A SMALL PSEUDOCODE NOTATION PRIMER

This document is intended to aid you as a student in learning, understanding, and using the pseudocode notation used in the readings and assignments for the CSE 382 class.

The class is designed to be computer language agnostic. Because of this, it is necessary to be able to describe code in a language independent way. This document contains declarations and their associated definitions. There are a few examples of use when the examples may further illuminate the definitions.

1. Types

1.1. Value Types.

- *Integer*
- *Real*
- *Boolean*
- *acc* - an accumulator of any type, including a list.

1.2. Operator Types.

- $\bigoplus$ - a symbol representing an operator to be specified elsewhere.
- $\lceil\rceil$ - an operator representing the ceiling behavior.
- $\lfloor\rfloor$ - an operator representing the floor behavior.
- $+, -, \cdot, /$ - addition, subtraction, multiplication, and division as per standard mathematics.

2. Lists

2.1. Declaring Lists.

- [] - an empty list containing elements of any type.
- [] ::: *Type* - an empty list containing element of the specified type.
- $[a]$ - a list that may or may not be empty containing elements of any type.
- $[a]$ ::: *Type* - a list that may or may not be empty containing elements of the specified type.

2.2. **Building Lists.** Any example listed here can be used with lists of a specified type or lists of any type. The examples listed here use lists of any type to reduce verbosity.

- $x : [a]$ - creates a new list that is the result of prepending x to the non-empty list $[a]$.
- $x : []$ - creates a new list that is the result of prepending x to an empty list.
- $[a] : x$ - creates a new list that is the result of appending x to the non-empty list $[a]$.
- $[] : x$ - creates a new list that is the result of appending x to an empty list.
- $[a] : [b]$ - creates a new list by appending lists containing different elements.
- $[a] : [a]$ - creates a new list by appending lists containing the same elements.

2.3. **Reducing Lists.**

- $h \mid t$ - popping the first element of a list. h is the first element or head of the original list and t is a list consisting of the elements in the first list without the head in their original order.

3. Function Signatures

All function signatures follow the same pattern. They begin with the name of the function. This is separated from the set of parameters by two colons, ::. The parameter set is space separated. This is then followed by a right arrow representing an assertion of truth. The arrow points at the value of the function (in programmer terms, the return value).

$$function\_name :: list \rightarrow value$$

Examples:

- $length :: [a] \rightarrow Integer$ - a function named *length* with a single parameter that is a list that may or may not be empty. The value of the function is an $Integer$.
- $comparator :: x\ y \rightarrow r$ - a function called *comparator* that has two parameters and has a value of undetermined type.
- $in :: x\ [a] \rightarrow Boolean$ - a function called *in* that has two parameters, a value and a list that may or may not be empty and has a defined value type. In this case, $Boolean$.
- $b\_search :: a\ [b]\ (f :: x\ y \rightarrow Integer) \rightarrow c$ - a function called b_search that has three parameters, a value, a list that may or may not be empty, and a function named f that itself has two parameters and a value of $Integer$. The b_search function's value is of undeclared type.

4. Key Words

- *when* - a guard for execution of a code sub-statement or set of sub-statements.
- or, and, xor - as traditionally defined in logic.