# Exercises 2 (Values and types) – Solutions

**2A.** *(Primitive types)*

Primitive type for amounts of money up to £100,000.00 (represented as a multiple of £0.01):

(a) In C the safest choice would be `long`. (The type `int` would not have a sufficient range if the C compiler were to choose a 16-bit representation.)

(b) In Java the type `int` would be suitable. (A 32-bit representation is guaranteed.)

**2B.** *(Composite types)*

(a) Set of values and cardinality of each C type:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\text{SUIT} & = & \{0, 1, 2, 3\} & \#\text{SUIT} & = & 4 \\
\text{CARD} & = & \text{SUIT} \times \{0, 1, \ldots, 255\} & \#\text{CARD} & = & 4 \times 256 \\
\text{HAND} & = & \{0, 1, 2, \ldots\} \rightarrow \text{CARD} & & & \\
\text{OPTION} & = & \{0, 1\} & \#\text{OPTION} & = & 2 \\
\text{TURN} & = & \text{OPTION} \times \text{CARD} & \#\text{TURN} & = & 2 \times 1024
\end{array}$$

(b) Set of objects in the Java program:

$$\begin{array}{rcll}
\text{OBJECT} & = & \ldots & \text{(objects of predefined classes)} \\
& & + A\ (\text{INT} \times \text{FLOAT}) & \\
& & + B\ \text{BOOL} & \\
& & + C\ (\text{BOOL} \times \text{CHAR}) & \\
& & + \ldots & \text{(objects of other declared classes)}
\end{array}$$

(c) Modified set of objects in the Java program:

$$\begin{array}{rcll}
\text{OBJECT} & = & \ldots & \text{(objects of predefined classes)} \\
& & + A\ (\text{INT} \times \text{FLOAT}) & \\
& & + C\ (\text{BOOL} \times \text{CHAR}) & \\
& & + \ldots & \text{(objects of other declared classes)}
\end{array}$$

**2C.** *(Relationship between arrays and functions)*

(a) To implement the mapping $\{false \rightarrow true, true \rightarrow false\}$:

(i) Initialize an array $a$ such that $a[false] = true$ and $a[true] = false$. (This is simplest in C, with $false = 0$ and $true = 1$.)

(ii) Define a function $f$ : BOOL → BOOL, such that $f$ yields *true* when its argument is *false*, and yields *false* when its argument is *true*.

(b) To implement the factorial function over the integers 0 through 10:

(i) Initialize an array $a$ such that $a[0] = 1$, $a[1] = 1$, $a[2] = 2$, $a[3] = 6$, etc.

(ii) Define a function $f$ : INT → INT, using a loop or recursion, such that $f(0) = 1$ and $f(n) = n \times f(n–1)$ for $n > 0$.

(c) Arrays and functions are fundamentally different in that the mapping represented by an array is stored in its entirety (and therefore must be a finite mapping), whereas a function is applied to its arguments on demand (and therefore may be an infinite mapping).

**2D.** *(Type systems)*

For example, Python:

(a) Python's primitive types include INT, FLOAT, and STRING.

(b) Python's composite types are:

$$\begin{array}{rcl}
\text{TUPLE} & = & \{0, 1, 2, \ldots\} \rightarrow \text{VALUE} \\
\text{LIST} & = & \text{VOID} + (\text{VALUE} \times \text{LIST}) \\
\text{DICT} & = & \text{VALUE} \rightarrow \text{VALUE}
\end{array}$$

plus objects, which are tagged DICTs.

(c) In Python a recursive type can be defined by declaring a class with one or more instance variables of the same class.

(d) Python is dynamically typed.

**2E.** *(Static vs dynamic typing)*

[Outline answer:]

(a) Programs most easily implemented in a dynamically-typed language include those that manipulate data whose type cannot be predicted in advance (e.g., data mined from a website or web form). In such programs, some variables will contain data of unknown type.

(b) Programs most easily implemented in a statically-typed language include all those in which every data item has a type that can be predicted in advance. In such programs, all variables will contain data of known type.