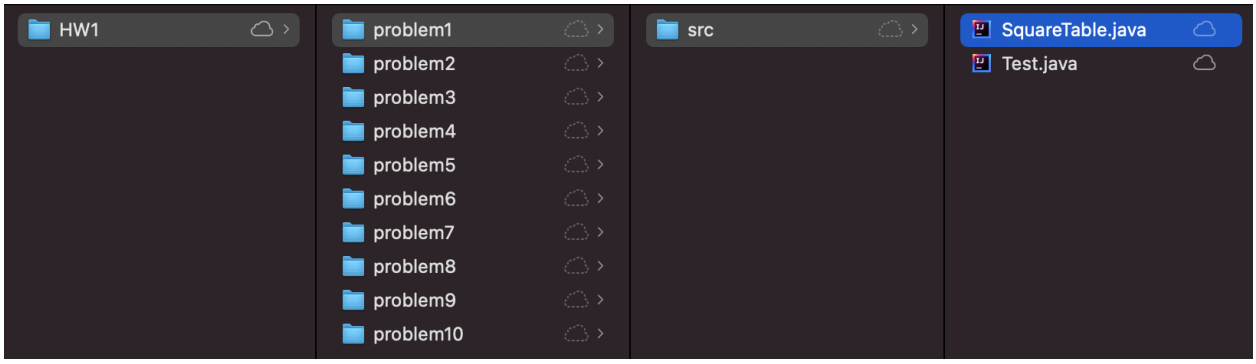
AY 2022 Assignment 1 [7 Marks]

Date / Time	16 September 2022 – 30 September 2022 23:59
Course	**[M1522.600] Computer Programming**
Instructor	Youngki Lee

- You can refer to the Internet or other materials to solve the assignment, but you ***SHOULD NOT*** discuss the question with anyone else and need to code **ALONE**.
- We will **use the automated copy detector to check the possible plagiarism** of the code between the students. The copy checker is reliable so it is highly likely to mark a pair of codes as the copy even though two students quickly discuss the idea without looking at each other's code. Of course, we will evaluate the similarity of a pair compared to the overall similarity for the entire class.
- We will do the manual inspection of the code. In case we doubt that the code may be written by someone else, we reserve the right to request an explanation about the code. We will ask detailed questions that cannot be answered if the code is not written by yourself.
- For the first event of plagiarism, you will get 0 marks for the specific assignment. You will fail the course and be reported to the dean if you copy others' code more than once.
- Download and unzip "HW1.zip" file from the Autolab. The file contains the skeleton codes for 10 questions (in "/problemX" directory for question #X).
- **Do not modify the overall directory structure**. Simply fill in the codes in the appropriate files.
- Do not write comments in Korean in the source code. Autolab doesn't recognize non-ASCII characters and will output compile errors.
- Do not use external libraries.
- Do not use Java Collections Framework (i.e., ArrayList, Hashmap, …). (https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html)
- If you have any questions, feel free to ask any in the Slack channel.

Submission Guidelines

1. For submission, compress the entire “HW1” directory into a single zip file.
2. The directory structure of your submitted file (after unzipping) should look like the following. When you extract the zip file, **there must be the HW1/ directory.**
   Note that you will get **0 marks** for the wrong submission format.



3. Submit the zip file on Autolab.

Overview

1. The goal of this assignment is to implement simple Java programs.
2. Implement the **Target function** indicated in each question. **Do NOT change the directory structure, existing file names, existing class names, or existing function signatures.** You may add new files, classes, or functions if necessary.
3. Assume all inputs are correct. This means that you do not have to handle wrong inputs.
4. The trailing spaces after the last line of the output will be ignored. However, the intermediate lines should NOT have trailing spaces when output has multiple lines.

Answer	Student Output	Is it correct?
abc	abc	O
abc abc	abc abc	O
abc abc	abc abc	X

5. You can test your code with Autolab. It will score your code based on some test cases we provide. Take a look at ‘Autolab Guide.pdf’.
6. Although Autolab provides some useful test cases, we strongly recommend that you use additional test cases of your own. You can easily test your code by adding test cases in the Test.java file.

7. We will use a more comprehensive set of test cases for evaluation, so the final score may differ from what you see in Autolab.
8. There is **NO partial point** for a question. That means the final score for each question will be either 0 or 100. However, partial points will be displayed in Autolab when you do this homework to help you test your code.

Question 1: Square Table [0.7 Marks]

Objective: Write a program that prints out all squares that are greater than or equal to the first input number `n` and less than or equal to the second input number `m`.

Target Function: `public static void printSquareTable(int n, int m)` (in SquareTable.java)
Input: int n, m (1 <= n <= m <= 900).
Output: Square table that corresponds to input numbers N and M (see the example below). The output format is **'k times k = k^2'**.
Note: As mentioned in the overview, the intermediate lines should **NOT** have trailing spaces!

Input	Output

`n=25 m=49`	`5 times 5 = 25`
	`6 times 6 = 36`
	`7 times 7 = 49`

`n=1 m=23`	`1 times 1 = 1`
	`2 times 2 = 4`
	`3 times 3 = 9`
	`4 times 4 = 16`

`n=3 m=3`	

Question 2: Sum of Fibonacci Numbers [0.7 Marks]

Objective: Write a program that prints the sum of N smallest Fibonacci numbers.

Description: Fibonacci Numbers, commonly denoted as F_n, form a sequence called the Fibonacci sequence. The sequence starts with the two numbers, 0 and 1, and each number

in the sequence is the sum of its two preceding ones, that is, $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ for n > 1. Print F_0 ~ F_{N-1} given N as the input.

Reference: https://en.wikipedia.org/wiki/Fibonacci_number

Target Function: `public static void printFibonacciNumbers(int n)` (in FibonacciNumbers.java)
Input: Integer N (1 <= N <= 46).
Output: The first N Fibonacci Numbers from the smallest to the largest and their sum. If the sum has more than five digits, print only its last five digits (including leading zeros).
Hint: Be careful with the types!

Input	Output
5	0 1 1 2 3 sum = 7
30	0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 sum = 46268

Question 3: Drawing Figure [0.7 Marks]

Objective: Write a program that prints the appropriate star(*) pattern for a given number.

Target Function: `public static void drawFigure(int n)` (in DrawingFigure.java)
Input: Integer N (1 <= N <= 50).
Output: The corresponding star pattern as below.
Note: As mentioned in the overview, the intermediate lines should **NOT** have trailing spaces!

Input	Output
1	*
2	* ***
3	* * * *****

4	<pre> * * * * * *******</pre>

Question 4: Character Finder [0.7 Marks]

Objective: Write a program that

1. Takes an arbitrary string composed of alphabets.
2. Prints all alphabets appearing in the string and the index of their first & last occurrence. Print them in lexicographic order (A, B, C, …, Y, Z, a, b, c, …, y, z). Do NOT print out alphabets that have never appeared in the input string. See the examples below.
3. The index starts with 0. (The first character of the string has index 0.)

Target Function: public static void findCharacter(String str) (in CharacterFinder.java)
Input: String str (1 <= str.length() <= 200) composed of alphabets.
Output: For every appearing alphabet **x**, print **"x: i j"** in a line where x is the alphabet and i, j is the index of the first and the last occurrences, respectively.
Note: As mentioned in the overview, the intermediate lines should **NOT** have trailing spaces!

Input	Output
abb	a: 0 0 b: 1 2
bbbaAabaccHadAHbcbHHAdcbbeHaH	A: 4 20 H: 10 28 a: 3 27 b: 0 24 c: 8 22 d: 12 21 e: 25 25

Question 5: IP Address [0.7 Marks]

Objective: Write a program that counts the number of valid IP addresses that can be made from the given string.

Description:

- A valid IP address is composed of four integers that are separated by dots.
- Each integer is in the range of 0 ~ 255 (inclusive).
- Each integer should **NOT** have leading zeros.
- Ex) “0.0.0.0” and “147.46.1.255” are valid IP addresses.
- Ex) “01.0.0.0” and “147.46.1.256” are invalid IP addresses.

Given a string composed only of digits (0~9), print the number of valid IP addresses that can be made by inserting dots in the string. If there are no addresses that can be made, print 0.

Target Function: `public static void countValidAddress(String str)` (in IPAddress.java)
Input: String str (1 <= str.length() <= 20) which is composed only of digits.
Output: Number of valid IP addresses that can be made from the given string, by adding the dots.

Input	Output	Explanation
`0000`	`1`	`The only valid IP address is “0.0.0.0”`
`101023`	`5`	`Valid IP addresses are "1.0.10.23","1.0.102.3","10.1.0.23","10.10.2.3","101.0.2.3"`

Question 6: Prime Factorization [0.7 Marks]

Objective: Write a program that factorizes the given integer into prime numbers.

Description: A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers. The smallest prime number is 2, and the second smallest prime is 3. Prime factorization is finding which prime numbers multiply together to make the given number. For example, 12 can be factorized into 2 x 2 x 3.
Reference:
https://en.wikipedia.org/wiki/Prime_number
https://en.wikipedia.org/wiki/Integer_factorization

Target Function: `public static void factorize(int n)` (in PrimeFactorization.java)
Input: Integer n (1 <= n <= 1000).
Output: Prime factors in increasing order, separated by whitespace. Don’t print anything when n = 1.

Input	Output
12	2 2 3
2	2

Question 7: Palindrome String [0.7 Marks]

Objective: Write a program that finds the longest palindrome substring.

Description: A palindrome is a string that is the same when you read it forward from the beginning or backward from the end. For example, “abcba” is a palindrome and “abc” is not a palindrome. Among the substrings of the given string, print the longest palindrome. If there are multiple palindrome substrings with the same length, print the one that appears earlier.

Target Function: `public static void printLongestPalindromeSubstring(String str)` (in PalindromeString.java)
Input: String str (1 <= str.length() <= 100) which consists of only alphabets and digits.
Output: The longest palindrome substring.

Input	Output
babbd	bab
abc	a
zaabbccbbaaz	zaabbccbbaaz

Question 8: Matrix Multiplication [0.7 Marks]

Objective: Write a program that runs matrix multiplication with the two input matrices.

Description: Implement a method that prints out the result of the matrix multiplication. The output format should be as follows.

- Each row of the result matrix should be printed in one line.
- The numbers of each row should be separated by whitespace.

Reference: https://www.mathsisfun.com/algebra/matrix-multiplying.html

Target Function: `public static void multiply(int[][] A, int[][] B)` (in MatrixMultiplication.java)
Input: Two 2D int arrays of size N x M and M x K (1 <= N, M, K <= 10).
Output: Result matrix of the matrix multiplication.
Note: As mentioned in the overview, the intermediate lines should **NOT** have trailing spaces!

Input	Output

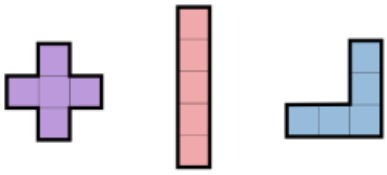
`A =`
`[[1 2]`
`[3 4]`
`[5 6]]`
`B =`
`[[1 1 2],`
`[1 0 0]]`
	`3 1 2`
	`7 3 6`
	`11 5 10`

Question 9: PentominoSum [0.7 Marks]

Objective: Write a program that calculates the maximum sum of the numbers that one pentomino covers.

Description: A pentomino is a shape where five 1x1 squares are connected together. You are trying to place a pentomino on a board of size N x M. Each cell of the board has an integer written on it. When you place a pentomino on the board, five numbers should overlap with the pentomino. Write a program that finds out the maximum sum of the numbers that can overlap with a single pentomino. A pentomino may be rotated or flipped.

Use the following three pentominoes.



Target Function: `public static void printMaxSum(int[][] board)` (in PentominoSum.java)
Input: 2D array of integers of size N x M (5 <= N, M <= 10).
Output: The maximum sum.

Input	Output
[[1 1 1 1 1] [1 4 4 5 1] [1 2 1 4 1] [1 2 1 4 1] [1 1 1 1 1]]	21

Explanation: The maximum sum can be obtained by placing ¬ shaped pentomino as follows.

1	1	1	1	1
1	4	4	5	1
1	2	1	4	1
1	2	1	4	1
1	1	1	1	1

Question 10: Pang Pang Pang! [0.7 Marks]

Objective: Write a program that simulates the Pang Pang Pang game.

Description: Pang Pang Pang game runs as follows.

1. You are given a N x M board where each tile of the board has an integer number (1~5).
2. When three or more consecutive tiles (horizontally or vertically) have the same numbers, they disappear. This is called Pang!
3. The remaining tiles fall down to fill in the empty spaces. (An empty space is represented by integer number 0.)
4. Repeat 2 and 3 until there are no more tiles to Pang!
5. Print what the board looks like.

Target Function: `public static void afterPang(int[][] board)` (in PangPangPang.java)
Input: 2D int array of size N x M (1 <= N, M <= 10), where each element of array is between 1 and 5 (inclusive).
Output: The board after all the Pang! is done. Each row should be printed in each line, and each number in a row is separated by a whitespace.
Note: As mentioned in the overview, the intermediate lines should **NOT** have trailing spaces!

Input	Output
[[5 2 3 4 1] [3 2 2 3 3] [4 2 1 2 4] [2 2 2 1 4] [4 4 3 3 4]]	0 0 0 0 0 5 0 0 0 0 3 0 0 4 0 4 0 3 3 0 4 4 2 2 0

Explanation: