NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
MOCK PRACTICAL ASSESSMENT II FOR
Semester 1 AY2024/2025

CS2030S Programming Methodology II

November 2024 Time Allowed 60 minutes

INSTRUCTIONS TO CANDIDATES

1. This practical assessment consists of **one** question. The total mark is **0**. **ALL** marks are given on the condition that reasonable efforts have been made to solve the given tasks. There is no mark for style but there is a penalty of up to **0 marks** for style.
2. This is a CLOSE BOOK assessment. You are only allowed 1 double-sided A4 cheat-sheet. No online resources/digital documents are allowed, except those accessible from the PE nodes (peXXX.comp.nus.edu.sg) (*e.g., man pages are allowed*).
3. You should see the following files/directories in your home directory.
   - `Q1.java`, `Q2.java`, and `Q3.java` are for you to edit and solve the given task.
   - `Test1.java`, `Test2.java`, and `Test3.java` are for testing each question.
   - `StreamAPI.md` and `Maybe.md`, for references to Java's Stream and `cs2030s.fp.Maybe` API.
4. Solve the programming tasks by editing `Q1.java`, `Q2.java`, and `Q3.java`. You can leave the files in your home directory and log off after the assessment is over. There is no separate step to submit your code.
5. Only the files directly in your home directory will be graded. Do not put your code under a subdirectory.
6. Some questions are to be answered in **functional-style**. There should be only a single statement, usually a `return` statement. No other statements are allowed including declaration of local variables. You should use lambda expression (*no anonymous class*) without declaring blocks (*including adding new methods*). You should not have conditionals and no loop.
7. You are not allowed to add any `import` statement.
8. Write your student number on top of EVERY FILE you edited as part of the `@author` tag. Do not write your name.
9. To compile your code, run `javac -Xlint:unchecked -Xlint:rawtypes *.java`. You can also compile the files individually if you wish.
10. You can run each test individually. For instance, run `java Test1` to execute `Test1`.
11. You do not have to check for style for the mock PA2.

IMPORTANT: If the submitted classes `Q1.java`, `Q2.java`, and `Q3.java` cannot be compiled, 0 marks will be given for the corresponding task. Make sure your program compiles by running

```
javac -Xlint:unchecked -Xlint:rawtypes *.java
```

before submission.

Q1. Consider a non-negative n. We can compute the next value of n with the following function.

```
int f(int n) {
  if (n % 2 == 0) {
    return n / 2;
  } else {
    return 3 * n + 1;
  }
}
```

The conjecture is that repreated application of `f` (i.e., `f(f(f(f(...(f(n)))))))`) will eventually reach 1 regardless of the initial value of n. We assume the conjecture true. Take for instance, starting from 3. We will get the following sequence: $3 \rightsquigarrow 10 \rightsquigarrow 5 \rightsquigarrow 16 \rightsquigarrow 8 \rightsquigarrow 4 \rightsquigarrow 2 \rightsquigarrow 1$. The number of steps to reach 1 is 7.

Implement the function `int collatz(n)` to find the number of steps needed to reach 1. Note that the function `f` above is not implemented and you cannot use conditionals to implement `f`.

```
public static long collatz(int n) {
  return 0L; // TODO
}
```

NOTE: Solve this problem in functional-style.

For the next two question, you are given two classes called `BusStop` and `Bus`. A `Bus` contains a list of `BusStop`. You can get the name of the `BusStop` using the method `getName` and you can get the bus stop number using the method `getNumber`. Study the two classes carefully.

Q2. The method `findBusStop` find and return the list of name of even numbered `BusStop` given a list of bus as they appeared in the list of buses. You are guaranteed that the bus stop numbers are unique and there is at least one bus stop satisfying the criteria. Convert the method into **functional-style**.

Q3. The method `longestOdd` find and return name of odd numbered `BusStop` with the longest name given a list of bus as they appeared in the list of buses. If there is no such bus stop, return `Maybe.none()`. Convert the method into **functional-style**.

END OF PAPER