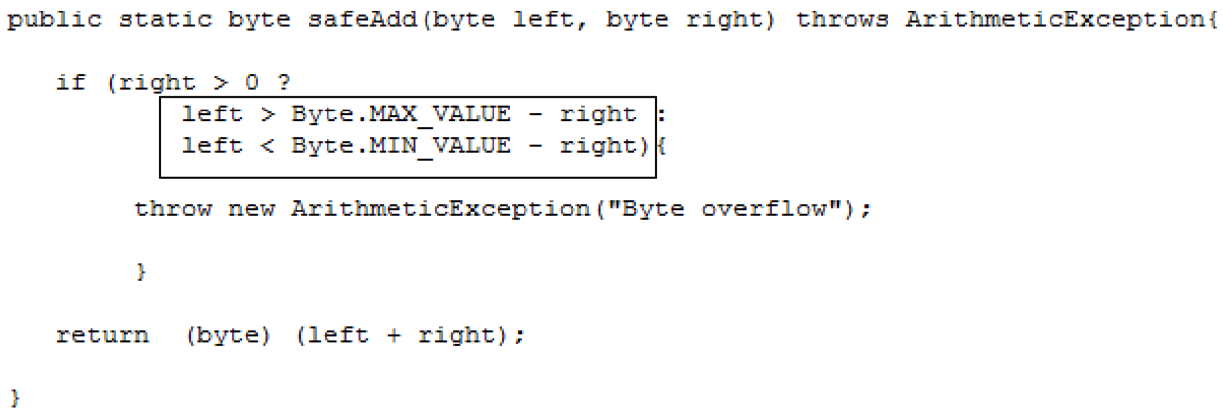
CSC 438/539 Systems and Software Security
Instructor: Dr. Natarajan Meghanathan

Sample Questions on Module 4 - Secure Coding Standards

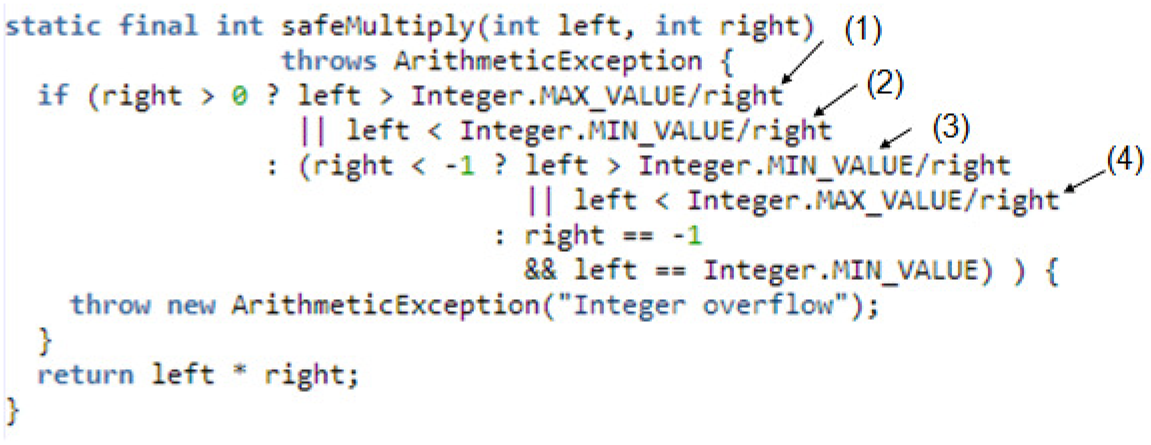
1) Consider the following Java code snippet for safe addition. Instead of doing like how it appears in the boxed portion of the code, can we check whether
left + right > Byte.MAX_VALUE and left + right < Byte.MIN_VALUE? Why or why not? Justify your answer.



```
public static byte safeAdd(byte left, byte right) throws ArithmeticException{

   if (right > 0 ?
          left > Byte.MAX_VALUE - right :
          left < Byte.MIN_VALUE - right){

        throw new ArithmeticException("Byte overflow");

        }

   return  (byte) (left + right);

}
```

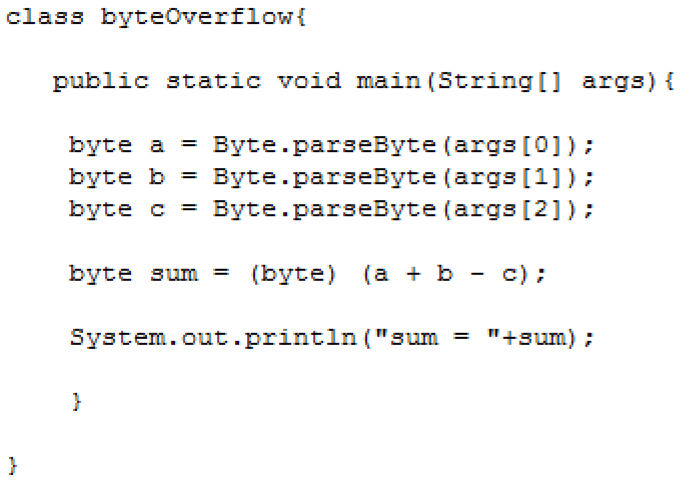
2) Using the safeAdd Java code given in question 1 as your reference, write the Java code to do safe subtraction of two byte variables *left* and *right*.

3) For what values of the int variables left and right would each of the inequalities (1) through (4) become true? Give an example for each case. For simplicity, assume Integer.MAX_VALUE = 127 and Integer.MIN_VALUE = -128.

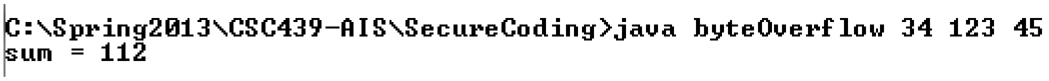


```
static final int safeMultiply(int left, int right)
                 throws ArithmeticException {
  if (right > 0 ? left > Integer.MAX_VALUE/right
                  || left < Integer.MIN_VALUE/right
                : (right < -1 ? left > Integer.MIN_VALUE/right
                                || left < Integer.MAX_VALUE/right
                              : right == -1
                                && left == Integer.MIN_VALUE) ) {
    throw new ArithmeticException("Integer overflow");
  }
  return left * right;
}
```

4) Consider the Java code below



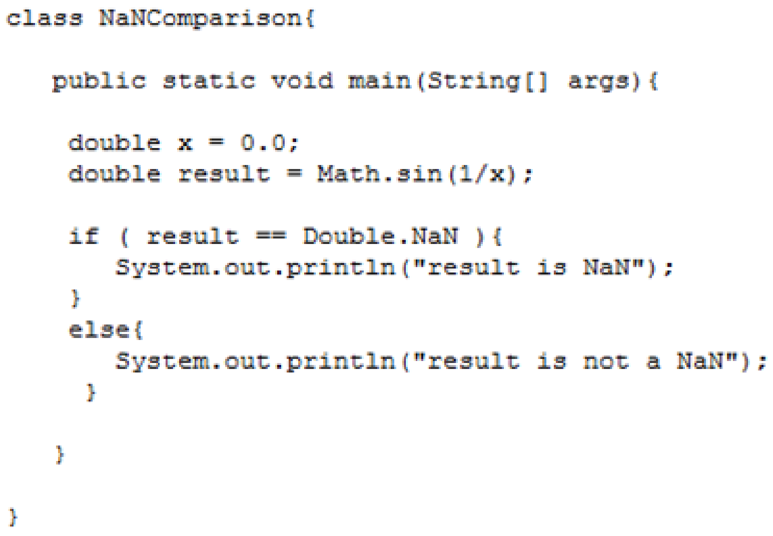
```
class byteOverflow{

   public static void main(String[] args){

    byte a = Byte.parseByte(args[0]);
    byte b = Byte.parseByte(args[1]);
    byte c = Byte.parseByte(args[2]);

    byte sum = (byte) (a + b - c);

    System.out.println("sum = "+sum);

    }

}
```



```
C:\Spring2013\CSC439-AIS\SecureCoding>java byteOverflow 34 123 45
sum = 112
```

Why is that we did not get an overflow error when we add a = 34 and b = 123? Explain.

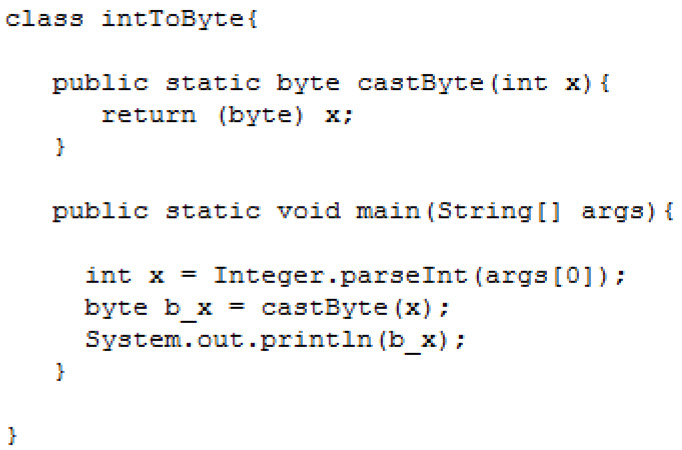
5) What is the problem with the following Java code? Modify it, if needed, so that it prints the appropriate error message when x = 0.



```
class NaNComparison{

   public static void main(String[] args){

    double x = 0.0;
    double result = Math.sin(1/x);

    if ( result == Double.NaN ){
       System.out.println("result is NaN");
    }
    else{
       System.out.println("result is not a NaN");
     }

   }

}
```

6) What is the problem with using floating point variables as loop counters? Explain with a simple Java code as example.

7) What is the output of this Java program for each of the following input values for x? Explain your answer.



```
class intToByte{

   public static byte castByte(int x){
      return (byte) x;
   }

   public static void main(String[] args){

     int x = Integer.parseInt(args[0]);
     byte b_x = castByte(x);
     System.out.println(b_x);
   }

}
```

(a) x = 140 (b) x = 158 (c) x = 30