CSCI 3155: Principles of Programming Languages
Exercise sheet #7
13th June 2007

Group name:_____

Pointers, References, Arrays

Exercise 1. Arrays are a fundamental part of many efficient algorithms and included in all major programming languages. While they tend to behave similarly even in different languages, there are a couple of smaller differences that we will examine in this exercise.

(a) (**Skill 8.1**) What is the difference between heap-dynamic and stack-dynamic arrays, and how can programmers take advantage of this difference?

(b) (**Skill 2.4**) Does MYSTERY support (non-fixed) stack-dynamic arrays? Explain.

Exercise 2. Consider the following MYSTERY program:

```
TYPE T0 = [1 TO 10];

VAR X : ARRAY T1 OF INTEGER;

VAR Y : ARRAY T1 OF INTEGER;

VAR Z : [5 TO 20];

TYPE T = [5 TO 20];

TYPE T2 = ARRAY [1 TO 15] OF T;

VAR AB : T2;

VAR AC : ARRAY [1 TO 15] OF T2;

VAR AD : ARRAY [1 TO 15] OF [5 TO 20];

VAR AE : ARRAY [1 TO 15] OF T2;

VAR W : INTEGER;

BEGIN
    W := 1;
    X[W] := W;      (* 0 *)
    X := Y;         (* 1 *)
    AB[1] := Z;     (* 2 *)
    AC := AD;       (* 3 *)
    AD[1] := AB;    (* 4 *)
    AE := AC;       (* 5 *)
END
```

Assume that `INTEGER` values are coerced to subrange types and checked at runtime.

(a) (**Skill 7.7**) Mark all the places in which type constructors construct a new name with distinct labels.

(b) (**Skills 7.7, 7.6**) Assume that `TYPE` declarations create transparent aliases (i.e., `TYPE` declarations are expanded before comparison). Further assume by-name equivalence for subrange types and structural equivalence for arrays. Which lines will trigger errors, and why? Use your labels to explain the reasons.

(c) (**Skills 7.7, 7.6**) You re-run with a different MYSTERY implementation. Now you observe type errors (only) in lines 3 and 4. Find a set of type equivalence rules that explains this behaviour.

Exercise 3. You are asked to develop a language for a real-time system (i.e., a language that must have predictable performance behaviour). The system must provide dynamic memory allocation with implicit heap management.

(a) (**Skill 9.4**) What approach do you choose, and why?

(b) (**Skill 9.4**) Describe the disadvantages of your decision, and give an example where your decision is inferior to the alternative.

Exercise 4. Not all languages use garbage collection: C, C++ and Pascal ask programmers to handle memory themselves.

(a) Argue in favour of explicit heap management, and give an example that demonstrates its superiority to implicit heap management (garbage collection).

(b) (**Skills 9.2, 9.3**) Using our criteria, argue in favour of implicit heap management and explain why it is superior to explicit heap management. Make sure to enumerate all problems that implicit heap management solves, by name, and describe them briefly.