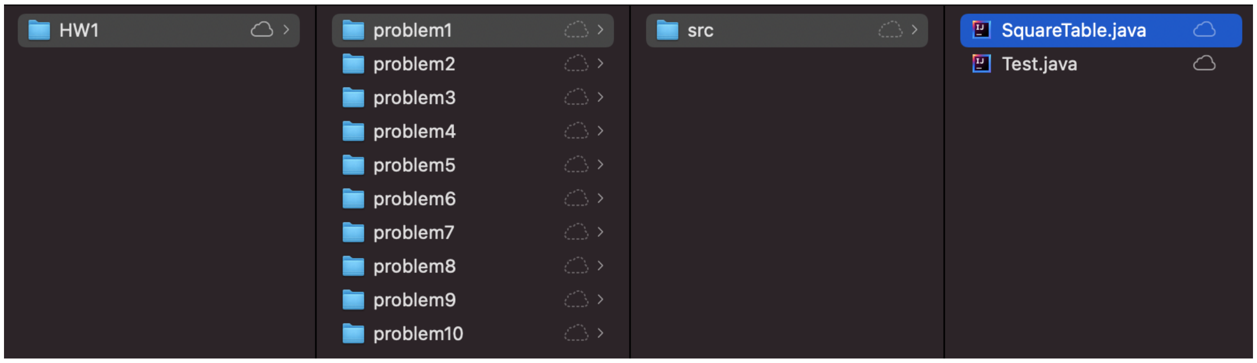
Assignment #1 [6 Marks]

Submission Date / Time	31 March 2022 23:59
Course	**[M1522.000600] Computer Programming**
Instructor	Jae W. Lee

- You are allowed to discuss with fellow students to understand the questions better, but your code must be **SOLELY YOURS**. You **MUST NOT** show your code to anyone else, or vice versa.
- We will use an automated copy detector to find plagiarism cases of the code among students. The copy checker is reliable so that it is highly likely to mark a pair of code as 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 also manually inspect the code. We reserve the right to request an explanation about the code for any suspicious case. We will ask detailed questions that cannot be answered if the code is not written by yourself.
- If any of such cases happens, you will get 0 marks for the assignment, and we may report this case to the department for an additional penalty.
- Download and unzip “HW1+SKELETON.zip” file from NeweTL. The file contains the skeleton codes for the 10 questions (in “/problemX” directory for Question #X).
- **Do not modify the overall directory structure**. Simply fill in the codes in appropriate files. It is okay to add new directories or files if needed.
- Do not use any external libraries. It is not allowed, either, to use JAVA Collection Framework.
- Assume all inputs are well formed. This means that you do not have to handle invalid inputs, and the trailing spaces will be ignored.
- Contact TAs if you have any questions.

Submission Guidelines

1. For submission, compress the entire “HW1” directory in 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.



3. Name the compressed file “20XX-XXXXX.zip” (your student ID).
4. Submit your code on the neweTL.
5. Double-check if your final zip file is properly submitted.
6. Note that you will get 0 marks for the wrong submission format.

Assignment 1 Overview.

1. The goal of this assignment is to implement simple Java programs.
2. All inputs and outputs are console inputs and outputs.
3. Questions 1-8 are worth 0.5 marks each. Questions 9 and 10 are worth 1 mark, respectively.

Question 1: Squares Table [0.5 Marks]

Objective: Write a program that prints out all squares that are less or equal to the input number N.

Note:

1. Outputs always start with **'1 times 1 = 1'**
2. Output format is **'k times k = k^2'**

Target Function: `public static void printSquareTable(int n)` (in SquareTable.java)
Input: int N (1 <= N <= 900)
Output: Square table corresponds to input number N.

Input	Output

`49`	`1 times 1 = 1`
	`2 times 2 = 4`
	`3 times 3 = 9`
	`4 times 4 = 16`
	`5 times 5 = 25`
	`6 times 6 = 36`
	`7 times 7 = 49`

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

Question 2: Sum of Fibonacci Numbers [0.5 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 <= 40)
Output: The first N Fibonacci Numbers from the smallest to the largest and the sum of those N Fibonacci Numbers. If the sum has more than five digits, print only its last five digits (including leading zeros).
e.g. For input 30, the actual sum is 1346268. Output to print is 46268.
For input 33, the actual sum is 5702886. Output to print is 02886.

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`

33	`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 832040 1346269 2178309`
	`sum : 02886`

Question 3: Drawing Figure [0.5 Marks]

Objective: Write a program that prints the appropriate pattern for a given number.

Note:

1. If there are two or more numbers in a row, there is a space between the adjacent numbers.
2. There is no space before the first number and after the last number in the longest row.
3. The length of every row (including spaces) is the same as the longest row. This means that each row except for the longest one has spaces after the last number.
4. Print consecutive number with **mod 10.**

Target Function: `public static void drawFigure(int n)` (in DrawingFigure.java)
Input: Integer N (1 <= N <= 100)
Output: The corresponding pattern as below.

Input	Output
1	`1`

2	`  1`
	`1 2 1`
	`  1`

3	`    1`
	`  1 2 1`
	`1 2 3 2 1`
	`  1 2 1`
	`    1`

7	`            1`
	`          1 2 1`
	`        1 2 3 2 1`
	`      1 2 3 4 3 2 1`
	`    1 2 3 4 5 4 3 2 1`
	`  1 2 3 4 5 6 5 4 3 2 1`
	`1 2 3 4 5 6 7 6 5 4 3 2 1`
	`  1 2 3 4 5 6 5 4 3 2 1`
	`    1 2 3 4 5 4 3 2 1`
	`      1 2 3 4 3 2 1`
	`        1 2 3 2 1`
	`          1 2 1`
	`            1`

12

```
                      1
                    1 2 1
                  1 2 3 2 1
                1 2 3 4 3 2 1
              1 2 3 4 5 4 3 2 1
            1 2 3 4 5 6 5 4 3 2 1
          1 2 3 4 5 6 7 6 5 4 3 2 1
        1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
      1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
    1 2 3 4 5 6 7 8 9 0 9 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 0 1 0 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 0 1 2 1 0 9 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 0 1 0 9 8 7 6 5 4 3 2 1
    1 2 3 4 5 6 7 8 9 0 9 8 7 6 5 4 3 2 1
      1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
        1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
          1 2 3 4 5 6 7 6 5 4 3 2 1
            1 2 3 4 5 6 5 4 3 2 1
              1 2 3 4 5 4 3 2 1
                1 2 3 4 3 2 1
                  1 2 3 2 1
                    1 2 1
                      1
```

Question 4: Character Pattern Search [0.5 Marks]

Objective: Write a program that

1. Takes an arbitrary string composed of alphabets.
2. Searches for the patterns of alphabets from the input string.
3. If found, prints the corresponding output which is the middle letter of the pattern.
4. Search for the following two patterns:
    - consecutive 3 alphabets that are all uppercase or all lowercase letters. (Reverse order like ‘CBA’ is not considered. Also, ‘zab’ are not consecutive letters.)
    - three same letters such that an uppercase letter is between two lowercase letters.

Pattern	Output
ABC	B
def	e
rRr	R

Target Function: public static void searchCharPattern(String str) (in CharacterPattern.java)
Input: String str (1 <= length <= 200)
Output: Characters that are the middle letters of the patterns. See an example below.

Input	Output
bcdt	c
qwABCDopPp	BCP
bbbaAababccHadDdAHbcbahHhAdcbbeHaAaH	AbDHA

Question 5: Hex Number Counter [0.5 Marks]

Objective: Write a program that converts a decimal number to hexadecimal number and counts the frequency of each hexadecimal digit (0-f).

Description:

1. There is an input decimal number in range 1 <= N <= 1000000.
2. Convert an input number to corresponding hexadecimal number.
3. Count the frequency of each number (0~f) from the hexadecimal number.
4. Print the counted frequency in the increasing order of the numbers. Do NOT print the numbers that have never been counted.
   (For number a~f, print letters **in lowercase**.)

Target Function: `public static void countHexNumbers(int n)` (in HexNumberCounter.java)
Input: input number as integer.
Output: Converted hexadecimal number and number counts with the increasing order of the numbers.

Input	Output
`7313`	`1c91`
	`1: 2 times`
	`9: 1 times`
	`c: 1 times`
`657265`	`a0771`
	`0: 1 times`
	`1: 1 times`
	`7: 2 times`
	`a: 1 times`

Question 6: Prime Numbers [0.5 Marks]

Objective: Write a program that prints prime numbers between two input numbers, m and n, inclusively.

Description: A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers (greater than 1). The smallest prime number is 2, and the second minority is 3. Print prime numbers in ascending order. If there is no prime number within the range, just print out nothing.

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

Target Function: `public static void printPrimeNumber(int m, int n)` (in PrimeNumbers.java)

Input: Natural number m and n (1 <= m < n <= 1000)

Output: Prime numbers in an ascending order.

Input	Output
5 29	5 7 11 13 17 19 23 29
200 202	

Question 7: Decreasing String [0.5 Marks]

Objective: Write a program that finds the longest substring where each alphabet in the substring is in a descending order.

Description: A substring is a contiguous sequence of characters within a string. Among substrings, there are decreasing substrings, where the alphabet on the left is always higher than the one on the right in the lexical order. Find the length of the longest decreasing substring. If a substring contains multiple identical alphabets, the substring is not a decreasing substring.

(e.g., Input: “abdfedccbgdcba” ⇒ Longest decreasing substring: “gdcba” ⇒ Output: 5)

(Note: “fedccb” is NOT the longest decreasing substring. A substring with the length of 1 is also considered a decreasing substring.)

Target Function: `public static void printLongestDecreasingSubstringLength(String inputString)` (in DecreasingString.java)

Input: A string that consists of alphabets in small letters. (1 <= input string length <= 200)

Output: Length of the longest decreasing substring.

Input	Output
abcdaeca	3
abdfedccbgdcba	5
abab	2
z	1

Question 8: Squared Matrix [0.5 Marks]

Objective: Write a program that squares (matrix multiplication with itself) the input matrix.

Note:

1. Print a row of matrices in one row.
2. If there are two or more elements in a row, there is a space between the adjacent elements.
3. There is no space before the first element and after the last element in the row.

Input Matrix	Output Matrix

`1 0 1`
`2 2 1`
`2 0 1`
	`3 0 2`
	`8 4 5`
	`4 0 3`

Description: Implement a method that prints out the squared matrix for a given matrix.

Target Function: `public static void printSquaredMatrix(int[][] matrix)` (in MatrixSquare.java)
Input: N, and a N by N size matrix composed of 2D int arrays. (1 <= N <= 10). All elements in the matrix should be positive numbers.
Output: Squared matrix.

Input	Output

`2`
`3 1`
`0 2`
	`9 5`
	`0 4`

`3`
`1 2 3`
`4 5 6`
`7 8 9`
	`30 36 42`
	`66 81 96`
	`102 126 150`

Question 9: Fractional Numbers [1 Mark]

Objective: Write a program that calculates the input expression with four fundamental arithmetic operations.

Description:

1. The input expression will be composed of two numbers and an operator.
2. Two numbers and an operator are separated with spaces.
3. The input number can be two types.
    a. Fractional numbers: two natural numbers with ‘/’ between them (ex. 2/3)
    b. Natural numbers (ex. 1, 2, 3, …) (1 <= N <= 999)
4. There are 4 operators: ‘+’, ‘-’, ‘*’, ‘/’
5. All input numbers are positive, but the result can be negative.
6. The result should be an irreducible fraction. (You can use FractionalNumber.gcd())
7. If the result is negative, put ‘-’ first. DO NOT make the denominator minus.
   (-2/3 ⇒ Good, 2/-3 ⇒ Bad)
8. If the result is a natural number, DO NOT use ‘1’ as the denominator.
   (3 ⇒ Good, 3/1 ⇒ Bad)

Useful Resources:

- String: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

Target Function: `public static void printCalculationResult(String equation)` (in FractionalNumberCalculator.java)

Input: Equation to solve.

Format: <Number><Space><Operator><Space><Number>

Output: Calculated result. The result should be an Irreducible Fraction.

Input	Output
`2/3 + 3/4`	`17/12`
`6/9 - 2`	`-4/3`
`1/2 + 1/2`	`1`

Question 10: Card Game Simulator [1 Mark]

Objective: Write a program that simulates a card game.

Description:

1. There are 20 unique cards with 5 numbers (1-5) and 4 letters (A,B,C,D).
2. Player A and B start the game with 10 cards each. (Input contains this information)
    a. Input consists of 2 lines.
    b. The first line indicates Player A's card list, and the second line indicates Player B's card list. (See Example Input and Output)
3. The two players take turns playing cards.
4. The players choose the card to play with the following rules.
    a. The game starts with Player A. Player A uses the card with the largest number. If there are multiple cards with the largest number, Player A will choose the card with priority A>B>C>D among the cards. (e.g. 5A, 5B, 5C -> choose 5A.)
    b. After that, if the next player has a card that has the same number as the previous player's, that card is used. If there are multiple cards with the same number, Player will choose the card with priority A>B>C>D among the cards. (e.g. 5B, 5C -> choose 5B.)
    c. If the next player does not have a card that has the same number as the previous player's but has the same letter card, the player will choose the same letter card with the largest number.
    d. If the next player does not have either the same number cards or the same letter cards, the current player wins the game.
    e. If the two players used all the cards, Player B wins.

5. Simulate the game by printing the sequence of the cards.
6. Print out the result message when the winner is decided.

Useful Resources:

- String: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html
- PrintStream: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/io/PrintStream.html
  (Type of System.out is PrintStream)

Target Function: `private static void simulateCardGame(String inputA, String inputB)` (in CardGameSimulator.java)

Input: The first String ⇒ Player A's card list

The second String ⇒ Player B's card list

Output: Simulated card game progress and result.

Input	Output

`1A 3D 4D 5A 4A 4C 3C 4B 5B 2A`
`5D 2D 1B 2B 3A 2C 5C 1C 3B 1D`
	`Player A: 5A`
	`Player B: 5C`
	`Player A: 5B`

	Player B: 5D Player A: 4D Player B: 2D Player A: 2A Player B: 2B Player A: 4B Player B: 3B Player A: 3C Player B: 3A Player A: 3D Player B: 1D Player A: 1A Player B: 1B Player A loses the game!
4A 1A 3B 5B 1B 4D 3A 2A 3D 5A 2D 4B 4C 1C 5C 5D 2B 2C 1D 3C	Player A: 5A Player B: 5C Player A: 5B Player B: 5D Player A: 4D Player B: 4B Player A: 4A Player B: 4C Player A loses the game!
1B 4D 3A 5C 4A 4B 5B 4C 3C 2A 2C 1D 3D 5A 3B 1C 1A 2D 5D 2B	Player A: 5B Player B: 5A Player A: 5C Player B: 5D Player A: 4D Player B: 3D Player A: 3A Player B: 3B Player A: 3C Player B: 2C Player A: 2A Player B: 2B Player A: 4B Player B loses the game!
1A 2A 3A 4A 5A 1B 2B 3B 4B 5B 1C 2C 3C 4C 5C 1D 2D 3D 4D 5D	Player A: 5A Player B: 5C Player A: 5B Player B: 5D Player A loses the game!