Write code using the language of your choice (Java, C, C++, Python, C# are preferred -- let me know if you want to use something else) to implement regular expressions to screen for the following:

1. Social Security Number (can be with dashes, whitespace, or no spaces at all)
    1. **2 points Extra credit if you make sure the SSN is allowable based on numbering rules by the Social Security Administration**
2. US Phone number - parentheses are optional, as well as the dash between the last two sections
    1. **2 points Extra credit if you allow only official area codes**
3. E-mail address
4. Name on a class roster, assuming zero or more middle initials - Last name, First name, MI (e.g. Smith, John, L
5. Date in MM-DD-YYYY format - separators can be -'s, /'s -- **<u>you must make sure months, days, year are valid (this includes leap years)</u>**
6. House address - Street number, street name, abbreviation for road, street, boulevard or avenue (full version of those items should also be accepted)
7. City followed by state followed by zip as it should appear on a letter
    1. **2 points extra credit if you make sure the 2 character state abbreviation (e.g. WA) is valid**
8. Military time (no colons used and leading 0 is included for times under 10)
9. US Currency down to the penny (ex: $123,456,789.23)
10. URL, optionally including http:// or https://, upper and lower case should be accepted
11. A password that contains at least 10 characters and includes at least one upper case character, one lower case character, one digit, one punctuation mark, and does not have more than 3 consecutive lower case characters
12. All words containing an odd number of alphabetic characters, ending in "ion"

- Place each regular expression in its own function/method. Provide documentation so that I can easily follow your code. Utilize good naming conventions, whitespace, etc.
- Note that you should try and use regular expression(s) to solve the problem first and foremost, but it is okay to include a little helper code where necessary or it makes sense.
- **BE SURE AND DOCUMENT ANYTHING YOU COULD NOT GET TO WORK, OR ANY SHORTCOMINGS FOR A GIVEN TASK.**

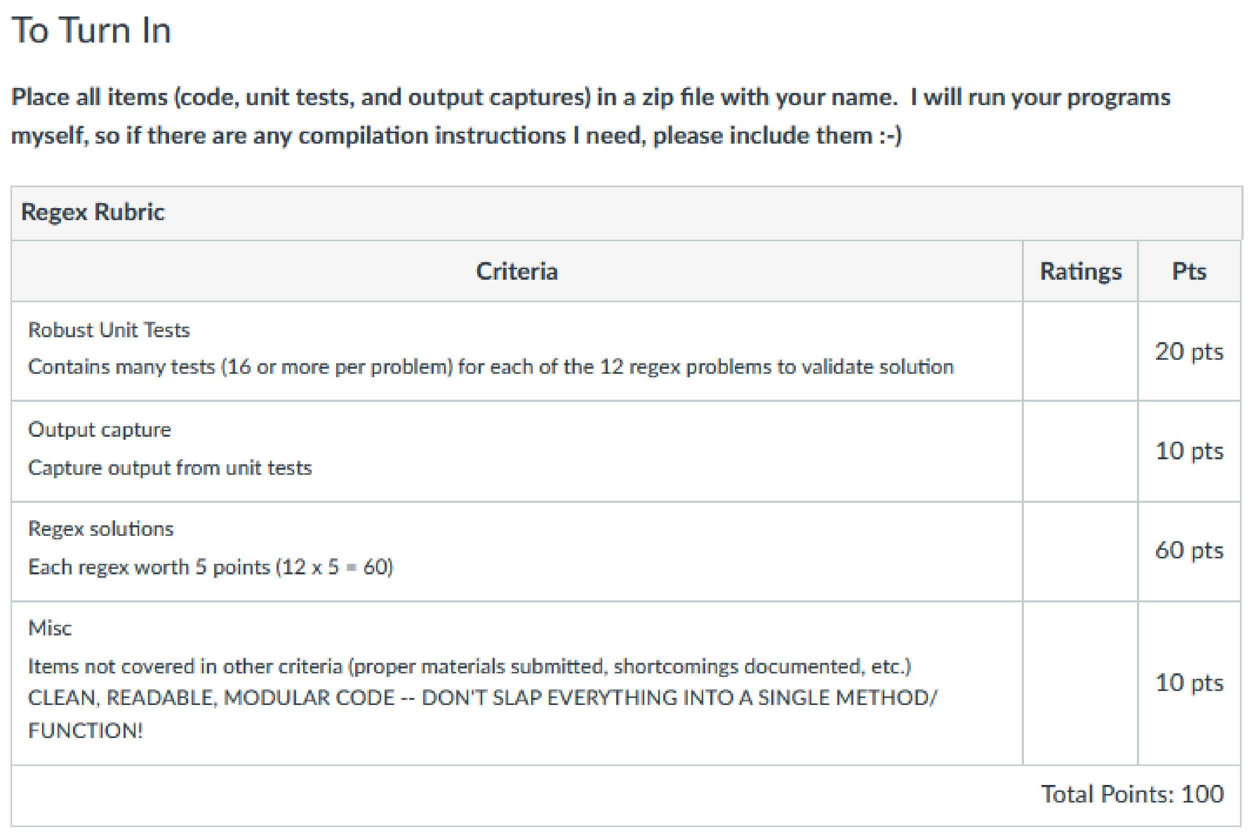
TESTING

Write unit tests for each of the 12 categories of regular expressions. You must have **<u>at least</u>** 16 unit tests for each category (at least 8 that check for valid results and at least 8 that check for invalid results). More specifically, tests must include expressions that should be accepted as valid and those that should be rejected. Be sure and test all major edge cases for each category. Name your test methods to clearly describe what they are testing (e.g. testValidSSNAllDigits, testValidSSNDashes, testRejectSSNOneSpaceOneDash, etc.).

<u>IMPORTANT NOTE</u>: It is ok to collaborate on unit tests with your team members. If you do this, please note it as part of your submission.

Turn your unit tests in as part of your source code. **Also, include output captures that CLEARLY show the results of running your unit tests. How you capture the output is up to you, but make sure the names of the tests run are shown :-)**

Make sure you document any cases you could not get to work. This may result in fewer points lost for that problem.



To Turn In

Place all items (code, unit tests, and output captures) in a zip file with your name. I will run your programs myself, so if there are any compilation instructions I need, please include them :-)

Regex Rubric

Criteria	Ratings	Pts
Robust Unit Tests Contains many tests (16 or more per problem) for each of the 12 regex problems to validate solution		20 pts
Output capture Capture output from unit tests		10 pts
Regex solutions Each regex worth 5 points (12 x 5 = 60)		60 pts
Misc Items not covered in other criteria (proper materials submitted, shortcomings documented, etc.) CLEAN, READABLE, MODULAR CODE -- DON'T SLAP EVERYTHING INTO A SINGLE METHOD/ FUNCTION!		10 pts
		Total Points: 100