The Art+Logic Programming Challenge

Thanks for your interest in Art+Logic.

This file contains the instructions for the first part of the Art+Logic Programming Challenge.

This test is an important part of our hiring process; over the years, we have learned that there's not always a strong correlation between the contents of a candidate's resume and their ability to design and implement code at the level that we demand. The only way to determine how well someone writes code is to ask them to roll up their sleeves and write some.

Completing this test will give you an opportunity to display your abilities on some tasks that are not too different from those that are routinely encountered on Art+Logic projects. It's fairly typical on a project that a developer will be handed a fairly high-level specification for a piece of functionality and be asked to design and implement a clean, elegant, flexible solution to the problem in a short period of time. We also end up needing to jump into projects that are already in progress. Sometimes this means stepping into an existing Art+Logic project, but frequently it means inheriting a pile of broken and undocumented code written by a client (and typically, the original programmer responsible for this code is long gone, so debugging skills and detective work are required). Please bear in mind while completing this exercise that your goal should not be just to demonstrate your competence. We evaluate many more applicants than we have the capacity to hire; this test is key to identifying the most exceptional.

Part 1: Encoding and Decoding Values

This should take an hour or so.

For this task, you need to write a small program including a pair of functions that can

1. convert an integer into a special text encoding and then
2. convert the encoded value back into the original integer.

Assuming that your solution works correctly and cleanly enough to move forward in this process, these functions will need to be used in your part 2 submission.

The Encoding Function

This function needs to accept a signed integer in the 14-bit range [-8192..+8191] and return a 4 character string.

The encoding process is as follows:

1. Add 8192 to the raw value, so its range is translated to [0..16383]

2. Pack that value into two bytes such that the most significant bit of each is cleared
   Unencoded intermediate value (as a 16-bit integer):

   ```
   00HHHHHH HLLLLLLL
   ```

   Encoded value:

```
0HHHHHHH 0LLLLLLL
```

3. Format the two bytes as a single 4-character hexadecimal string and return it.

Sample values:

Unencoded (decimal)	Intermediate (decimal)	Intermediate (hex)	Encoded (hex)
0	8192	2000	4000
-8192	0	0000	0000
8191	16383	3fff	7F7F
2048	10240	2800	5000
-4096	4096	1000	2000

The Decoding Function

Your decoding function should accept two bytes on input, both in the range [0x00..0x7F] and recombine them to return the corresponding integer between [-8192..+8191]

Hi byte	Lo byte	Value
40	00	0
00	00	-8192
7F	7F	8191
50	00	2048
0A	05	-6907
55	00	2688

Your Task

You should supply source code (and whatever additional collateral files may be necessary or useful for us to understand, validate, and test your submission) for a working program that can accept text input in either decimal integer or hexadecimal format and convert between the two formats.

This program can be a command line application that reads from stdin or a file and writes to stdout or a file, or could be a web page or mobile/desktop app that uses a text edit box to accept input values by typing or pasting (or loading a file) and a button to convert that input into the desired output format so you can encode/decode sample data as described below.

This program must be written using one of our preferred programming languages:

- C++
- C#
- JavaScript
- Java *(see below)*
- Objective-C
- PHP
- Python
- Swift

Please don't submit solutions in Java unless you are specifically asking to be considered as an Android developer.

We'd prefer that you limit the use of any third-party libraries or frameworks where possible—remember that the goal here is to help us see how awesome your code can be when you're working on a problem where none of those libraries exist yet.

Your completed submission will be evaluated by a senior Art+Logic developer, who will be looking for several things, but most importantly:

- Does this code work correctly?
- Is it well designed and cleanly implemented?
- Was it completed in a reasonable amount of time?
- If I had to maintain this code, would that make me happy or angry?

To help us verify that your code works correctly, please include a plain text file in your submission package named "ConvertedData.txt" that contains encoded values for these integers:

- 6111
- 340
- -2628
- -255
- 7550

and decoded integers for these hexadecimal values:

- `0A0A`
- `0029`
- `3F0F`
- `4400`
- `5E7F`

In production, we also require that code written for Art+Logic conform to the conventions outlined in the Art+Logic Programming Style Guide (at http://styleguide.artandlogic.com). For the purposes of this part of the challenge, we're not *requiring* that you follow Art+Logic style, but reading the Style Guide will help you understand what we think good, clean code looks and feels like.

Additionally, we'll be considering how your skills and experience match up against the types of work that we are most commonly hired to do. Our clients come to us with a wide variety of needs and requirements, and our goal in recruiting is to find great developers who we would be able to staff onto as many different types of projects as possible. It's difficult for us to keep some sorts of specialists busy—for example, we don't do a lot of legacy J2EE or Perl work, so an amazing developer with those skills but not much else would be difficult for us to keep busy, and therefore that developer would be unlikely to proceed very far into our recruiting process.

Our ideal candidates have a good mix of

- **'Modern' web development**—HTML5 and its associated APIs, heavy client-side JavaScript (we also make use of JavaScript libraries/frameworks like jQuery, CoffeeScript, Angular.js, Backbone.js) and server-side development, most often using dynamic languages like Python, PHP, and Ruby.

- **Mobile development**—Most commonly we are engaged to do iOS development, but we also do Android work.

- **'Other'**—we still do a fair amount of Windows and Mac desktop development, and occasionally encounter old-school enterprise-style client/server applications or embedded systems development.

The form where you'll send us your submission contains a section where you can indicate your experience with a number of our most commonly encountered languages, platforms, and frameworks. We'll use that information along with the results of the part 1 evaluation to decide which developers to invite to the next part of the challenge.

Please package your source and any other files accompanying your submission (including the `ConvertedData.txt` file) into a ZIP file named `firstName_lastName_Part1.zip` and upload the file when completing the submission form at http://www.artandlogic.com/alpc1.

Please do *not* include any executable, `.jar` or other `.zip` files inside your submission. For security reasons, our systems will reject any submissions that do include any of these file types without notifying either of us. Complete information on the prohibited file types can be found at https://support.google.com/mail/answer/6590?hl=en.

Note that one of the prohibited file extensions is `.js` — if your solution contains JavaScript source, please rename these files to use the extension `.javascript` to ensure that your submission reaches us.

If you have questions, email recruiting@artandlogic.com.

File version 2018-01-30