Programming Languages

August 19, 2001

Question 1

Programming Languages such as C do not provide automatic initialization of variables. This may lead to nasty errors. We would like to extend the language with the idea of unbound variables. Each variable created and not assigned is in a special state, called unbound. As soon as the variable is assigned, it loses its unbound status. We also introduce a special function to test whether a variable is unbound or not. The following is an example

```
int X;
int y;

y := 10;
if (unbound(X)) then
        ...
```

Use of unbound variables in expressions should lead to a run-time error. Address the following points:

[25 pts] Describe a reasonable way a general programming language (e.g., C) can implement these features (with emphasis on what can be done at compile time and what has to be done at run time); please do not make any assumptions regarding the underlying hardware platform.

[35 pts] Provide the denotational semantics for a simple imperative language with these features. The syntax is presented in the appendix A below. Make sure to provide all the components of the denotational semantics.

[10 pts] Discuss whether the same behavior (i.e., support the **unbound** test and generate exceptions when unbound variables are used in expressions) can be accomplished by simply providing automatic initialization of variables (as in Java).

Question 2

An *associative array* is an array indexed by strings. In languages like SNOBOL these arrays are declared with a fixed size, while in other languages, such as ABC, no limitations are imposed. This is a simple example (here we are not making any assumptions about size of the array):

```
assoc_array a;

a["enrico"] := 10;
a["good morning"] := a["enrico"] * 2;
```

[**10 pts**] Describe an *efficient way* to implement these features—i.e., a method that allows to quickly access the element of the array. Assume in this case that we are dealing with arrays of a specified maximum size.

[**20 pts**] Provide the denotational semantics in the case of array of unlimited size. In this case you need to provide *only*:

(a) a description of the semantic algebra that represents an associative array;

(b) the valuation function to process the declaration of the array;

(c) the valuation function to process the assignment of a value to an element of the array;

(d) the valuation function to process an occurrence of an array element within an expression.

Appendix A

Basic Grammar for the language:

Program	::=	*Declarations Command*
Declarations	::=	*Declarations* ; *Declarations*
		var *Identifier*
Command	::=	*Command*; *Command*
		Identifier **:=** *Expression*
		if *Expression* **then** *Command* **else** *Command*
		if (unbound(*Identifier***)) then** *Command* **else** *Command*
Expression	::=	*Identifier*
		Numeral
		Expression **+** *Expression*
		Expression ***** *Expression*