CSC236, Summer 2004, Assignment 4

Due: Thursday, July 29th, 10 pm

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 part marks. However, if there is any question you cannot see how to even begin, leave it blank you you will receive 20% of the marks for that question.

New policy: When you turn in your neat, finely-crafted, final version, indicate the question you feel you did the best work on by writing that question number on the front page of your assignment, and circling it. If we assign extra weight to any question, we will assign extra weight to the question you have circled.

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, and staple this page to the front of your assignment.

Name _____

Student # _____

Name _____

Student # _____

1. Use the first-order language of arithmetic defined in Exercise 1, Course Notes page 183, to construct the formulas below, assuming that the domain $D = \mathbb{N}$, the natural numbers. You are also free to use the additional formula $Prime(x)$, defined in Exercise 2, to express the predicate "x is prime."

   (a) Give a formula that expresses "there are infinitely many primes."

   (b) Give a formula that expresses "there are infinitely many composite numbers."

   (c) Give a formula that expresses "$x^2 + y^2 = z^2$."

   (d) Give a formula that expresses "there are k consecutive composite numbers."

   (e) Give a formula that expresses "Any natural-number power of 11 equals 1 mod 10."

2. State whether each formula is valid or not. Prove your claim.

   (a)
   $$\forall x \exists y \left(\forall y F(x,y) \rightarrow \exists x M(x,y)\right) \leftrightarrow \exists y \forall x \left(\forall y F(x,y) \rightarrow \exists x M(x,y)\right)$$

   (b)
   $$\forall x \exists y \left(\forall y F(x,y) \rightarrow \exists y M(x,y)\right) \leftrightarrow \exists y \forall x \left(\forall y F(x,y) \rightarrow \exists y M(x,y)\right)$$

   (c)
   $$\left(\forall x F(x,y) \vee \forall x M(y,x)\right) \leftrightarrow \forall x (F(x,y) \vee M(y,x))$$

   (d)
   $$\left(\forall x F(x,y) \wedge \forall x M(y,x)\right) \leftrightarrow \forall x (F(x,y) \wedge M(y,x))$$

3. Either prove the java code for `binSearch2` correct with respect to its precondition/postcondition pair, or provide a counter-example of input for which it fails.

```
/**
 * Precondition: int[] A is sorted in non-decreasing order
 * Postcondition: return smallest index 0 <= i <= A.length-1
 *                  such that A[i] >= n, or else return A.length.
 */
public static int binSearch2(int[] A, int n) {
  int f = -1, l = A.length;

  while (f != l-1) {
    int mid = (f + l) / 2;

    if (A[mid] >= n) {
      l = mid;
    }
    else {
      f = mid;
    }

  }

  return l;
}
```

4. Denote a ternerary digit as a TRIT, and an array of ternary digits as a TRIT CORE. In the (lamentably uncommented) methods below, methods `tritCore.tweak` and `tritCore.groupTweak` are defined, for managing trit cores. Either prove that `tritCore.groupTweak` correctly satisfies its postcondition for every valid input, or provide a counterexample of valid input for which it fails. You may assume, without proof, that `tweak` is correct with respect to its specification.

```
/**
 * Precondition: digit is an array of integers of length > 0 &&
 *                 from, to, intermediate are distinct elements of {0,1,2}
 * Postcondition: if (for some 0 <= k < digit.length
 *                      digit[0, ...,k-1] == intermediate && digit[k] == from) {
 *                      return true && set digit[k] = to}
 *                else {
 *                  return false;
 *                } &&
 *                no other element of digit is changed
 */
public static boolean tweak(int[] digit, int from, int to){
  for (int i = 0; i < digit.length; i++) {
    if (digit[i] == from) {
      digit[i] = to;
      return true;
    }
    if (digit[i] == to) { return false; }
  }
  return false;
}

/**
 * Precondition: digit is an array of integers of length >= n > 0 with values in {0,1,2} &&
 *                   to, from, intermediate are distinct elements of {0,1,2} &&
 *                   if (n>1) {digit[0, ..., n-1] == from}
 *                     else if (n==1) {for some 0 <= k < digit.length
 *                                        digit[0, .., k-1] == intermediate && digit[k] == from}
 * Postcondition:    if (n>1) {digit[0, ..., n-1] == to}
 *                     else if (n==1) {digit[k] == to}
 *                   all other elements of digit are unchanged && return true
 * */

public static boolean groupTweak(int[] digit, int n,
                                 int from, int to, int intermediate) {
  boolean status= true;
  if (n == 1) {
    status= tweak(digit, from, to);
  }
  else {
    status = status && groupTweak(digit, n-1, from, intermediate, to);
    status = status && groupTweak(digit, 1, from, to, intermediate);
    status = status && groupTweak(digit, n-1, intermediate, to, from);
  }
  return status;
}
```

5. Although `Vector` doesn't provide the most efficient implementation, `baseSort` outlines an $O(n)$ (yes, you read that correctly!) sorting algorithm. Either prove that the method `baseSort` correctly satisfies its postcondition whenever its precondition is satisfied, or provide a counter-example.

```
/**
 * Precondition: numList is a vector of Integers whose values are natural
 *                   numbers && base is an integer > 1
 * Postcondition: numList contains the same Integers as before,
 *                  with values in non-decreasing order
*/

public static void baseSort(int base, Vector numList) {
  boolean sorted = false;
  Vector[] digit = new Vector[base];

  for (int i = 0; i < base; i++) {
    digit[i] = new Vector();
  }

  int magnitude = 1;

  while (!sorted) {
    sorted = true;

    // sublist by current digit
    for (int i = 0; i < numList.size(); i++) {
      int currentInt = ((Integer) numList.elementAt(i)).intValue();
      int digitIndex = currentInt / magnitude;
      sorted = sorted && digitIndex < base; // or last iteration
      digitIndex %= base;
      digit[digitIndex].add(new Integer(currentInt));
    }

    // combine sublists
    numList.clear();
    for (int i = 0; i < base; i++) {
      numList.addAll(digit[i]);
      digit[i].clear();
    }

    magnitude *= base;
  }
}
```

6. Either prove that the method below satisfies its postcondition whenever its precondition is satisfied, or else exhibit a valid input for which it fails.

```
/**
  * Precondition: a and b are natural numbers
  * Postcondition: result[0] is the greatest common divisor of a and b &&
  *                result[0] = (a * result[1]) + (b * result[2])
  *
  * definition: d is the greatest common divisor of a and b if and only if:
  *             d is a natural number that divides both a and b, and
  *             any natural number that divides both a and b also divides d.
  */
 public static int[] MoreEuclid(int a, int b) {
   int[] result= {a,1,0};
   int tmpInt= 0;

   if (b != 0) {
     result = MoreEuclid(b, a % b);
     tmpInt= result[1];
     result[1] = result[2];
     result[2] = tmpInt - (result[1] * (a / b));
   }

   return result;
 }
```