Assignment 6. Developing Loops

Due on Tuesday, 30 October, by midnight (submitted electronically). Note that each question will be graded on the following basis: Does the initialization make the invariant true? Does the invariant of the loop together with the falsity of the loop condition imply the result? Does the body make progress toward termination? Does the loop body maintain the invariant? Also, you methods MUST have specifications and your loops MUST have invariants.

You need execute and check out only the function of Question 1. However, it is in your best interest to check them all by running test cases.

Quiz on Thursday, 25 October. You will have to write this checklist from memory. To prove

```
// {Q}
init;
// {invariant: P;  bound function: t}
while (B) S
// {R}
```

do the following.

1. Prove that P is true initially: {Q} init {P}
2. Upon termination R holds: P and !B imply R
3. Each iteration decreases t.
4. t is really a bound function: P and B imply t > 0.
5. P is invariant: {P and B} S {P}

Remember that a Hoare triple like {Q} init {P} is a true-false statement with meaning:

Execution of init begun in a state in which Q is true is guaranteed to terminate with P true.

Quiz on Tuesday, October 30. Memorize and be able to use the definition of "Big-Oh" notation given at the bottom of page 160 of Weiss:

Definition. Function t(n) is O(f(n) if there are positive constants c and n_0 such that

$t(n) <= c*f(n)$ for all $n >= n_0$.

Quiz on Thursday, November 1. Be able to write algorithms binary search, partition, and selection sort. Memorize each by (1) memorizing the pre- and post-conditions, (2) memorizing the invariants, and (3) practicing developing the loop (with initialization) using the pre- and post-conditions and the invariant.

All the information you need is given in the handout on correctness.

In order to submit your assignment: Get a fresh copy of file RecursionTest.java and JLiveWindow from the online handout of assignment 1. Insert into file RecursionTest.java the 5 methods that you are asked to write in this exercise. Make sure that method buttonPressed is set to test the answer to Question 1!!!! Make a zipped folder that contains

(a) RecursionTest.java

(b) A file that gives your name and netid.

Submit the zipped folder.

Question 1 (a). In the recitation for the week of 22 October, we wrote an algorithm for storing in a String variable s the decimal representation of a nonnegative integer. Here, we ask you to write a function with the following specification:

```
// = the binary representation of nonnegative n
public static String binaryRep(int n)
```

A nonnegative integer n is written in decimal as the String $d_k d_{k-1} \ldots d_1 d_0$ for some k, where:

(1) Each d_i satisfies $0 <= d_i < 10$

(2) d_k, the most significant digit, is > 0.

(3) $n = d_k*10^k + \ldots + d_1*10^1 + d_0*10^0$

The decimal representation of 0 is "".

In the same way, n is written in binary as the String

$b_k b_{k-1} \ldots b_1 b_0$ for some k, where:

(1) Each b_i satisfies $0 <= b_i < 2$

(2) b_k, the most significant bit, is 1.

(3) $n = b_k*2^k + \ldots + b_1*2^1 + b_0*2^0$.

For example, the binary representation of 6 is the String "110" because

$6 = 1*2^2 + 1*2^1 + 0*2^0$.

Do this:. Change method buttonPressed so that it has this specification:

```
// Display in String field 0 the binary rep.
// of the integer in int field 0 and return null.
```

Check out your method binaryRep thoroughly.

Question 1 (b). Answer the following questions. Use the program of part 1 to help you.

1. What's the binary representation of 2^2, of 2^3, of 2^4? For n>0, what's the binary representation of 2^n?

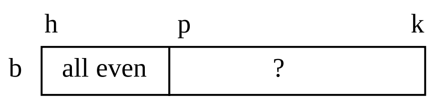
2. What's the binary representation of 2^4-1, 2^5-1, and 2^6-1? For n>0, what's the binary representation of 2^n-1?

Place your answers to Question 1(b) in file RecursionTest.java, as a comment AFTER method binaryRep.

Question 2. Write a method with this heading:
// = index of first odd value in b[h..k] (k+1 if none)
public static int firstOdd(**int**[] b, **int** h, **int** k)

The body should be a loop (with initialization), and it MUST use the following information:

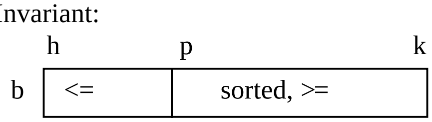
Precondition Q: h <= k+1
Postcondition R: h <= p <= k+1 and
b[h..p-1] contains only evens and
(either p = k+1 or b[p] is odd)
Bound function: k+1-p
Invariant:b[h..p-1] contains only even values



Question 3. Write a method with this heading:
// Sort b[h..k]
public static void selectionsort(**int** b[], **int** h, **int** k)

The body should be a loop (with initialization), and it MUST use the following information.

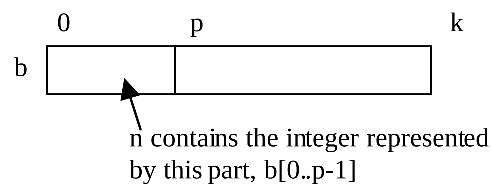
Precondition Q: h <= k+1
Postcondition R: b[h..k] is in ascending order
Bound function: p-h
Invariant:



Question 4. Write a method with this heading:
// = the integer whose binary rep is b[0..k-1]
public static int intBinary(**int**[] b, **int** k) {

Thus, array segment b[0..k-1] contains the bits of the binary representation of an integer n. For example, if b = (0, 0, 1), the integer is 4. The method body should have a loop, and the loop MUST use the following information -- note the use of two extra variables, p and exp.

Precondition Q: k >= 0,
0 <= b[j] <= 1 for all j in the range 0..k-1
Postcondition R: n is the integer represented by
b[h..k-1]
Bound function: k-p
Invariant: exp = 2**p and
n is the integer represented by b[h..p-1]:



Question 5. Write a method with this specification:
// = gcd(a,b), for a>0, b>0
public static int gcdloop(**int** a, **int** b)

The method body should have a loop (with initialization) that stores into a variable p the greatest common divisor of positive integers a and b (which we write as gcd(a,b)). You must use the following information:

Precondition Q: a > 0 and b > 0
Postcondition R: p = gcd(a,b)
Bound function: p+q
Invariant:gcd(a,b) = gcd(p,q)

Use the following facts:

gcd(x,x) = x (for all x)
gcd(x,y) = gcd(x-y,y) for x > y
gcd(x,y) = gcd(x,y-x) for y > x