Chapter 2: Programming Project 3 (Python)

(Split digits)

Write a program that prompts the user to enter a four-digit integer and displays the number in reverse order. Here is a sample run:

```
Enter an integer: 5213

The reversal is

3

1

2

5
```

For a hint on this program, please see https://liangpy.pearsoncmg.com/pyrevel1e.html. If you get a logic or runtime error, please refer to https://liangpy.pearsoncmg.com/faq.html.

Hint:
Use integer in this program. The last digit of number is number % 10. Discard the last digit using number // 10.
Step 1: Get the last digit n1. n1 = number % 10.
Step 2: Get the second last digit n2. number = number // 10. n2 = number % 10.
Step 3: Get the third last digit n3. number = number // 10. n3 = number % 10.
Step 4: Get the first last digit n4. number = number // 10. n4 = number % 10.
Step 5: Now print n1, n2, n3, and n4.

Chapter 20: Programming Project 2: (Java)

(*Evaluate expression*)

Modify LiveExample 20.9 EvaluateExpression.java to add operators `^` for exponent and `%` for modulus. For example, `3 ^ 2` is `9` and `3 % 2` is `1`. The `^` operator has the highest precedence and the `%` operator has the same precedence as the `*` and `/` operators.

Your program should prompt the user to enter an expression, evaluate the expression, and display the result.

Sample Run

```
Enter an expression: (5 * 2 ^ 3 + 2 * 3 % 2) * 4
(5 * 2 ^ 3 + 2 * 3 % 2) * 4 = 160
```

Class Name: `Exercise20_23`

For a hint on this program, please see https://liveexample.pearsoncmg.com/javarevel2e.html. If you get a logic or runtime error, please refer to https://liveexample.pearsoncmg.com/faq.html.

Hint:

Study the code in LiveExample 20.12 in Section 20.11 using some samples, for example, 1+2, 2 * 3 - 3, etc.

Modify the example EvaluateExpression.java incrementally. One step at a time and you will know which step you are struggling.

Step 1. Read the input expression from the console rather than passing from the command line. (The code in the example passes the expression from the command line.)

Step 2. The operator % can be implemented similar to the * and / operators. Add the code for processing % in lines 51-61 in LiveExample 20.12.

Step 3. The operator ^ has the highest precedence. However, note that the ^ operator is right-associative, meaning that 2 ^ 3 ^ 2 is same as 2 ^ (3 ^ 2). In lines 51-61 in LiveExample 20.12, the program processes the * and / operators, add the code for processing the ^ operator after this block.