ICS 46 Lecture B Fall 2019 Assignment 1: Summation Puzzles

Due Tuesday, October 15, 11:59 PM.
Note that this is one day later than the syllabus originally stated.

Introduction

In lecture week one, we saw that recursion can make some seemingly laborious problem solving into a straight-forward exercise. In project 0, we saw that we can check if a solution is valid to summation puzzles. In this assignment, we're going to solve summation puzzles.

In a summation puzzle, you are given three strings of the form POT + PAN = BIB. Typically each is a word, often with a theme to the three chosen. Your goal is to assign a *distinct* digit to each letter in the equation in order to make the resulting true. For example, if the puzzle is POT + PAN = BIB, the mapping P:2, O:3, T:1, A:7, N:4, B:5, I:0 will solve this, as 231 + 274 = 505.

Getting Started

Before you begin work on this project, there are a couple of chores you'll need to complete on your ICS 46 VM to get it set up to proceed.

Refreshing your ICS 46 VM environment

Even if you previously downloaded your ICS 46 VM, you will probably need to refresh its environment before proceeding with this project. Log into your VM and issue the command ics46 version to see what version of the ICS 46 environment you currently have stored on your VM. Note, in particular, the timestamp; if you see a version with a timestamp older than the one listed below, you'll want to refresh your environment by running the command ics46 refresh to download the latest one before you proceed with this project.

If you're unable to get outgoing network access to work on the ICS 46 VM — something that afflicts a handful of students each quarter — then the ics46 refresh command won't work, but an alternative approach is to download the latest environment from the link below, then to upload the file on to your ICS 46 VM using SCP. (See the Project #0 write-up for more details on using SCP.) Once the file is on your VM, you can run the command **ics46b refresh_local NAME_OF_ENVIRONMENT_FILE**, replacing **NAME_OF_ENVIRONMENT_FILE** with the name of the file you uploaded; note that you'd need to be in the same directory where the file is when you run the command.

The file is linked from the "public" ICS 46 page; click this link and enjoy the amazing web design skill that put it together: https://www.ics.uci.edu/~mikes/ics46/

Creating your project directory on your ICS 46 VM

A project template has been created specifically for this project, containing a similar structure to the basic template you saw in Project #0.

Decide on a name for your project directory, then issue the command **ics46b start YOUR_CHOSEN_PROJECT_NAME project1** to create your new project directory using the project1 template. (For example, if you wanted to call your project directory proj1, you would issue the command ics46 start proj1 project1 to create it.) Now you're ready to proceed!

Reviewing related material

I encourage you to read your textbook; in the second edition, section 3.5 deals with recursion. Your book is good at getting to the point, so this should not be a long read. Furthermore, you should look at your notes from lecture on Friday of week one when we discussed the *n* Queens problem and solving it via recursion.

Requirements

You are required to implement the function puzzleSolver in proj1.cpp. This function should return true if, and only if, the puzzle is solvable: that is, if there is a mapping of the letters that appear in the three strings to *distinct* digits such that the sum of the first two is the third. No string will have a value larger than 4,294,967,295 in its correct substitution, nor will the addition have any integer-overflow to check for. If you do not know what integer overflow is, you do not need to check during this assignment (although it's worth knowing in general).

For this project, you have a few requirements:

- You must implement the function puzzleSolver in proj1.cpp**.** You may assume it is called with three valid non-empty strings as parameters and with an otherwise empty map. The strings will always consist only of all-capital letters.
- Your solution must explicitly use recursion. You may not solve this by using a function like std::next_permutation (from <algorithm>) to enumerate possibilities.
- The function must return a boolean indicating whether or not the puzzle *has* a solution.
    - If the puzzle *does not* have a solution, your function should return false.
    - If the puzzle *does* have a solution your function should return true **and** have the map<char, unsigned> parameter containing the mapping needing to verify. That is, the four parameters to the `puzzleSolver` function need to be such that a correct solution to project 0 would return true with those parameters.

You *may* use standard libraries as appropriate, unless there is one that makes solving this problem trivial. I am unaware of any such part of the library.

You are **explicitly permitted** to use stl::set, stl::queue, and stl::stack if you so choose. You are pretty much required to use stl::map. You are welcome to ask anything you want about these libraries, or to look up material about them online. Information about how to use an explicitly-permitted library may always be shared among classmates, but refrain from telling one another how you solved a problem in the assignment with them. For example, answering "how do I check if an element is in a std::set?" is great and encouraged, while answering "what did you use std::set for in your project?" is not.

A good reference for the STL container classes (such as those listed above, including stl::map) is http://www.cplusplus.com/reference/map/map/ .

If you would like to reuse *your code* (not that of someone else) from project 0 for part of this project, you are welcome to do so.

Deliverables

After using the gather script in your project directory to gather up your C++ source and header files into a single **project1.tar.gz** file (as you did in Project #0), submit that file (and only that file) to Checkmate. Refer back to Project #0 if you need instructions on how to do that. This time, it should give you the correct file name, insert innocent-looking face emoji here.

You will submit your project via Checkmate. Keep in mind that that you're responsible for submitting the version of the project that you want graded. We won't regrade a project simply because you submitted the wrong version accidentally. (It's not a bad idea to look at the contents of your tarball before submitting it; see Project #0 for instructions on how to do that.)

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus.

Grading
Your grade for this project will be two-thirds correctness: I will run some number of test cases using Google test. Each is worth some number of points and is graded based on whether or not your code correctly determines if the puzzle has a solution, and if so, what it is. If it is determined that your program does not make an attempt to solve the problem at hand, you will not get these points, regardless of the result from testing. The tests will look a lot like the tests in your Google Test starting directory for this assignment; if you pass those, you're off to a good start, but it's not a guarantee.

The other one-third is style. ICS 46 isn't strictly about C++, but good programming style is important. Your important variable names should have meaningful names, your code should be appropriately commented, and so on.