**CPSC 326: Homework 5** *Due: Mar. 28*

**Goals:**

- Implement the various instructions supported by the MyPL VM.
- Practice using the instructions by writing two example programs.

**Instructions:**

1. Use the GitHub Classroom link (posted in Piazza) to copy the starter code into your own repository. Clone the repository in the directory where you will be working on the assignment.
2. For this assignment, you should have all the files you need for HW-5 from the starter code. You should not need to copy in any files from prior assignments.
3. Complete the `VM` implementation in `VM.java`. Be sure to go through the various files and ensure you understand the design of the VM (with frames, frame templates, instructions, etc.), helper functions, and so on.
4. Ensure your code passes the unit tests provided in `VMTests.java`.
5. Implement the two example programs using VM instructions within the files `HW5Prog1.java` and `HW5Prog2.java`. The goal is to create VM instructions for the corresponding MyPL programs given in the `.java` files, respectively. Note that enabling debugging via the `VM` class can help if you run into issues implementing the programs.
6. Ensure your two example programs work correctly (see below).
7. Create additional unit tests as specified in the TODO comment at the end of `VMTests.java`.
8. Create a short write up as a **pdf file** named `hw5-writeup.pdf`. For this assignment, your write up should provide a short description of the unit tests you created and any challenges and/or issues you faced in finishing the assignment and how you addressed them. The description of the tests can be short, but should state why you designed the tests the way you did (i.e., justify why the test is non-trivial / interesting, and what it is actually testing).
9. Submit your program by ensuring all of your code and writeup is pushed to your GitHub repo. You can verify that your work has been submitted via the GitHub page for your repo.

**Additional Requirements:** Note that in addition to items listed below, details will also be discussed in class and in lecture notes.

1. Additional tips and tricks for this assignment will be discussed in class. You will likely have a number of questions—be sure to post them early to piazza to give enough time for them to be answered and for you to understand the answers. You should also be monitoring piazza for the questions being asked (and the answers).
2. Be sure to look over the unit tests and ensure you understand what the various instructions are supposed to do. The unit tests, while not exhaustive, can be viewed as examples of what the instructions are supposed to do.

3. To run the example programs (that you need to write), first be sure to compile your code:

   ```
   mvn compile
   ```

   To run `HW5Prog1`, use the command:

   ```
   mvn exec:java@Prog1
   ```

   To run `HW5Prog2`, use the command:

   ```
   mvn exec:java@Prog2
   ```

4. `HW5Prog1` should implement the sum to primes program. Here is an example run of the program:

   ```
   mvn exec:java@Prog1
   [INFO] Scanning for projects...
   [INFO]
   [INFO] ------------------------< cpsc326:cpsc326-mypl >------------------------
   [INFO] Building cpsc326-mypl 1
   [INFO] --------------------------------[ jar ]---------------------------------
   [INFO]
   [INFO] --- exec-maven-plugin:3.3.0:java (Prog1) @ cpsc326-mypl ---
   Please enter integer values to sum (prime to quit)
   >> Enter an int: 4
   >> Enter an int: 10
   >> Enter an int: 9
   >> Enter an int: 5
   The sum is: 23
   Goodbye!
   [INFO] ------------------------------------------------------------------------
   [INFO] BUILD SUCCESS
   [INFO] ------------------------------------------------------------------------
   [INFO] Total time:  28.357 s
   [INFO] Finished at: 2025-03-16T13:17:40-07:00
   [INFO] ------------------------------------------------------------------------
   ```

5. `HW5Prog2` requires basic struct and array object creation and access. Note that you must implement the entire MyPL program. The result of running the program should be:

   ```
   mvn exec:java@Prog2
   [INFO] Scanning for projects...
   [INFO]
   [INFO] ------------------------< cpsc326:cpsc326-mypl >------------------------
   [INFO] Building cpsc326-mypl 1
   [INFO] --------------------------------[ jar ]---------------------------------
   [INFO]
   [INFO] --- exec-maven-plugin:3.3.0:java (Prog2) @ cpsc326-mypl ---
   ```

```
The average win percentage is: 0.7
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.202 s
[INFO] Finished at: 2025-03-16T13:18:32-07:00
[INFO] ------------------------------------------------------------------------
```

**Homework Submission and Grading.** Your homework will be graded using the files you have pushed to your GitHub repository. Thus, you must ensure that all of the files needed to compile and run your code have been successfully pushed to your GitHub repo for the assignment. Note that this also includes your homework writeup. This homework assignment is worth a total of 30 points. The points will be allocated according to the following.

1. **Correct and Complete (20 points).** Your homework will be evaluated using a variety of different tests (for most assignments, via unit tests as well as test runs using specific input files). Each failed test will result in a loss of 4 points. If 6 or more tests fail, but some tests pass, 6 points (out of the 30) will be awarded as partial credit. Note that all 30 points may be deducted if your code does not run, large portions of work are missing or incomplete (e.g., stubbed out), and/or the specified techniques, design, or instructions were not followed. Because assignments build on each other, in most cases you will need all tests to pass before moving to the next assignment.

2. **Evidence and Quality of Testing (5 points).** For each assignment, you must provide additional tests that you used to ensure your program works correctly. Note that for most assignments, a specific set of tests will be requested. A score of 0 is given if no additional tests are provided, 1–4 points if the tests are only partially completed (e.g., missing tests) or the tests provided are of low quality, and 5 if the minimum number of tests are provided and are of sufficient quality.

3. **Clean Code (2 points).** In this class, "clean code" refers to consistent and proper code formatting (indentation, white space, new lines), use of appropriate comments throughout the code, no debugging output, no commented out code, meaningful variable names and helper functions (if allowed), and overall well-organized, efficient, and straightforward code that uses standard coding techniques. A score of 0 is given if there are major issues, 1 if there are minor issues, and 2 if the "cleanliness" of the code submitted is satisfactory for the assignment.

4. **Writeup (3 points).** Each assignment will require you to provide a small writeup addressing challenges you faced and how you addressed them as well as an explanation of the tests you developed. Additional items may also be requested depending on the assignment. Homework writeups do not need to be long, and instead, should be clear and concise. A score of 0 is given if no writeup is provided, 1 if parts are missing, and 2 if the writeup is satisfactory.