# CSCI 1110: Assignment 1

*Due: 2:00 pm, Monday, February 6, 2023*

The purpose of this assignment is to provide you programming practice with loops, arrays, 2D arrays and methods in Java. For each problem you will be provided with sample inputs and expected to generate corresponding outputs. All inputs are done via the console (unless specified otherwise) and all outputs are to be done to the console as well. You must submit your assignment via Codio, and you can test it by clicking the Check It button.

For the first part of this assignment, your task is to implement salary calculation for all employees in total with input in order. For the second part of the assignment, you need to improve the salary calculation function which accepts arbitrary input and requires specific output format.

## Problem 1: Salary Calculation

Suppose Willy's Wing World restaurant would like to develop a salary calculation application. You are going to help them design the basic logic. The main function is to calculate an employee's salary in total based on the input data. For example, Alice's normal working time is $t$ hours per week. Each week, she receives $d$ dollars for each hour of normal work; but if she works beyond $t$ hours, then she would receive $D$ dollars for each extra hour beyond $t$ hours. The same thing happened to every employee, and the output will be the salary for all employees.

Write the body of the program called **`Problem1.java`** whose main function computes how much the company needs to pay for all the employees in a week. Assume $t$, $d$, $D$, and $T$ are all non-negative integers. The input to the program is two lines: (1) the number of employees and (2) $t$, $d$, $D$ and $T$, respectively (the second line is separated by a space). Detailed information is given as follows.

### Input

The input has two parts

- The number of employees, and it is an integer number
- Working hour and payment data of each employee

The working hour and payment data contain the following information

- Normal working hour $t$
- Wages per hour of normal work $d$
- Extra wages per hour of extra work $D$
- Actual working hour $T$

The console will print "Please input the number of employees:". After inputting the integer number, the console will print "Please input the data of employees:". Then the data of employees should be given in the order of $t$, $d$, $D$ and $T$. See example below.

### Processing

Your program should be able to calculate the total salary the company needs to pay in a week. Employees have their own normal working hour, and regular wage. However, if their working hours exceed the normal working hour, extra wage will be paid to the employees.

### Output

The output will be printed in the console, which is the total salary of all employees in a week.

### Example

| Input | Output |
|---|---|
| `Please input the number of employees: 3`<br>`Please input data of employees:`<br>`40 10 20 45`<br>`35 15 18 20`<br>`40 20 10 40` | `1600` |

## Problem 2: Sophisticated Salary Calculation

Write a Java class called **`Problem2.java`** whose main function does the exact same thing as problem 1, except that the input and output formats are different. The detailed information is given as follows.

### Input

- The first line is the same as in Problem1
- The second line $t$, $d$, $D$ and $T$ are still given in a single line for each employee, but in arbitrary order. To avoid ambiguity, the name of each variable is given, e.g., $d = 10$. See below for an example. Also, the output is given in three lines: the middle line starts with the salary, a space, and then the word "Dollars". The first and the last line are filled with *symbol; the number of stars is equal to the number of characters in the second line

### Processing

Your program should perform the same task as in Problem 1 with the following modifications:

- Your program must handle the input data of employees in an arbitrary order, rather than in order as $t$, $d$, $D$ and $T$ in Problem 1
- You will need a class `StringBuilder` to construct a blank string builder with a capacity of 16 characters
  - The way to use it is: `StringBuilder str = new StringBuilder()`
  - You can use the method `str.append(s)` appends the mentioned String `str` with the existing String `s` or other types like boolean, char, int, double, float, etc.
- Your program should include a method called `digits` which calculates the number of digits in the salary in order to match the number of stars in the output

### Output

The output format is shown in the following example. There are three lines: (1) The first line is with stars (*); (2) The second line is the value of total salary plus one space plus the word "Dollars"; (3) The third line is also with stars. One thing needs to be noted is that the number of stars of the first line and the third line is equal to the number of characters in the second line.

## Example

| Input | Output |
|---|---|
| Please input the number of employees: 2<br>Please input data of employees:<br>T=45 d=10 t=40 D=20<br>t=35 D=20 T=5 d=15 | **********<br>575 Dollars<br>********** |

# What to Hand In

This assignment must be submitted in Codio via the Brightspace page.

# Grading

The assignment will be graded based on three criteria:

**Functionality**: "Does it work according to specifications?" This is determined in an automated fashion by running your program on a number of inputs and ensuring that the outputs match the expected outputs. The score is determined based on the number of tests that your program passes. So, if your program passes t/T tests, you will receive that proportion of the marks.

**Quality of Solution**: "Is it a good solution?" This considers whether the approach and algorithm in your solution is correct. This is determined by visual inspection of the code. It is possible to get a good grade on this part even if you have bugs that cause your code to fail some of the tests.

**Code Clarity**: "Is it well written?" This considers whether the solution is properly formatted, well documented, and follows coding style guidelines. A single overall mark will be assigned for clarity. Please see the Java Style Guide in the Assignment section of the course in Brightspace.

The following grading scheme will be used:

| Task | 100% | 80% | 60% | 40% | 20% | 0% |
|---|---|---|---|---|---|---|
| **Functionality (20 marks)** | Equal to the number of tests passed. | | | | | |
| **Solution Quality (20 marks)** | Approach and algorithm are appropriate to meet requirements for Problem1 and 2. | Approach and algorithm are appropriate to meet requirements for Problem 1, and some requirements for Problem 2. | Approach and algorithm are appropriate to meet requirements for Problem 1, but not Problem 2. | Approach and algorithm will meet some of the requirements of Problem 1. | Reasonable attempt made at solution for Problem 1. | No code submitted or code is not a reasonable one. |
| **Code Clarity (10 marks)** | As outlined below | | | | | |

## Code Clarity (out of 10)

- 2 marks for identification block at the top of code
- 2 marks for appropriate indentation when necessary
- 2 marks for use of blank lines to separate unrelated blocks of code
- 2 marks for comments throughout the code as needed
- 2 marks for consistent coding style throughout