CS 2110 Homework 01
Bases and Bitwise Operations

Max Everest

Summer 2023

Contents

1 Objective

1.1 Purpose

This assignment will test and strengthen your understanding of bitwise operators. Although data can represent anything from numerical values to English characters (ASCII), all digital data eventually boils down to bits. Thus, we can use bitwise operators to cleverly manipulate data programmatically. The concepts of bitwise operations that you apply here will be used throughout the remainder of this course, including in machine language, assembly language, and C programs.

1.2 Tasks

The homework is split into a Java coding portion worth 75%, and a docker checkoff worth 25%. Both parts are due **June 1st 11:59PM**. Afterwards, we will accept late submissions until June 2nd 11:59PM for a penalty of 25% of your overall grade.

1.3 Criteria

Your coding portion grade will be fully autograded. Your grade will be based on your methods returning the correct values **while following all the constraints listed at the top of each file.**

You may submit your code to Gradescope as many times as you like until the deadline. We will only grade your last submission. We have also provided a local checker that you can test your code with.

2 Docker Instructions

1. Please use the docker installation guide or the website to see the full instructions to download docker.
   (a) To access the pdf, go to the course page on canvas → 'Files' → 'Docker Resources' → 'Docker_Guide_2110.pdf'
   (b) Alternatively, you can use the docker website to see the same content: `https://gt-cs2110.github.io/cs2110docker/`
2. After installation, visit a TA after lab or during office hours and show them the GT logo background to get checked off (Background displayed at 'Deliverables')
3. If you encounter an issue, check Ed Discussion for pinned posts or other people with the same problems. Feel free to make your own Ed post if you weren't able to find a solution.

3 Coding Instructions

Your job is to finish implementing the methods within these files **under the constraints listed in the top of each file**:

1. `BitVector.java`
2. `Bases.java`
3. `Operations.java`

We recommend frequently running the local autograder or submitting to Gradescope as you progress throughout the assignment to verify that your implementations are correct and follow all the constraints.

An `Examples.java` file is also included which shows and explains examples of two methods similar to those used in your assignment. This is just provided for reference, so there are no tasks to be completed within it.

4 How to run the local auto-grader & verifier

1. Make sure that the `hw1checker.jar` file is in the same folder as your `Bases.java`, `BitVector.java`, and `Operations.java` files.
2. Navigate to this folder in your command line.
3. Run the desired command below.

4.1 Commands

1. Test all methods and verify that no banned operations are being used (checks all 3 files):

   ```
   java -jar hw1checker.jar
   ```

   Your grade will be dependent on the output of this command.

   On Windows and Mac, you can also double click the `hw1checker.jar` in your file explorer to test and verify all 3 files. The results will be placed in a new file called `gradeLog.txt`. Any errors with compilation, infinite loops, or other runtime errors will be placed in a new file called `errorLog.txt`.

2. Test & verify all methods in a single file. Useful when you're only intereted in the results for one file. For example, using `Bases.java`:

   ```
   java -jar hw1checker.jar -g Bases.java
   ```

3. Test all methods in a single file without running verifier. This means that this will only run the unit tests, and will not check for the use of banned operations. Useful for when you just want to try and get something that works. For example, using `Bases.java`:

   ```
   java -jar hw1checker.jar -t Bases.java
   ```

4. Verify all methods in a single file without running tests. For example, using `Bases.java`:

   ```
   java -jar hw1checker.jar -v Bases.java
   ```

5. Any combination of files can also be graded, tested, or verified at the same time. For example grading, `Bases.java` and `Operations.java` simultaneously:

   ```
   java -jar hw1checker.jar -g Bases.java Operations.java
   ```

4.2 Note for M1 Mac Users

When running the autograder, you may see some `UnsatisfiedLinkError`s above the autograder output with the message starting *"cannot open shared object file"*. Ignore them, they will not affect your score.

5 Deliverables

1. Please upload the following 3 files to the "Homework 1" assignment on Gradescope:
   (a) `Bases.java`
   (b) `Operations.java`
   (c) `BitVector.java`
2. Once you have successfully completed docker installation, you should be able to access a website like this....

(See 'Instructions (for Docker)' for more info on docker installation and how to visit this website)

6 Hints

6.1 Printing numbers in different bases

Remember that all numbers are stored in your computer as binary. When you perform operations such as `System.out.println()`, the computer does the translation into another base for you. All you need to do is tell the computer how you are representing your numbers, and how to interpret them.

For example, you can specify 16 in decimal, octal, or hexadecimal like so:

```
System.out.println(16);   // decimal (base 10), the default
System.out.println(020);  // octal (base 8), precede the number with a zero
System.out.println(0x10); // hexadecimal (base 16), precede the number with a "0x" (zero x)
```

You can also tell Java to print out your number in different bases using a method called `printf`. `printf` is the GRANDFATHER of all printing functions! When we get to C programming, you will be using it a lot. It is useful if you would like to write your own tester as well.

`printf` takes a variable number of arguments, the first of which is a format string. After the format string come the parameters. The formatting for the numbers is controlled by the format string.

6.1.1 Example

```
System.out.printf("In decimal: %d", 16);
System.out.printf("In octal: %o", 16);
System.out.printf("In hexadecimal: %x", 16);
```

The `%d`, `%o`, or `%x` get replaced by the parameter passed in. printf does not support printing the number out in binary.

For more information about `printf` read `http://en.wikipedia.org/wiki/Printf`.

6.2 Multiplication and division

You may find that there are times in which you need to use division or multiplication, but are not allowed to. Recall from lecture that you can use bitshifting to multiply or divide by powers of 2; this concept isn't found in the book, but is in the lecture slides.

7 Rules and Regulations

7.1 General Rules

1. All files you submit for assignments in this course should have your name at the top of the file as a comment for any source code file, and somewhere in the file, near the top, for other files unless otherwise noted.
2. When preparing your submission you should submit all three files you edited to Gradescope (either individually or in a zip archive).
3. Do not submit compiled files, that is .class files for Java code and .o files for C code. Only submit the files we ask for in the assignment.
4. Do not submit links to files. The autograder does not understand it, and we will not manually grade assignments submitted this way as it is easy to change the files after the submission period ends.

7.2 Submission Guidelines

1. You are responsible for turning in assignments on time. This includes allowing for unforeseen circumstances. If you have an emergency let us know **before** the due date and provide documentation (i.e. note from the dean, doctor's note, etc). No extensions will be made after the due date.
2. You are responsible for ensuring that what you turned in is what you meant to turn in. After submitting you should be sure to download your submission into a brand new folder and test if it works.
3. Make sure to start working on your assignments early, and keep track of all due dates. See the syllabus for information regarding late submissions.

7.3 Academic Misconduct

Please remind yourself of the academic misconduct policy listed in the syllabus. Academic misconduct is taken very seriously in this class. Quizzes, timed labs and the final exam are individual work.

Collaboration is limited to high-level discussion. Homework assignments will be examined using computer programs to find evidence of unauthorized collaboration.

Submissions that are copies that have been superficially modified to conceal that they are copies are also considered unauthorized collaboration.

You are forbidden to supply a copy of your homework to another student via electronic means. This includes uploading your code publicly to places such as pastebin or a public GitHub repository. Both of you will be charged.

7.4 Is collaboration allowed?

Collaboration is allowed on a high level, meaning that you may discuss design points and concepts relevant to the homework with your peers, share algorithms and pseudo-code, as well as help each other debug code. What you shouldn't be doing, however, is pair programming where you collaborate with each other on a single instance of the code. Furthermore, sending an electronic copy of your homework to another student for them to look at and figure out what is wrong with their code is not an acceptable way to help them, because it is frequently the case that the recipient will simply modify the code and submit it as their own.

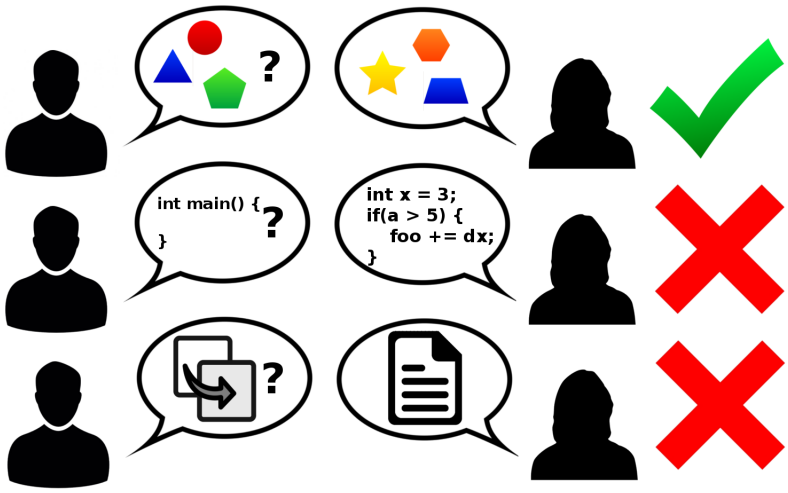

Figure 1: Collaboration rules, explained colorfully