CSCI 3155: Principles of Programming Languages
Exercise sheet #
2007

Group name:_____

This is mostly a lab exercise, using the PL Detective. Note that we have not yet made the system fully reentrant, so please do *not* submit multiple tries to the same account simultaneously.

To justify your observations, please note down the submission numbers that prove your point, as provided by the PL Detective.

Due to technical limitations, the exercise numbers provided by the PL Detective do not completely match up with the exercise numbers used here; e.g., you will have to submit to "Exercise 802: Exercise 8.2" where, of course, the right-hand side is correct.

Recall the revised PL Detective policy:

- You will lose 0.5 points on each exercise if you need a total of 8 or more submissions to determine the answer, and another 0.5 if you need 16 or more submissions. Syntactically incorrect submissions are not counted. Note that each experimental exercise is worth 2 points in total.
- Guest submissions are disabled for the duration of the exercise.

The PL Detective may give you feedback to some of the questions.

Exercise 1. Today's reading on subtyping restricted itself to subtyping rules for integers, subrange types, and set types. Subtyping can be applied to other types as well; in today's exercises, we will apply the concept to some types not discussed in the handout.

Note that a language can define subtyping rules in any way it wishes. However, we will focus on "natural" subtyping rules, i.e., subtyping rules in which the subtype denotes a strictly smaller set of values than the supertype; this is the standard approach taken by programming languages, and you can use this assumption to justify your decisions.

For the purposes of this exercise, assume fully structural type equivalence.

(a) Insert the subtype/supertype symbol to indicate which is the super/subtype, or an equals/not-equals sign if appropriate:

`[1 TO 10]`	`INTEGER`
`[2 TO 20]`	`[1 TO 20]`
`[2 TO 20]`	`[1 TO 21]`
`ENUM { A, B }`	`[0 TO 1]`
`ENUM { A, C, D }`	`ENUM { A, B, C }`
`ENUM { A, D }`	`ENUM { D }`

(b) (**Skill 10.2**) Use one of the above examples to illustrate what a *widening* conversion is.

(c) (**Skill 10.2**) Use one of the above examples to illustrate what a *narrowing* conversion is.

(d) (**Skill 10.2**) Which of the two conversions is potentially unsafe? Give an example.

Exercise 2. Before the advent of object-oriented programming, unions were a popular means for having variables that could have "different types".

(a) (**Skill 7.8**) What is the difference between a *tagged union* and an *untagged union*? Give an example.

(b) (**Skills 7.8, 7.1**) Analyse the two to determine whether they are compatible with the idea of strong typing, and explain.

Exercise 3. For this exercise, assume that the PL Detective has been configured to not use any kind of subtyping.

Further assume that the PL Detective *always* uses structural equivalence for its array index ranges, but may use different equivalence for its value range.

(**Skills 7.6, 7.7**) Determine the type equivalence rules used for arrays, subrange types, integers, and named types, and explain.

Exercise 4. For this exercise, assume that the PL Detective uses pass-by-value and structural type equivalence rules.

Many object-oriented languages allow implicit widening and/or narrowing conversions for parameter passing and/or assignment.

(**Skills 10.1, 10.2, 10.3**) Determine the following:

(a) What kind of implicit conversions (if any) does the PL Detective allow for assignments? Explain.

(b) What kind of implicit conversions (if any) does the PL Detective allow for parameter passing? Explain.

(c) A language with subtyping and subranges has two sensible choices for the types of integer literals: Either a universal `INTEGER` type, or the smallest possible subrange type for the integer literal.

Do your previous results allow you to determine which of the two the PL Detective uses? If so, figure it out and explain.

All submissions are counted against the same submission limit.