

CS-M61
CONCEPTS OF PROGRAMMING LANGUAGES
Attempt 2 questions out of 3

Question 1

- (a) Explain what distinguishes *simple variables* from *composite variables*.
[6 marks]

- (b) In some major programming language, give a concrete example of each of:
- (i) a declaration of a *composite variable*,
 - (ii) an assignment statement involving *selective update* of the variable declared in (i), and
 - (iii) an assignment statement involving *total update* of the variable declared in (i).
- [6 marks]

- (c) Specify an analysis of the following concrete ADA constructs in terms of the abstract constructs and coercions introduced in the lecture notes:
- (i) variable declarations of the form ' $I : T ;$ ', where I is an identifier and T is a type; and
 - (ii) assignment statements of the form ' $E_1 := E_2 ;$ ', where E_1 and E_2 are expressions.
- [6 marks]

- (d) In some major programming language, give a concrete example of a type declaration such that variables of the declared type can be used to store *selectively-updatable lists of integers*, without limiting the number of integers that can be stored. Give also code to create and initialise a variable to store the list whose components are the integers 1 and 2, with 1 as the head of the list.
- [4 marks]

- (e) Let $IntL$ be a set of values representing lists of integers, such that $IntL$ contains the empty list nil , and, for each N in Int and L in $IntL$, $IntL$ contains the list $cons(N, L)$.

Give a set equation satisfied by $IntL$. Can it have more than one solution? Justify your answer.

[3 marks]

Question 2

- (a) Explain four significant differences between the concepts of *binding* and *storing*. Illustrate your answer with concrete fragments of code in one or more major programming languages.

[8 marks]

- (b) Specify an analysis of the ADA numerical constant declaration:

```
const n: Integer := 42;
```

without the use of abstract constructs that involve variable creation or assignment. What would be the pros and cons of an alternative analysis of constant declarations: as declarations of initialised variables that cannot be updated?

[6 marks]

- (c) What does *The Correspondence Principle* state concerning declarations and parameter mechanisms? Give three instances of this principle in terms of abstract constructs.

[6 marks]

- (d) Explain the difference between *statically scoped* and *dynamically scoped* languages. Illustrate your answer by giving a concrete example of code involving procedure declarations and procedure calls that would have a different effect when executed if the language were to be dynamically scoped instead of statically scoped.

[5 marks]

Question 3

Consider the following ADA package declaration:

```
package Dates is

  type Date is
    record
      year: Integer;
      month: Integer range 1 .. 12;
      day: Integer range 1 .. 31;
    end record;

  function add (d: Date, i: Integer) return Date is
  begin
    ...
  end;

end package;
```

where ‘...’ is appropriate code to add *i* days to the date *d*.

- (a) Give a statement to declare and initialise a variable of type `Date` with a value representing the date 9th June 2006, followed by a statement to add 40 days to the stored date.

[6 marks]

- (b) Specify a complete analysis of the above package declaration in terms of the abstract constructs introduced in the lecture notes, indicating the analysis of the omitted procedure body by ‘ $\llbracket B \rrbracket$ ’. (Hint: subrange types of integers in ADA are analysed as $\text{range}(\text{Int}_1, \text{Int}_2)$, and function parameters in ADA are like ‘in’ parameters of proper procedures.)

[12 marks]

- (c) Explain the concept of *encapsulation*. Declare an *abstract type* in ADA that corresponds to the `Dates` package. The abstract type should encapsulate the definition of the type `Date`, and define a function `make` to create a value of type `Date` from arguments giving the year, month, and day of the month. You should provide the definition of `make`, but leave the body of `add` as ‘...’.

[7 marks]