CSC1003 Assignment 1

Important Notes:

1. The assignment is an individual project, to be finished on one's own effort.
2. The work must be submitted before 6pm Oct. 14, 2024 (Monday), Beijing Time. This is a firm deadline. No late submissions are accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% marks will be deducted.

Marking Criterion:

1. The maximum score a student could obtain in this assignment is 100 marks (i.e. more than 100 marks will be counted as 100 marks).
2. Two java programs are to be submitted. Each program will be evaluated with several unseen test cases. A submission obtains the full score if and only if both programs pass all test cases.

Running Environment:

1. The submissions will be evaluated in the OJ system running Java SDK. It is the students' responsibility to make sure that his/her submissions are compatible with the OJ system.
2. The submission is only allowed to import four packages of (java.lang.*; java.util.*; java.math.*; java.io.*) included in Java SDK. No other packages are allowed.
3. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.

Submission Guidelines:

1. The score is based on your submission on OJ, but you also need to submit your code (that is, **directly** submit *the screenshots, TestFibonacci.java and TestMathExpr.java*) on blackboard. Otherwise, you will get 0 for this assignment.
2. Inconsistency with or violation from the guideline leads to marks deduction.z
3. All students are reminded to read this assignment document carefully and in detail. No argument will be accepted on issues that have been specified in this document.

Exercise One (15 marks):

Finish at least three problems (Click to access) within P1000 to P1009 online (i.e. P1000, P1001, P1004 – P1009), and upload screenshot(s) to show that you finished the exercise. You can earn 15 marks without requiring "AC" by attempting to complete 3 questions.

Exercise Two (50 marks):

Fibonacci series is defined by:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …
Represented by an array, it would be: a[0] = 1, a[1] = 1, a[2] = 2, …, a[n] = a[n-1] + a[n-2].
Write a Java program named "**TestFibonacci.java**" which reads ***T*** groups of two non-negative numbers from the **console by using System.in**. Denote the two numbers by *n* and *d*, and the program is expected to output *d* elements of the series from **n-th** element (i.e. *a[n-1]*) in the reverse order to the **console** by using **System.out**. An example is given below.

An example of console input	Expected console output
2 5, 3 7, 7	5, 3, 2 13, 8, 5, 3, 2, 1, 1
3 6, 1 1, 1 3, 3	8 1 2, 1, 1

Note:

1. $1 \leq d \leq n \leq 18,\ 1 \leq T \leq 20.$
2. Each line of the input file consists of a positive integer, a comma, a space, and a non-negative number. The output is expected to be a series of numbers, separated by a comma and a space. If there is no number to output, the output is an empty line. It is guaranteed that the input is valid and d > 0.
3. The first line of each test case input will be an integer N (In the example above, N equals to 3). This integer N represents the number of lines of data to be input next. This part of the code is already given in the template, please refer to the program template section below.

Exercise Three (50 marks):

Write a java program named "TestMathExpr.java" to evaluate mathematical expressions with two operands (nonnegative integers) and one operator (+, -, *, or /).

An example of console input	Expected console output
5 3 + 2 4 * 4 155 / 5 157 / 5 158 / 5	5 16 31 31 31
4 3 - 5 0 + 7 7 * 0 100 / 0	-2 7 0 invalid

Note:

1. Each row of the console input consists of a non-negative integer *a*, a space, an operator *p*, a space, and another non-negative integer *b*, with $0 \leq a, b \leq 500,\ p \in \{+, -, *, /\}$.
2. The output is an integer (**round down**) if the input mathematical expression is valid, or "invalid" if the expression is invalid.
3. The first line of each test case input will be an integer N (In the example above, N equals to 5 and 4). This integer N represents the number of lines of data to be input next. This part of the code is already given in the template. Please refer to the program template section below.

Program Template:

Program 1:

TestFibonacci.java | TestMathExpr.java

```java
import java.util.*;

public class TestFibonacci {

    static Scanner input = new Scanner(System.in);

    // here is the function you need to implement
    public static void parse_line(int n, int d) {

    }

    public static void main(String[] args) throws Exception {
        int line_number = Integer.parseInt(input.nextLine());
        for(int i=0; i<line_number; i++) {
            String s = input.nextLine();
            String t[] = s.split(", ");
            int n = Integer.parseInt(t[0]);
            int d = Integer.parseInt(t[1]);
            TestFibonacci.parse_line(n, d);
        }
    }
}
```

Program 2:

TestFibonacci.java | TestMathExpr.java

```java
import java.util.*;

public class TestMathExpr {

    static Scanner input = new Scanner(System.in);

    // here is the function you need to implement
    public static void parse_line(String s1, String s2, String s3) {

    }

    public static void main(String[] args) throws Exception {
        int line_number = Integer.parseInt(input.nextLine());
        for(int i=0; i<line_number; i++) {
            String s = input.nextLine();
            String t[] = s.split(" ");
            TestMathExpr.parse_line(t[0], t[1], t[2]);
        }
    }
}
```

1. Each test case on OJ may have multiple lines of input, but in these templates, you only need

to complete the implementation of the method “**parse_line()**” , which processes each line of input. Please refer to the template of “TestFibonacci.java” and “TestMathExpr.java” above carefully.

2. These templates are just references, which means you don't need to write code based entirely on them. You can change any line of code in the sample file, or even write a new file from scratch, but you have to meet the required functions defined above.
3. Students can unlimitedly submit their code to the OJ system, but only the last submission is used for scoring. This means that if your first submission is already excellent, then the next submission will potentially pull your score down.