CS 454 Theory of Computation Spring 2018

Project # 1 *Due: Feb 25, 2018 (mid-night)*

The project will be done in teams of size no larger than three. Each team will submit one solution – as a single program that when run asks if the user wants to solve Problem 1 or 2, then takes input for that problem (in case of problem 2, state clearly how the digits must be entered) and repeat the process until the user quits. The project will be submitted through moodle.

PROBLEM 1:

Write a function *count* that computes the number of strings *w* of length n over {*a, b, c*} with the following property: In any substring of length 4 of *w*, all three letters *a*, *b* and *c* occur. For example, strings like *abbcaabca* are included in the count but a string like *aabbcabac* is not included since *aabb* (the first four letters) does not have a *c*.

The idea is to create a DFA M for the language $L = \{ w \mid$ in any substring of length 4 of *w*, all three letters *a*, *b* and *c* occur$\}$. Then create the transition matrix of M, and use the matrix formula presented in class to implement a function that takes as input an integer *n*, and outputs the number of strings of length *n* accepted by the DFA *M*.

When your main function runs, it will ask for an integer input *n*, and output the number of strings of length *n* with the specified property. The range of *n* will be between 1 and 300. The answer should be exact, not a floating-point approximation so you should use a language that supports unlimited precision arithmetic like Java or Python or a library like GMP (in case of C++).

Some test cases:

Test case 1:
n = 137
number of strings of length *n* is : 611926697614991224161489884186654673 6

Test case 2:
n = 100

number of strings of length *n* is : 98780220763817840013188490 0

PROBLEM 2:

Write a function *MinString* that takes as input a DFA and outputs a string w of shortest length (lexicographically first in case of more than one string) accepted by the DFA. The algorithm Breadth-First Search for solving this problem will be presented in class. Use this function to write another function *FindInt* that takes as input a positive integer x, and a subset S of {0, 1, 2, …, 9}

and outputs the smallest positive integer y that is a multiple of x, and has only digits (in decimal) from the set S.

Some test cases:

Input: k = 26147, Digits permitted: 1 3
Shortest multiple of k using digits {1, 3}: 1113313113

Input: k = 198217, Digits permitted: 1
Shortest multiple of k using digits {1}: integer containing 10962 ones (Your output should be a string of this many 1's.)