Programming Exercise 02 - Conditionals

Authors: Brian, Nicolas, Tomas, Ishuma

Problem Description

Typically, we expect the numbers we use to be represented in the decimal numeral system, base-10. However, this is not the only way we can represent numbers. Computers will often store data in the binary numeral system, base-2. When tallying, people will often use the unary numeral system, base-1. For this assignment, we will be exploring the ternary numeral system, base-3. To represent a decimal number, which we will call *n*, in ternary, we can perform the following algorithm:

- Begin with an empty string to hold our answer.
- Compute *n* modulo 3 and append it to the front of our answer.
- Divide *n* by 3 (what kind of division should occur?).
- Repeat the above two steps until *n* is zero.

To demonstrate, let's apply the algorithm to *n* = 100 and see what happens.

1. **`100 % 3 is 1, so our partial answer is "1"`**
2. **`100 / 3 is 33`**
3. **`33 % 3 is 0, so our partial answer is "01"`**
4. **`33 / 3 is 11`**
5. **`11 % 3 is 2, so our partial answer is "201"`**
6. **`11 / 3 is 3`**
7. **`3 % 3 is 0, so our partial answer is "0201"`**
8. **`3 / 3 is 1`**
9. **`1 % 3 is 1, so our partial answer is "10201"`**
10. **`1 / 3 is 0`**
11. **`We reached 0. The final answer is "10201"`**

We can verify that the final answer is correct because

$$1 \cdot 3^4 + 0 \cdot 3^3 + 2 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 = 81 + 0 + 18 + 0 + 1 = 100$$

Notice how this conversion algorithm repeats the same two steps over and over (mod and division). This means we can implement it with iteration a.k.a. loops!

Your assignment will consist of 2 main parts:

- Start with some arbitrary number in decimal and create the ternary string representation.
- Print out information about the ternary representation.

Solution Description

1. Create a class called `Ternary`.
2. Create the `main` method.

THIS GRADED ASSESSMENT IS NOT FOR DISTRIBUTION.
ANY DUPLICATION OUTSIDE OF GEORGIA TECH'S LMS IS UNAUTHORIZED.

3. Inside the `main` method, create these variables:
    a. Create an **int** called `initialNum` and assign it any positive integer from 1-1000.
    b. Create a **String** called `answer` and assign it the empty string, "". This will hold our ternary representation as we build it.
    c. Create an **int** called `zeroCount` and assign it the value 0. This will count the number of zeroes in our answer string.
    d. Create an **int** called `oneCount` and assign it the value 0. This will count the number of ones in our answer string.
    e. Create an **int** called `twoCount` and assign it the value 0. This will count the number of twos in our answer string.
    f. Create an **int** called `currentNum` and assign it the value `initialNum`.
4. Print `initialNum`.
5. Create a `while` loop that terminates when `currentNum` is equal to 0.
6. Within the `while` loop, create a local **int** variable called `digit` and assign it the value `currentNum % 3`.
7. Use if-else if-else statements to do the following:
    a. If `digit` is a 0, increment `zeroCount`.
    b. If `digit` is a 1, increment `oneCount`.
    c. If `digit` is a 2, increment `twoCount`.
8. Concatenate `digit` to the front of the string `answer`.
9. Re-assign the value of `currentNum` to be `currentNum / 3`.
10. After the `while` loop, use **three** print statements to output the following (without the angle brackets) on separate lines:
    a. "Decimal representation: <`initialNum`>"
    b. "Ternary representation: <`answer`>"
    c. "<`zeroCount`> zeroes, <`oneCount`> ones, and <`twoCount`> twos"
11. Create an **int** variable `digitSum` and assign it to the sum of the digits in the ternary representation.
    a. One way of evaluating this sum is 0 * `zeroCount` + 1 * `oneCount` + 2 * `twoCount`.
12. Finally, use a `switch` statement to print **one** of the following statements according to the value of `digitSum % 5`:
    a. 0: "The ternary digits sum to a multiple of 5!"
    b. 1: "The ternary digits almost summed to a multiple 5!"
    c. 4: "So close!"
    d. Any other value: "Nope!"

Example Outputs

Please refer to the PE clarification thread on the course forum for examples.

HINT: To help debug your code, try inserting print statements in places where variables are changed.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Ternary.java`

Make sure you see the message stating the assignment was "submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder test are provided as a courtesy to help "sanity test" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or professor via the class forum for clarification.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the assignment (submit early and often). We will only grade your latest submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine. (We are not using checkstyle for this programming exercise, but in future HWs we will).

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Allowed Imports

To prevent trivialization of the assignment, you are not allowed to use any imports.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Collaboration

Only discussion of the assignment at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going too far. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs.

You **MAY NOT** use code generation tools to complete this assignment. This includes generative AI tools like ChatGPT.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- **Check on Ed Discussion for a note containing all official clarifications and sample outputs**

It is expected that everyone will follow the Student-Faculty Expectations document, and the Student Code of Conduct. The professor expects a **positive, respectful, and engaged academic environment** inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. **No inappropriate language is to be used, and any assignment, deemed by the professor, to contain inappropriate, offensive language or threats will get a zero**. You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.

THIS GRADED ASSESSMENT IS NOT FOR DISTRIBUTION.
ANY DUPLICATION OUTSIDE OF GEORGIA TECH'S LMS IS UNAUTHORIZED.