Christopher Fallas Aguero
CSIT313_01
HW5

1)

a) In the case of “if if else”, the ‘else’ would be associated with the second ‘if’. In the code, the first ‘if’ statement turns out to be false, meaning that it would go to the ‘else’ statement. But the print statement shows this to not be the case since it did not print the “else” statement. This is because the ‘else’ statement is associated with the second ‘if’ statement.

b) You would put the ‘else’ statement right after the ‘if’ statement you want to associate it with. In the code, I put the ‘if’ and ‘else’ statements in their own block to specify that these two statements are associated with each other.

2)

a) Java does have short-circuit evaluation. The code shows this because it prints out the else statement. If it didn’t have short-circuit evaluation, the program would crash when attempting to divide 5 by 0, which is the second part of the ‘if’ expression.

b) You would enforce full evaluation by first evaluating the expressions, and storing those results in boolean values. Then you will do the ‘if else’ statement with those boolean values. The code shows this, by having the boolean expression evaluated first and then the ‘if’ statement.

3)

a) In Java, there are labeled breaks. You label the block you want to break out from and when the break command gets sent, it breaks out from all the loops under the label.
In C, I used a flag to see if reached the last loop. If it did, it won’t be able to go into the print ‘if’ statement.

For the readability, I thought the goto statement was much easier to understand when compared to using a flag. This is because with the goto statement, you can easily see where the next execution will be, but with the flag, you have to know what the value means. But this is for C.
For Java, since it has labeled breaks, this was the easiest to read. Everything in the label block will be broken out of when the break is called. So in my opinion, in order of best readability to worst, it would be labeled breaks > goto > flags.