CS 2110 Homework 01
Bitwise Operations

Vivian Thiebaut, Julia Grigni, Sameer Suri, Mudit Gupta

Fall 2020

Contents

1 Objective

1. To understand the bitwise operators
2. To use bitwise operators to complete tasks
3. To understand ASCII

2 Instructions

1. Make sure all 4 files are in the same folder:
   (a) `Bases.java`
   (b) `Operations.java`
   (c) `BitVector.java`
   (d) `hw1checker.jar`
2. Open a terminal / command prompt and navigate to the folder that all the files are in.
3. Run the following command to see your grade for `BitVector.java`:

```
java -jar hw1checker.jar -g BitVector.java
```

4. It should show all the test cases you are failing and give a 0/40 score.
5. Implement one of the functions in `BitVector.java` and re-run step 3 until you get full credit for that part of `BitVector.java`.

Now complete all the other methods in each of the 3 Java files (the details on how to implement each method is described in the comment above the corresponding method). Run the verifier and autograder frequently to avoid errors and to make sure you are using only the allowed operations.

Note: We have included an `Examples.java` file which shows and explains examples of two methods similar to those used in your assignment, which may be useful if you get stuck or confused at any point.

3 How to run the 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 (see below).

3.1 Commands

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

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

Note: Your grade will be dependent on the output of the above command, as it will both test the output of your methods, and verify that you are not using banned operations. If you get stuck though, you can use some of the below commands to help you debug.

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`.

Test & verify all methods in a single file (using `Bases.java`). Useful for when you just want to look at one file at a time. For example:

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

Test all methods in a single file without running verifier (using `Bases.java`). 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:

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

Verify all methods in a single file without running tests (using `Bases.java` for example):

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

Any combination of files can also be graded, tested, or verified at the same time. For instance, Bases and Operations can be graded simultaneously as follows:

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

4 Rubric

The grade the autograder gives you should be the same as the grade you get (unless you intentionally hardcode just the solutions or try to hack the autograder).

5 Deliverables

Please upload the following 3 files the "Homework 1" assignment on Gradescope:

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

The grade returned is the result of a series of test cases and may not be your final grade.

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 GRANDDADDY 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. Starting with the assembly homeworks, any code you write must be meaningfully commented. You should comment your code in terms of the algorithm you are implementing; we all know what each line of code does.
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. This means that (in the case of demos) you should come prepared to explain to the TA how any piece of code you submitted works, even if you copied it from the book or read about it on the internet.
3. Please read the assignment in its entirety before asking questions.
4. Please start assignments early, and ask for help early. Do not email us the night the assignment is due with questions.
5. If you find any problems with the assignment it would be greatly appreciated if you reported them to the author (which can be found at the top of the assignment). Announcements will be posted if the assignment changes.

7.2 Submission Conventions

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). You can create an archive by right clicking on files and selecting the appropriate compress option on your system. Both ways (uploading raw files or an archive) are exactly equivalent, so choose whichever is most convenient for you.

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.

7.3 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 **IN ADVANCE** of the due time supplying documentation (i.e. note from the dean, doctor's note, etc). Extensions will only be granted to those who contact us in advance of the deadline and no extensions will be made after the due date.

2. You are also 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. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via Canvas/Gradescope. Under no circumstances whatsoever we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Canvas/Gradescope.

3. There is a 6-hour grace period added to all assignments. You may submit your assignment without penalty up until 11:55PM, or with 25% penalty up until 5:55AM. So what you should take from this is not to start assignments on the last day and plan to submit right at 11:54AM. You alone are responsible for submitting your homework before the grace period begins or ends; neither Canvas/Gradescope, nor your flaky internet are to blame if you are unable to submit because you banked on your computer working up until 11:54PM. The penalty for submitting during the grace period (25%) or after (no credit) is non-negotiable.

7.4 Syllabus Excerpt on Academic Misconduct

Academic misconduct is taken very seriously in this class. Quizzes, timed labs and the final examination are individual work.

Homework assignments are collaborative, In addition many if not all homework assignments will be evaluated via demo or code review. During this evaluation, you will be expected to be able to explain every aspect of your submission. Homework assignments will also be examined using computer programs to find evidence of unauthorized collaboration.

What is unauthorized collaboration? Each individual programming assignment should be coded by you. You may work with others, but each student should be turning in their own version of the assignment. Submissions that are essentially identical will receive a zero and will be sent to the Dean of Students' Office of Academic Integrity. Submissions that are copies that have been superficially modified to conceal that they are copies are also considered unauthorized collaboration.

You are expressly forbidden to supply a copy of your homework to another student via electronic means. This includes simply e-mailing it to them so they can look at it. If you supply an electronic copy of your homework to another student and they are charged with copying, you will also be charged. This includes storing your code on any site which would allow other parties to obtain your code such as but not limited to public repositories (Github), pastebin, etc. If you would like to use version control, use github.gatech.edu

7.5 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.

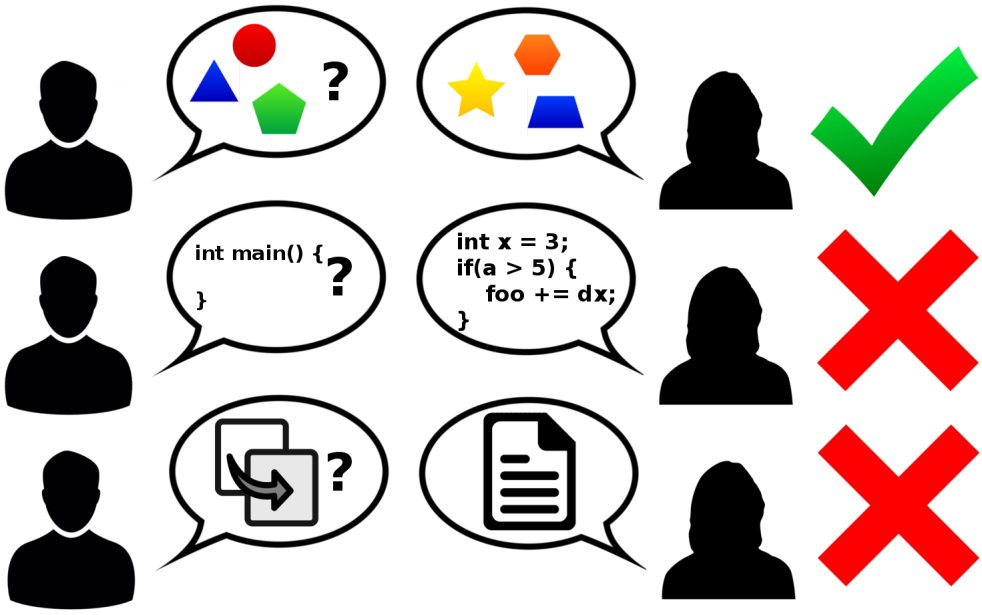

Figure 1: Collaboration rules, explained colorfully