# CSC236, Summer 2005, Assignment 3

Due: Thursday July 14th, 10 am

Danny Heap

## Instructions

Please work on all questions. Turn in the outline and structure of a proof, even if you cannot provide every step of the proof, and we will try to assign some part marks. However, if there is any question you cannot see how to even begin, leave it blank and you will receive 20% of the marks for that question.

Be sure to give full credit to any sources you consult (other than course notes, TAs, and the instructor) in preparing this problem set. If you try to pass off somebody else's work as your own for credit, you are committing an academic offense, and that can entail serious consequences. Any ideas that you do not attribute to someone else are assumed to be the ideas of the author(s) listed below, and will be evaluated for grading.

Write your name(s) and student number(s) (maximum of two names and two student numbers) in the space below.

Name ______________________________

Student # ______________________________

Name ______________________________

Student # ______________________________

The methods listed below may be examined and run by downloading `A3Examples.java` from the web page. If you have any questions about Java, please ask. You may assume (although it's not true) that the Java type `int` includes all integers.

1. Use the technique from lecture to prove the following method is correct with respect to its specification (precondition and postcondition).

```
/**
 * pow(b,p) returns b^p.
 * @param b an integer base to exponentiate.
 * @param p a natural number power to exponentiate to.
 * @return b^p.
 * Precondition: b is an integer and p is a natural number.
 * Postcondition: b^p is returned.
 */
public static int pow(int b,int p) {
  int bToTheP = 1;
  int i = 0;
  while (i != p) {
    bToTheP *= b;
    ++i;
  }
  return bToTheP;
}
```

2. Use the technique from lecture to prove the following method is correct with respect to its precondition/postcondition pair.

```
/**
 * recPow(b,p) returns b^p (more efficiently).
 * @param b an integer base to exponentiate.
 * @param p a natural number power to exponentiate to.
 * @return b^p.
 * Precondition: b is an integer and p is a natural number.
 * Postcondition: recPow(b,p) terminates, and when it does b^p is returned.
 */
public static int recPow(int b, int p) {
  int bToTheP = 1;

  if (p % 2 == 1) {
    bToTheP = b;
  }

  if (p < 2) {
    return bToTheP;
  }

  // p/2 is integer division
  return bToTheP * recPow(b, p/2) * recPow(b,p/2);
}
```

3. Use the technique from lecture to prove the following method correct with respect to its precondition/postcondition pair.

```
/**
 * div(m,n) returns quotient and remainder for m/n
 * @param m a natural number dividend.
 * @param n a positive natural number divisor.
 * @return [q,r] such that m = qn + r and 0 <= r < n.
 * Precondition: m is a natural number, n is a positive natural number.
 * Postcondition: [q,r] is returned and m = qn + r, 0 <= r < n.
 */
public static int[] div(int m, int n) {
  int[] quotRem = {0,0}; // initialized to [0,0]
  while (m != quotRem[0] * n + quotRem[1]) {
    if (quotRem[1] < n-1) {
      ++quotRem[1];
    }
    else {
      ++quotRem[0];
      quotRem[1] = 0;
    }
  }
  return quotRem;
}
```

4. Define $T(n)$ by:

$$T(n) = \begin{cases} c, & 1 \leq n < 5 \\ a_1 T(\lfloor n/5 \rfloor) + a_2 T(\lceil n/5 \rceil) + dn, & n \geq 5, \end{cases}$$

...where $a_1, a_2 \in \mathbb{N}$, $a_1 + a_2 = 5$, and $c, d \in \mathbb{R}^+$ (the positive reals). Prove there is some positive constant $\kappa \in \mathbb{R}$, such that $T(n) \leq \kappa n \log_5 n$, for all $n \geq 5$. You may NOT use the Master Theorem.