Engineering 9867 Advanced Computing Concepts
Assignment #1

Due: Tuesday, March 12 at 0900

1. [10 points] Express the following in predicate logic, using the given predicate symbols and types.

   a) [3 points] There is a smallest integer.
   Predicates: $\leq$
   Types: **Integer**

   $\exists i : \textbf{Integer}, \forall j : \textbf{Integer}, i \leq j$

   b) [3 points] The array `A[N]` is bitonic. (An array is said to be *bitonic* iff the elements are in non-decreasing order in some initial portion of the array, and in non-increasing order for the remainder. For example, $[1, 1, 2, 3, 4, 4, 3, 2, 1]$ is bitonic, but $[1, 1, 2, 3, 4, 3, 4, 2, 1]$ is not.)
   Predicates: $<, \leq, >, \geq$

   $\exists i, 0 \leq i < \mathtt{N} \land (\forall j, 0 < j < i \rightarrow \mathtt{A}[j] \geq \mathtt{A}[j-1]) \land (\forall j, i < j < \mathtt{N} \rightarrow \mathtt{A}[j] \leq \mathtt{A}[j-1])$

   c) [4 points] The definition of "$\lim_{x \to a} f(x) = L$". (Hint: Quantify variables x, ϵ and δ over **Real** and relate $|f(x) - L|$ to ϵ and $|x - a|$ to δ.)
   Predicates: $<, \leq$
   Types: **Real**

   $$\forall \epsilon : \textbf{Real}, \left(\epsilon > 0 \rightarrow \exists \delta : \textbf{Real}, \begin{pmatrix} \delta > 0 \land \\ \forall x : \textbf{Real}, (0 < |x - a| < \delta \rightarrow |f(x) - L| < \epsilon) \end{pmatrix}\right)$$

2. [10 points] A *permutation* of an array is an array containing exactly the same values in another order, i.e.,

$$permuation(a,b) \stackrel{\text{df}}{=} \left(\begin{array}{l} length(a) = length(b) \wedge \\ \forall i, (0 \leq i < length(a) \rightarrow \left(\begin{array}{l} card(\{j \mid 0 \leq j < length(b) \wedge a[i] = b[j]\}) = \\ card(\{j \mid 0 \leq j < length(a) \wedge a[i] = a[j]\}) \end{array} \right) \end{array} \right)$$

Prove that the number of permutations of an array of length N is $N!$.

> Proof by natural induction. Let $perm(N) \stackrel{\text{df}}{=}$ the number of permutations of an array of length N.
>
> **Base case** $N = 1$ — $perm(1) = 1 = N!$.
>
> **Induction** Inductive hypothesis: $perm(N-1) = (N-1)!$
>
> Let $a_0, a_1, a_2, \ldots a_{N-2}$ denote the values in an array of length $(N-1)$ (in some canonical order). An array of length N contains one additional value, a_{N-1}. There are N possible positions for this in the array (i.e., at the beginning, following the first value, following the second value, ..., at the end). For each of these positions of a_{N-1}, a_0 through a_{N-2} may be in any of their possible orders, so
>
> $$\begin{aligned} perm(N) &= N \times perm(N-1) \\ &= N \times (N-1)! \qquad \text{by I.H.} \\ &= N! \end{aligned}$$

3. [15 points] In this question you are to reason about a C++ function `int gcd(int x, int y)` which returns the greatest common divisor of the natural numbers `x` and `y`.

   a) [5 points] Give the specification for this function. You may find it helpful to recall that any common divisor, d, of natural numbers x and y, will also be a divisor of the GCD of x and y. You may use the following predicate in your specification:
   $divisor(d, x) \stackrel{\text{df}}{=} (\exists q : \mathbf{int}, 0 < q \wedge x = d \times q)$

   pre: $\mathtt{x} \geq 0 \wedge \mathtt{y} \geq 0$

   post: $\mathtt{result} \geq 0 \wedge divisor(\mathtt{result}, \mathtt{x}_0) \wedge divisor(\mathtt{result}, \mathtt{y}_0) \wedge$
   $\forall i : \mathbf{int}, i \geq 0 \rightarrow (divisor(i, \mathtt{x}_0) \wedge divisor(i, \mathtt{y}_0)) \rightarrow divisor(i, \mathtt{result})$

   b) [10 points] Implement the function in C++ and add comments to your implementation to reason, as formally as possible, that it is correct. You may find it helpful to recall the property of natural numbers, that
   $\forall x, y : \mathbf{int}, (0 \leq x \wedge 0 \leq y) \rightarrow gcd(x, y) = gcd(y, x\%y)$

```
int
gcd(int x, int y)
{
  while (y > 0) {
    // INV: gcd(x, y) = gcd(x_0, y_0)
    // VAR: y
    int z;
    // gcd(y, x % y) = gcd(x_0, y_0)
    z = x % y;
    // gcd(y, z) = gcd(x_0, y_0)
    x = y;
    // gcd(x, z) = gcd(x_0, y_0)
    y = z;
  }
  return x;
}
```

4. [15 points] A *palindrome* is a string that is the same when read forward and backward. Some examples of palindromes are "ABBA", "radar" and "200202202002". In this question you are to reason about a C++ function `bool isPalindrome(const string& s)`, which returns `true` if `s` is a palindrome and `false` otherwise.

   a) [5 points] Give the specification for this function.

   pre: *true*

   post: $\texttt{result} = \forall i, (0 \leq i < \texttt{s.size()}/2 \rightarrow \texttt{s}[i] = \texttt{s}[\texttt{s.size()} - 1 - i])$

   b) [10 points] Implement the function in C++ and add comments to your implementation to reason, as formally as possible, that it is correct.

```
bool
isPalindrome(const string& s)
{
  bool result = true;
  int size = s.size();
  int i = 0;

  while (result && i < size/2) {
    // INV: result = (A)j, (0 <= j < i -> s[j] == s[size-1-j])
    // VAR: size/2 - i
    result = (s[i] == s[size-1-i]);
    i++;
  }
  return result;
}
```