Optional programming assignment Math 42, Spring 2015

This assignment is a chance for you to implement some algorithms related to the topics we've covered in the course. It will consist of four programming tasks, which will be due once per week starting on Friday April 10.

This assignment is entirely optional. If you complete it, your score can partially offset one exam grade or your homework grade. You may also choose to complete an optional essay assignment for the same bonus. If you complete both the essay and the programming assignment, I will grade both and use the higher score for the grading bonus.

Ground rules

- You may program in any language and on any platform that you like.
- You are not permitted to view or copy other students' source code, or to "check answers" by comparing your output files.
- You are permitted (and encouraged) to discuss the problems and outline algorithms with your classmates and with me, as long as you do not share specific code or submitted answers.

One exception to the second rule: I don't mind if you share source code that *only concerns file handling*, or any other technical issues that are not related to the number theory in the problem.

Format

Each week, a new task will be posted on the website. The task will describe the problem your program must be able to solve, and will provide some sample input and output.

When you believe that you have a working program, you may download an input file, run your code on the file, and write the output to another file. You will submit your output file and your source code.

There will be two types of input files: "small inputs" and the "large input." The small inputs will contain easier cases (e.g. the input numbers will be smaller), while the large input will have harder cases that require a faster algorithm to solve.

There will be ten small inputs available. You only need to solve one of them (there is no bonus for solving more than one). You may only attempt any given input file once (but if you fail, you can try again with a different small input). There will only be one large input, which will not be available for download until the day before the due date of the assignment.

Remark on runtime

I will always make sure that there is a (reasonable accessible) solution that would complete any input file in less than ten minutes on a personal computer. In most cases, the solution I have in mind will be able to solve the large input in less than one minute. If you write a solution that takes several hours, this is fine, but there is probably a faster option.

Submission instructions

When you decide to attempt to solve an input file, download the file from the course website and use your program to generate an output file. Email **both your output and your source code** to me. The file name of your output file must follow **exactly this format**: if your email address is `XXX@brown.edu`, and the name of the input file is `YYY.in`, then your output file must be titled `XXX_YYY.out`. For example, my Brown email address is `nathan_pflueger@brown.edu`, so if I am submitting an answer for the input file `small3.in`, I would title my output file as follows.

`nathan_pflueger_small3.out`

At least once per 24 hours, I will download any problem submissions and run an automatic grading script. The script will determine whether your output is correct and send you an email with the result. Your file name must follow exactly the right format, otherwise the automatic grader will either not know what to score it against or will not send the response to the correct email address.

Effect on your course grade
You will receive a "raw score" out of eight points. For each programming task, one point is available for the small inputs (which you will receive as long as you correctly solve *at least one*), and one point is available for the large input. As long as your raw score is at least 1, your score for the assignment overall will be 60% plus 5% times your raw score. For example, if you solve a small input for all four tasks, your score for the programming assignment will be 80%. If you only correctly solve one input file for one task, you will get 65% for the assignment (I want it to be worth your while to attempt at least one).

This percentage score will factor into your final grade as 5% of the total. This 5% will be removed from one of the other four categories (homework, midterm 1, midterm 2, final), whichever will increase your course grade the most. For example, the first midterm is 20% of your final grade by default, but this may shrink to 15% to make room for your score on the programming assignment.

If this would harm your course grade, then I won't do it. You are not putting yourself at risk by attempting the assignment.

Suggestions

The small inputs will usually be solvable by a "naive" algorithm (for example, if you need to check if a number is prime, just trying divisors up to $\sqrt{n}$ would be fast enough for the small inputs). You should use them to check if you understand the problem and have noticed any subtle issues with it. Since you can try ten times, you don't need to be afraid to try an input file to test your understanding. Once you solve the small input, you will probably need to improve your algorithm to have a chance at solving the large. I recommend that you use your "naive" solution (once you know it is correct) to test your more efficient algorithm for correctness. You only get one shot at the large input, so you should thoroughly test your code in any way you can before submitting.

You may find it helpful to create your own input files for testing purposes. You are free (and encouraged) to share these files, and the correct outputs for them, with each other to help with testing. The only thing you are not allowed to share is your own source code, or your answers to the official input files (small and large).

Each task description will specify some limits on the numbers in the input. Pay close attention to these, since they may affect which types of variables you will need to use. For example, if a parameter in the input is guaranteed to be less than 10^9, this means that you can use a 32-bit integer.