CS2110 Summer 2016

Homework 2

Author: **Kevin Simon**

Rules and Regulations

General Rules

1. Starting with the assembly homeworks, Any code you write (if any) must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the 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.

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 may either submit the files individually to T-Square or you may submit an archive (zip or tar.gz only please) of the files (preferred). You can create an archive by right clicking on files and selecting the appropriate compress option on your system.

3. If you choose to submit an archive please don't zip up a folder with the files, only submit an archive of the files we want. (See **Deliverables**).

4. 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.

5. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

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 T-Square. When you submit the assignment you should get an email from T-Square telling you that you submitted the assignment. If you do not get this email that means that you did not complete the submission process correctly. 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 T-Square.

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 T-Square, 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.

Syllabus Excerpt on Academic Misconduct

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

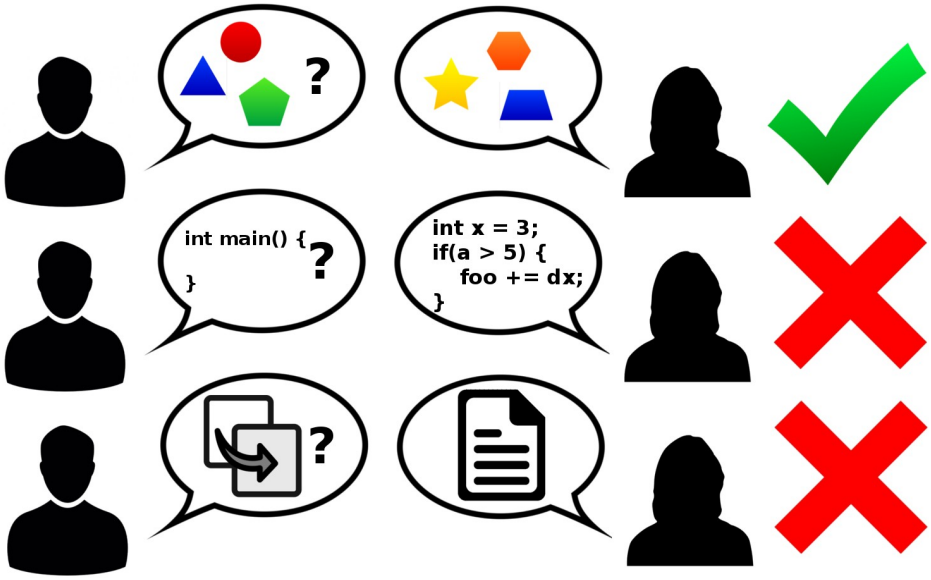
Homework assignments are collaborative, In addition many if not all homework assignments will be evaluated via demo or code review. During this evaluation, you will be expected to be able to explain every aspect of your submission. Homework assignments will also be examined using electronic computer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

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, as well as help each other debug code. What you shouldn't be doing, however, is paired programming where you collaborate with each other on a low level. 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, and it is often the case that the recipient will simply modify the code and submit it as their own.



Objectives

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

Overview

1. Three java files with method headers which you are to complete.

Resources

Begin by reading Chapter 2 of your text ("Introduction to Computing Systems"). Already read it? Read it again. No really, reading and understanding this chapter will prove to be a great advantage on this homework. It will help you understand several of the topics we will cover later in the course.

Coding

Time for some coding! Complete the methods given in the files HW2Bases.java, HW2BitVector.java, and HW2Operations.java. All of the methods have javadocs indicating what they should do, so make sure you read them carefully and follow the directions.

Each file has its own set of rules and limitations on what you may and may not use, so please look at the comments at the top of each file.

If you use anything that is banned, you will get no credit for that function. The point of this is to play with bit operations and learn how they work. You will not have all of these fancy Java API functions when we get to C programming. We have provided a verification program, HW2Verifier, which will notify you of code that breaks the guidelines. **We will run this on your submission and give no credit for functions that are found to use banned operations**.

Coding Guidelines for HW2

1. All bitmasks **MUST** be written in hexadecimal. This is the convention for masks and makes it very easy for the programmer to understand the purpose of the mask. If you write a mask in any other base you will lose points.
    - GOOD: (num & **0xFFE0**) >> 5 | (num & **0xF01F**) << 2 ^ **0x1**
    - BAD: (num & 65504) >> 5 | (num & 61471) << 2 ^ 1
    - Notice with the GOOD you can more easily tell what the line of code is doing.
    - Note the bolded parts are bitmasks!
2. Enter your name in the javadoc at the top of each file. Failure to do so will result in a point deduction!
3. Use HW2Tester to find out which functions worked part of the time, none of the time, or infinitely looped. The output only tells which functions need fixing, but a detailed log of what went wrong will be automatically generated as Results.log, which should be in the same directory as HW2Tester.
4. Use HW2Verifier to make sure none of your functions violate the banned operator guidelines.

Running HW2Verifier

HW2Verifier uses antlr4. The antlr4 runtime jar is sufficient, and can be downloaded here:

http://www.antlr.org/download/antlr-runtime-4.4.jar

Command Line

The HW2Verifier looks for your HW2 submission files (HW2Bases.java, HW2BitVector.java, HW2Operations.java) in the current working directory, and requires that the runtime jar be in your classpath to compile and run. Place antlr-runtime-4.4.jar in the same folder as the HW2Verifier files and HW2 submission files, and run:

```
javac -cp .:antlr-runtime-4.4.jar HW2Verifier.java
java -cp .:antlr-runtime-4.4.jar HW2Verifier
```

Note: This is UNIX syntax. The syntax is different for the Windows terminal, you have to change the colon to a semicolon. In other words, "-cp .;antlr-runtime-4.4.jar"

Eclipse

To set up HW2Verifier to run in Eclipse, make a new, separate Java project just for the verifier. Add HW2Verifier.java to the project's source folder (it will say there are errors at first, this is okay), and add your HW2 submission files to the project's root folder. The actual HW2 submission files must be in the root of the project folder; linking to them will not work. Right click the project, and go to Properties. Click Java Build Path, then the "Add JARs..." button, and add antlr-runtime-4.4.jar. You should now be able to run HW2Verifier.

If you do this, make sure you don't accidentally submit an empty copy of the HW2 template files! It's easy to accidentally do this if you copy the files for an Eclipse project. Likewise, it's also easy to get false negatives from the verifier if you copy the templates to the verifier's Eclipse project, and it runs by checking the empty templates every time instead of your actual submission files. The recommended way to set this up in Eclipse is to have one project for the verifier with your actual assignment files at the root of the verifier project, and another project for running the HW2 submission with **links** to those submission files which are located in the verifier's project, instead of separate copies.

Hints

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"
```

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.

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

(temporary link: http://www.cplusplus.com/reference/clibrary/cstdio/printf/)

Finally, 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.

Evaluation

1. Correctness of the output of your code
2. Implementation of code without banned operations
3. We will be running test cases of our own.

Deliverables

The file HW2Bases.java

The file HW2Operations.java

The file HW2BitVector.java

Once you have submitted the assignment, download the submission into a new directory and try running it to make sure you didn't forget anything. You alone are responsible for what you submit, and you risk getting a zero if you submit the wrong files, or forget some. **You have been warned!**