo. To find the specific number, we use the formula $((inputIndex\ -\ lowerBoundIndex)\ /\ N)\ +\ startingNthNumber$ where N is the number of digits. For example if inputted index is 200 we know that it refers to 104 because $((200\ -\ 188)\ /\ 3)\ +\ 100\ =\ 104$. Now to find the specific digit we are finding from the number, we simply variate the formula for modulo, $(inputIndex\ -\ lowerBoundIndex\ \%\ N)$ and the result will give us the specific digit, and ultimately the result of the algorithm.

Problem F: Equivalent Strings
Status: Wrong Answer
Difficulty Rating: 5
Solution Sketch/Write-Up/Narrative:

Equivalence is said when: two strings are equal, or halves of both strings are equal to half of the other string. From the halving algorithm, I understood the divide and conquer nature of the problem. First the algorithm must check if both inputted strings are literally equal with each other, and if they are, return that they are equivalent, else, we proceed with the algorithm.

The nature of one string's halves are equal to the other half of the other string is recursive meaning that even the half of the half has equivalence with the other string's half of the half. Therefore, we must implement a recursive function that continues to half and computes for the Least Equivalent String with a length of 1. At each call and at the end of the recursive calls (where all cases are base cases), we check for equivalence. To check for equivalence, we compare one string from String A, named A1 and if they are equivalent to String B's B1 and B2. Similarly, we check for A2 to both strings once more. We check equivalences with respect to it's ASCII value and its symbol as well. Do this for all instance and all must return true to say that both strings are equivalent, and if at least one returns false, we now that the string is not equivalent.