CENG101: Introduction to Computer Engineering

Homework 1

Deadline: SUNDAY, 24 NOVEMBER 2024 by 20:00

You will need to submit on Aybuzem your program with the .java extension as explained below.

Rules

1. Late submissions will NOT be accepted. A 0 grade will be given for late submissions.
2. No deadline extensions will be granted.
3. Your code will be evaluated based on predefined test cases, which you have not been provided with.
4. If your program does not compile and/or gives a run-time error then the maximum grade you'll get is 20% of the entire grade.
5. Your code must be developed individually. You cannot collaborate with your peers. You cannot post your code on any public repository, nor can you share it with anybody. Any violations to this code of conduct will be treated as academic misconduct.
6. Submitted code will be automatically checked using tools that detect plagiarism.
7. In case cheating is detected, the grade of this assignment will be zero and a further penalty of 10 points will be applied to the total course grade.
8. You may be asked to explain your code and rewrite some parts of it in front of the instructors.
10. Submit the tasks in 2 different java files named `Yourfirstname_Yourlastname_TaskN.java` where `N` is either `1 or 2`. Make sure to include the underscore '_' in the filename and **don't** insert space.
11. In order to be accepted, your submission
    - must be of text format with a '.java' extension (no Word or other formats and no archives will be accepted), correctly named as explained in Rule 10.
    - You will receive a zero grade if you do not submit the assignment or you do not adhere to this rule!

Warning: This assignment is designed for you to learn programming and have programming experience. If you struggle, please speak to us. Do not use chatGPT or any other similar websites to seek solutions for this assignment. We always perform a similarity check among all submissions.

Task 1 (40%): In the fictional world, there's a programming language known as `CENG_101`. Despite its simplicity, the language has an interesting structure.

This language operates with only one variable, named `A`. There are three possible operations:

- The `**` operation computes A^2 and assigns it to `A`.
- The `++` operation increases the value of `A` by `1`.
- The `--` operation decreases the value of `A` by `1`.

A statement in `CENG_101` is composed of one operation and the variable `A`, with no spaces between them. The statement only includes the characters `*`, `+`, `-`, and `A`. Executing a statement applies the specified operation to `A`.

A program in `CENG_101` consists of multiple statements, each provided on a separate line. These statements are executed sequentially. Your objective is to compute the final value of the variable `A` after the entire program has been executed. The initial value of `A` is 0.

Input

Your program will process a sequence of input consisting of `CENG_101` statements. The first input line contains a single integer `n`, representing the number of statements in the program.

Each of the next `n` lines contains one statement. Each statement includes exactly one operation (`**`, `++`, or `--`) and exactly one variable, `A`. The operation and the variable may appear in any order, and there are no blank lines in the input.

Output

Print a single integer to the console representing the final value of `A` after all statements have been executed. Ensure that the output contains only the integer, with no additional characters, strings, or spaces.

Important Points

1. Your output should be a single integer representing the final value of `A` after all statements in the input are processed. Do not print the output to a separate file.
2. Ensure that the output does not contain additional characters, strings, or spaces.
3. The variable `A` always starts with an initial value of 0.
4. The operation and the variable `A` can be written in any order as can be seen by the example below.
5. The only string methods you can use in this Task are `charAt()` and `length()`.

Example I/O (bold indicates user input)

```
> javac firstName_lastName_Task1.java
> java firstName_lastName_Task1
> 7
> ++A
> A++
> A**
> A--
> **A
> --A
> A--
> 7
```

(Your program will output a single integer **7** showing the final value of `A` after executing all these statements one after another.)

The explanation for obtaining `7` is as follows:

```
++A             -> A becomes 1
A++             -> A becomes 2
A**             -> A becomes 4
A--             -> A becomes 3
**A             -> A becomes 9
--A             -> A becomes 8
A--             -> A becomes 7
```

Task 2 (60%): In this task, you are asked to decode a string based on a specific encoding rule. The string consists of lowercase English letters, and it is transformed using the following procedure:

For each letter in the original string, its position in the alphabet is determined (for example, '`a`' is `1`, '`b`' is `2`, and so on).

- If the letter's position is a **single-digit** number (less than 10), its corresponding number is written directly.
- If the letter's position is a **two-digit** number (10 or greater), its corresponding number is written followed by a `0`.

For example, if the original string is "`lamp`", the encoding process would look like this:

'`l`' is the 12th letter in the alphabet, so "`120`" is added to the encoded string.
'`a`' is the 1st letter in the alphabet, so "`1`" is added.
'`m`' is the 13th letter in the alphabet, so "`130`" is added.
'`p`' is the 16th letter in the alphabet, so "`160`" is added. Thus, the encoded version of "`lamp`" becomes "`1201130160`".

Given an encoded string, your task is to decode it back to the original form by reversing this encoding process.

Input
Your program will read from a standard input, which contains multiple encoded strings. The format of the input is as follows:

- The first line of the input contains a single integer `n`, representing the number of encoded strings.
- Each of the next `n` lines contains two space-separated values:
    - An integer representing the length of the encoded string.
    - The encoded string itself, which follows the encoding rules described earlier.

Output
For each encoded string, your program should output its decoded version to the console. You will print exactly `n` decoded strings, each on a new line, in the same order as they appear in the input file.

Make sure the output only contains the decoded strings and nothing else. Do not add any extra space or new line.

Important Point

- The only string methods you can use in this Task are `charAt()` and `length()`.

Example

Assume input has the following content:

```
3
4 2606
7 1111100
5 11111
```

Then you are expected to output the following 3 decoded strings in this order:

```
zf
aaaaj
aaaaa
```