Department of Computer Science and Engineering
University of California, San Diego

INDIVIDUAL ASSIGNMENT 1

(**assigned**: Jan. 10th; **due**: Wednesday **Jan. 17th**, **before class**, i.e., 4:40pm sharp)

- The URL of the **class web page** is `http://www.sdsc.edu/~ludaesch/CSE130/`. You will find important announcements, lecture notes, assignments, etc. there.
- There is a **web bulletin board** for CSE130 where you can ask your fellow students or the TAs (who also monitor the board) general questions about CSE130, e.g., organizational questions, technical problems, questions to clarify assignments etc. The URL is `http://www.sdsc.edu/~ludaesch/CSE130/board/`.
- The first **discussion section** will be in the second week.
- There will be *individual assignments* and *group projects*. The weights on the assignments and projects vary; the individual assignments and group projects together account for 50% of the grade, the *midterm* and *final* together for the other 50%.
- For this individual assignment prepare a *concise* (and readable) answer sheet and hand it in on Jan. 17th, **before class**. Write your **name** and **email** address on the front page.

Problem 1 (Data Types) A good way of thinking of *types* is as *sets of values* on which a common set of operations makes sense. Consider the following types

- **boolean** (values are "*true*" and "*false*")
- **char** (characters)
- **integer**
- **real** (aka **float**)
- **string**
- **record** (aka **structure** in C) over some types $T_1, \ldots, T_k$
- **array of** T
- **set of** T
- **associative array from** T_1 **to** T_2

where T and T_i denote arbitrary given types.

a) Name for each of the above types *one typical operation*, as well as one operation that does *not* make sense for this type (but for another of the above types). Also note whether each type is *primitive* or *composite*.

b) When seen as a mathematical object, composite types can be described by their *signature*. For example, given the type declarations

```
type Person = (Name, Age)
type Name = string
type Age = integer
```

a corresponding signature for Person would be the Cartesian product (**string** $\times$ **integer**).

What are the signatures for

- **string** (seen as a composite type)
- **associative array from string to integer**
- **array of records**, indexed by an enumeration type T_e, and where each record has a component holding a **real** and a **string**.

Problem 2 (Trees) Consider a composite type Tree, defined as follows: There are two types of tree nodes, called *leafs* and *inner nodes*, respectively. Leaf nodes have no children nodes and hold string values. In contrast, an inner node has a finite number of children nodes $c_1, \ldots, c_k$, each of which is of type Tree, and a "tag" (or "name") component of type **string** (denoting what kind of inner node it is).

a) Give the signature for Tree (and the other involved types like Leaf and InnerNode).

b) Define the type Tree in concrete syntax in C or Java. How is an instance of Tree created or updated (e.g., adding or deleting children) in your language of choice?