CPSC 413 Fall 2019 — Optional Bonus Assignment 3

Due no later than Sunday, November 17, 2019, 13:00

Assignment 3 is worth 10 points. The bonus programming problem is worth **7 bonus points**. The total possible number of points is thus 17 points.

Input format

The input file contains a number of lines. The first line contains an integer n ($2 \leq n \leq 10$) which is the size of the alphabet. The alphabet is $\{0, 1, \ldots, n-1\}$. Note that the alphabet starts from 0 and ends at $n-1$. The second line contains a target symbol t ($0 \leq t \leq n-1$) which is the symbol we want to generate. Then follows n lines, each of which contains n numbers, separated by a single space. These n^2 numbers constitute the addition table. E.g., if $(2 \oplus 4) = 7$, then the fifth number in the third line is seven. If $(0 \oplus 1) = 3$, then the second number in the first line is three. If $(4 \oplus 0) = 0$, then the first number in the fifth line is zero. The last line is a string s over the alphabet. The string s is of length between 2 and 199 (and up to length 999 when possible).

Example of possible code

Here is some rudimentary code that may help in getting you started. I chose to code my solution in Perl which is a versatile scripting language I may use for quick short scripts. I run my script from the terminal with the command

```
./addition.pl addition_sample1.in
```

Here are snippets of my file which is called `addition.pl`.

```
#! /usr/bin/perl
#
# Example of use: running the command "addition.pl filename.in"
# Produces one of two output to stdout:
# "YES, target symbol can be formed" or
# "NO, target symbol can NOT be formed"

use strict;
```

```
use warnings;

use Time::HiRes qw( gettimeofday tv_interval );

MAIN : {

    # Verify there is exactly one argument
    (@ARGV == 1) or (die "Usage is: addition.pl filename.in\n");
    my $args = $#ARGV + 1;
    my $inputfilename = $ARGV[0];

    open(my $infile, "<", $inputfilename)
        or (die "Error: Could not read from file $inputfilename\n");

    # Read alphabet size
    my $line = readline($infile);
    chomp($line);
    my $alphabetsize = $line;

    # Read target symbol
    $line = readline($infile);
    chomp($line);
    my $target = $line;

    # Read addition table
    my @addtable;
    for (my $i=0; $i<$alphabetsize; $i++) {
        $line = readline($infile);
        chomp($line);
        my @thisline = split(' ', $line);
        for (my $j=0; $j<$alphabetsize; $j++) {
            $addtable[$i][$j] = $thisline[$j];
        }
    }

    # Read input string
    my $inputstring = readline($infile);
    chomp($inputstring);
```

```
    my @inputarray = split(//, $inputstring);
    my $n = scalar @inputarray;

    # Close input file
    close($infile);

    # Main part goes here
}
```

Collaboration and plagiarism

Note that no collaborations are allowed for the bonus programming problem.

For input and output questions, however, collaborations via D2L are both permitted and encouraged. Above, you can find PERL code for reading the input file. If you have code in a different language for reading the input file, I would be most thankful if you can email the code to me and I will post it to D2L for general sharing.

Requirements to your program

Your program must: run in polynomial-time, be well-written, not generate compile errors, not cause runtime errors, not have presentation errors, and **produce the correct output on every valid input**. No time-limits in terms of seconds will be given, but your program must run in polynomial time.

Your program must consist of a single file. Your file must be given the filename `<last_name><first_name><your_UofC_ID_Number>`, where you replace each of the three tokens with your identification. The extension (i.e. suffix) of your filename must uniquely identify your programming language of choice. A correctly formatted file name is e.g. `SmithJohnJoe12345678.py`. Include your UofC ID number in the first line of your submitted program. You may submit your program in C, C++, Java, Python, or Perl. Basic standard libraries are fine. Do **not** use any fancy, non-standard libraries. If your program does not compile on our vanilla systems, your submission will be rejected.

Your program must take one argument which is the name of the file containing the input. An example is `SmithJohnJoe12345678.py addition_sample1.in`. Your program must output its result to stdout. The output must consists of a single line of text, which must be either `YES, target symbol can be formed` or `NO, target symbol can NOT be formed`, followed by a line break. Your program must **not** generate any other text as output.

Submissions for the bonus problem will be compiled and run on our own test data. The text generated by your program will be compared against our own test output data, and it will be compared by an automated pattern matching algorithm. Your output must be exactly identical to our output. Do not include any extra spaces, line breaks, tabs, comments, special symbols, or any other additional characters in your output. Our pattern matching algorithm attempts to remove extra white space, but will otherwise declare your output wrong if it does not match our output file exactly.

You can find sample input and output files on D2L. You can use the Unix command `diff` to verify that your program's output is formatted correctly on the sample input provided.

Submission to bonus points

You may submit your bonus points assignment electronically through Desire2Learn on or before Sunday, November 17, 2019, 13:00. Please note that the deadline for the bonus points is Sunday at 13:00. No printed material needs to be submitted for the bonus points. Note that submissions are timestamped.

Note that no collaborations are allowed for the bonus programming problem. You are however welcome to post to the Desire2Learn moderated Assignment forum, and I will reply to your post as soon as possible.