Homework 1

WORKING WITH NUMBER REPRESENTATIONS AND OPERATIONS

VERSION 1.0

The objective of this assignment is to get you comfortable with numbering systems, bitwise operations, and common binary operations. The programming assignment should be implemented in Java. You are required to work alone on this assignment. The assignment accounts for 7.5% towards your cumulative course grade.

This assignment may be modified to clarify any questions (and the version number incremented), but the crux of the assignment and the distribution of points will not change. Any there any changes to the assignment, all changes will be document in the "Change History" section of this assignment,

1 Assignment Description

The goal of this assignment is for you to learn how to convert between numbering systems and how binary operations work. A secondary goal is for you to continue to develop your programming skills. This program should be developed in Java version 8, and your classes must reside in the package "cs250.hw1".

Your program will be provided with 3 arguments at the command line. You are required to perform a set of operations over these arguments. These operations have been broken up into tasks: the points distribution for each task and the restrictions (and accompanying deductions) are specified in the grading section of this assignment.

We will compile and run your program using the commands below, replacing "<num1>", "<num2>", and "<num3>" with numbers as described in the tasks. See the "Example Outputs" section for more examples on how your program will be run. Your code must compile and run on the Linux machines in the CSB 120 computer lab, using Java version 8 (the default on these computers). It is highly recommended to develop your code using VS Code, either directly on the machines or through the Remote SSH extension in VS Code. Other IDE's or text editors may not be supported by the instructors.

Command line compilation: `javac cs250/hw1/Operations.java`
Command line execution format: `java cs250.hw1.Operations <num1> <num2> <num3>`
Command line execution example: `java cs250.hw1.Operations 0x123 0b111 17`

Task 1:
The first task is to determine if the correct number of arguments have been provided. The arguments will be provided when the program is run, so they will be in the "args" of the main method. If the correct number of arguments (three) were provided, print a message stating so. Otherwise, print a message indicating the wrong number of arguments were provided, and terminate the program. See the "Example Outputs" section for the messages the program must print.

Note: the arguments checked must come from the "args" of the main method. Using the method "Scanner.nextLine()" (or a similar method) to read the numbers will result in a deduction for this task.

Task 2:
The second task is to identify the numbering system for each of the numbers provided. These numbers will either be binary, decimal, or hexadecimal. The order of the numbers is not guaranteed, so they can occur in any order. They will all be unsigned integers (i.e., positive, whole numbers). We will not test your code using multiple numbers of the same numbering system.

The following criteria must be used to determine which number system to use:

1. If the number starts with the "0b" prefix, it must be treated as a binary number. For example, the number **0b**101011001 is a binary number.
2. If the number starts with the "0x" prefix, it must be treated as a hexadecimal number. For example, the number **0x**159 is a hexadecimal number.
3. If the number does not start with either the "0b" or the "0x" prefix, it must be treated as a decimal number.

After determining which number system is used for a number, the program must print the number and its numbering system. See the "Example Outputs" section for the messages the program must print.

Note: this task does not perform any error checking. If the number "0b345" is given, using the above criteria, it should be identified as a binary number because it starts with the "0b" prefix.

Task 3:
The third task is to determine if the numbers are written correctly for their numbering format. The following criteria must be used to determine if a number is correct or not. The program must print each number and identify whether or not it is correct. See the "Example Outputs" section for the required messages. After checking all three numbers, if one or more of them are invalid, the program must terminate.

To determine if a number is valid, use the following criteria:

1. If the number is binary, after the "0b" prefix, the number must only contain 0's and 1's.
2. If the number is decimal, it must contain only the digits 0 through 9.
3. If the number is hexadecimal, after the "0x" prefix, the number must contain only the digits 0 through 9 and the letters "A" through "F", either in lowercase or uppercase.

Note: Task 4-8 can be completed using built-in Java methods and/or operators. You may use them to double check your answers (e.g., in a unit test), but you may not use them for any other reason in any task (even tasks 1-3). The use of these methods or operators will result in a 100% deduction for the tasks they are used in.

Task 4:
The fourth task is to convert each of the numbers into the other numbering systems. For example, the binary number must be converted into its equivalent decimal and hexadecimal formats. The program must print the original version of the number, the binary version, the decimal version, and the hexadecimal version. All binary numbers must start with the "0b" prefix, and all hexadecimal numbers must start with the "0x" prefix. See the "Example Outputs" section for an example of what these messages must look like.

You must implement the logic of the conversions yourself. You are allowed (and encouraged) to write your own methods which perform the conversions, and you may do so any way you like. However, using a Java built-in method to do the conversion is not allowed and will result in a **100% deduction** in any task they are used (even outside task 4). Some examples of banned functions are "Integer.parseInt()" and "Scanner.nextInt()". It is not possible for the instructors to find and explicitly ban all such functions, so any substantially similar methods that you did not personally write are also banned. If there is a question about whether a method is banned or not, please ask an instructor.

Hint: The n^{th} digit from the right in base b is b^{n-1}.

Task 5:
The fifth task is to compute the 1's complement of the binary form of each of the three numbers. For each of the numbers, your program must print the number in its original format, then the binary version of that number (without the "0b" prefix), then the 1's complement of that binary (without the "0b" prefix). See the "Example Outputs" section for the required messages to print.

You must implement the logic for 1's complement yourself. Using the "~" operator, or any Java built-in function, to implement this is not allowed and will result in a **100% deduction**. As with the banned methods of task 4, it is not possible for the instructors to find and explicitly ban all similar methods, but any substantially similar methods will carry the same penalty. If there is a question about whether a method is banned or not, please ask an instructor.

Recall**:** 1's complement is accomplished by flipping each bit in the binary representation. That is, a 1 becomes a 0, and a 0 becomes a 1.

Task 6:
The sixth task is to compute the 2's complement of the binary form of each of the three numbers. For each of the numbers, your program must print the original number (in its original format), then the binary version of the number (without the "0b" prefix), then the 2's complement of the number (without the "0b" prefix).

You must implement the logic for 2's complement yourself. Using "~num + 1", or anything substantially similar, is not allowed and results in a **100% deduction**. Any similar methods carry the same penalty. For questions about whether a method is banned or not, please ask an instructor.

Recall**:** 2's comp is accomplished by applying 1's comp then adding 1. See lecture slides

Task 7:
The seventh task is to compute the bitwise OR, AND, and XOR of the three numbers. Since these are bitwise operations, they must be performed on the binary form of the numbers. The program must print out each operation being performed followed by the result. See the "Example Outputs" section for how the program is required to print this information.

Recall that these operations, when performed using two numbers, behave as described below. Consider how these may change when working with three numbers.

- OR: If either of the values is a 1 the result is a 1 otherwise it is a 0.
- AND: If both values are 1 the result is 1 otherwise it is a 0.
- XOR: If either of the values are 1 and they are not both 1 the value is a 1 otherwise it is a 0.

You must implement the logic for these operations yourself. Using the "|", "&", or "^" operators, or anything substantially similar that you did not write yourself, will result in a **100% deduction**. For questions about whether a method is banned or not, please ask an instructor.

Task 8:
The eighth and final task is to compute the bitwise left and right shift of each of the three numbers. For both kinds of shifts, the binary form of each number should be shifted by two places.

Recall that these operations behave as described below. For the right shift, consider what should be produced when performing "1 >> 2".

- Left Shift: This is accomplished by appending an x number of 0's. See lecture slides
- Right Shift: Accomplished by moving each digit to the right x times. See lecture slides

You must implement the logic for these operations yourself. Using the "<<2", ">>2", or anything substantially similar that you did not write yourself, will result in a **100% deduction**. For questions about whether a method is banned or not, please ask an instructor.

2 Example Outputs

Notes:

- Capitalization and spaces do not matter.
- All numbers are printed in the order provided in the arguments.
- We will be using an automatic grading script, which will very strictly grade your program's output. To ensure you get full credit, please match the example outputs exactly. Variations in the format do not necessarily incur deductions, but can result in deductions if it is unclear what the output is.

Wrong number of arguments:
Command: java cs250.hw1.Operations 1

Output:
Task 1
Incorrect number of arguments have been provided. Program Terminating!

Invalid characters in the specified numbers:
Command: java cs250.hw1.Operations 15 0b1011 0xhello

Output:
Task 1
Correct number of arguments given.
Task 2
15=Decimal
0b1011=Binary
0xhello=Hexadecimal
Task 3
15=true
0b1011=true
0xhello=false

Example outputs where the program has been provided with the correct arguments leading to a successful run of the program:
Command: java cs250.hw1.Operations 15 0b1011 0xfa

Output:
Task 1
Correct number of arguments given.

Task 2
15=Decimal
0b1011=Binary
0xfa=Hexadecimal

Task 3
15=true
0b1011=true
0xfa=true

Task 4
Start=15,Binary=0b1111,Decimal=15,Hexadecimal=0xf
Start=0b1011,Binary=0b1011,Decimal=11,Hexadecimal=0xb
Start=0xfa,Binary=0b11111010,Decimal=250,Hexadecimal=0xfa

Task 5
15=1111=>0000
0b1011=1011=>0100
0xfa=11111010=>00000101

Task 6
15=1111=>0001
0b1011=1011=>0101
0xfa=11111010=>00000110

Task 7
1111|1011|11111010=11111111
1111&1011&11111010=00001010
1111^1011^11111010=11111110

Task 8
1111<<2=111100,1111>>2=11
1011<<2=101100,1011>>2=10
11111010<<2=1111101000,11111010>>2=111110

3 What to Submit

Use the CS250 *Canvas* to submit a single .tar or .zip file that contains your Java source code, plus a README.txt file. When the archive is opened, the directory structure should be as follows:

- cs250
  - hw1
    - Operations.java (and any other .java files needed for the program)
    - README.txt

The README.txt file must contain a description of each file submitted and any other information you feel that the TAs will need to grade your program.

Other files (such as the ".class" files from compiling your code) are allowed to be present and will be ignored.

Filename Convention: The archive file should be named as <FirstName>-<LastName>-HW1.tar. For example, if you are Cameron Doe and submitting for assignment 1, then the tar file should be named Cameron-Doe-HW1.tar. Note: Canvas sometimes adds an extra "-1", "-2", etc. to the file name, which is OK.

Tar File Command Format: `tar -cf <FirstName>-<LastName>-HW1.tar cs250`
Tar File Command Example: `tar -cf Cameron-Doe-HW1.tar cs250`

Zip File Command Format: `zip -r <FirstName>-<LastName>-HW1.zip cs250`
Zip File Command Example: `zip -r Cameron-Doe-HW1.zip cs250`

Note: we highly recommend using one of the commands above for creating your archive for submission. Creating an archive through your operating system's GUI (or another GUI program) may not zip the file correctly, potentially resulting in a deduction on the assignment.

4 Grading

The assignments must compile and function correctly on machines in the CSB-120 Lab. Assignments that work on your laptop on your particular flavor of Linux, but not on the Lab machines are considered unacceptable. The following commands can be used to compile your code, execute the program, and then archive your code either as a .tar or a .zip file (either format will be accepted).

Command line compilation: `javac cs250/hw1/Operations.java`
Command line execution format: `java cs250.hw1.Operations <num1> <num2> <num3>`
Tar File Command Format: `tar -cf <FirstName>-<LastName>-HW1.tar cs250`
Zip File Command Format: `zip -r <FirstName>-<LastName>-HW1.zip cs250`

This assignment will contribute a maximum of 7.5 points towards your final grade. The grading will also be done on a 7.5 point scale. The points breakdown is as follows:

1 point each for correctly performing Tasks 1, 2, 3, 4, 5, 6 and 7 (**7 points**)

0.5 point for Task 8

Deductions:
There is a 7.5-point deduction (i.e., you will have a zero on the assignment) if you:
(1) Build a GUI

There is a 1-point deduction if:
(1) The directories in the archive do not match the specification in the "What to Submit" section.
(2) The code does not compile on the Linux computers in the CSB 120 lab on Java version 8..
(3) The code does not accept the three numbers via the command-line arguments.
    a. For example, there is a deduction if the numbers are read using "Scanner.nextLine()".
(4) The code has an infinite loop and does not terminate.

Use of the following methods will result in a 100% deduction of the task points where used:
(1) Integer.decode (or the "decode" method from similar classes)
(2) Integer.parse (or the "parse" method from similar classes)
(3) Integer.toBinaryString (or the "parse" method from similar classes)
(4) Integer.toHexString (or the "parse" method from similar classes)
(5) Integer.valueOf (or the "parse" method from similar classes)
(6) Built in bitwise operations (~, ^, |, &, <<, >>)
    a. Note: Logical operations are allowed (!, ||, &&, etc.)

You are required to **work alone** on this assignment.

5 Late Policy

Please check the class policy on submitting late assignments. You are allowed to submit assignments up to 24 hours late with a 7.5% deduction, or up to 48 hours late with a 15% deduction.

6 Version Change History

This section will reflect the change history for the assignment (if needed) after the first public release of the assignment. It will list the version number, the date it was released, and the changes that were made to the preceding version. Any changes to the first public release are made to clarify the assignment; the spirit or the crux of the assignment will not change.

Version	Date	Comments