BFST F2015, First-year Project: Map of Denmark
Exercises January 26, 2015: White-box testing

All code from the exercises can be found on the course web page.

Exercise 1

Given the following code for computing the factorial of positive integers.

```
int factorial(int n) throws BadUserException {
  if (n < 0) throw new BadUserException()
  else {
    int res = 1;
    for (int i = 1 ; i <= n ; i++) {
      res = res * i;
    }
    return res;
  }
}
```

For each of the following coverage criteria, construct a expectancy table and coverage table using as few test cases as you can.

1. Statement Coverage (start by labelling statements)
2. Branch Coverage (start by labelling branches)
3. Path Coverage (using "zero, one, or many" for the loop)

Exercise 2

Your company has been hired to write a navigation system for ancient Rome. One of your clever colleagues wrote the following program for figuring out what the house numbers mean. He is pretty sure it works... but you decide to test it anyway. (the source code is on the learnIT page).

```
/* Parses generalized Roman numerals. Allows more than three Cs, Xs, and Is in a row */
public class Roman {
  public static String[] test = {"IX","XII","MMXII","MCM","MCEinar","MDCLXVI"};
  public static Integer[] answer = {9,12,2012,1900,null,1666};

  public static void main(String[] args) {
    for (int i=0 ; i < test.length ; i++) {
      System.out.println(fromRoman(test[i]) + "=" + answer[i] + "?");
    }
  }

  public static Integer fromRoman(String x) {
    x = x + "$";
    int value = 0;
    int pos = 0;
    while (x.charAt(pos) == 'M') { value += 1000; pos++; }
    while (x.charAt(pos) == 'D') { value += 500; pos++; }
    if (x.charAt(pos) =='C') {
      if (x.charAt(pos+1) == 'M') { value += 900; pos += 2; }
    }
    while (x.charAt(pos) == 'C') { value += 100; pos++; }
    while (x.charAt(pos) == 'L') { value += 50; pos++; }
    if (x.charAt(pos) == 'X') {
      if (x.charAt(pos+1) == 'C') { value += 90; pos += 2; }
    }
    while (x.charAt(pos) == 'X') { value += 10; pos++; }
    while (x.charAt(pos) == 'V') { value += 5; pos++; }
    if (x.charAt(pos) == 'I') {
      if (x.charAt(pos+1) == 'X') { value += 9; pos += 2; }
    }
    while (x.charAt(pos) == 'I') { value += 1; pos++; }
    if (pos < x.length() - 1) return null;
    return value;
  }
}
```

1. Identify and label the branching points in the program.
2. Compute the number of paths through the program, using "zero, one, or many" for each loop.
3. Construct an expectancy and coverage table, using Branch Coverage as coverage criterion
4. Fix any bugs found in the code.

Exercise 3

After looking at map of ancient Rome, you realize that the house numbers are all under 1000. You therefore decide to do an automated exhaustive test.

1. Write a program that reads test cases from the file `roman.txt`, found on the course web page.
2. Run each of the test cases on `Roman.fromRoman(String)`.
3. Fix any bugs found in the code.

Bonus exercise

1. Write a program (or google your way to one) that generates roman numerals from a given integer.
2. Use this program to test `Roman.fromRoman(String)`, in the same way as in exercise 3, but by generating random test cases as needed.
3. Fix any bugs found in the code.