Solutions

Each answer shows the steps, then the **result**.

2.1

(a) `LDM #105`: the operand is the value, so `ACC` = **105**

(b) `LDD 105`: the contents of address 105, so `ACC` = **27**

(c) `LDI 105`: 105 holds 27, so the contents of address 27, `ACC` = **91**

(d) `LDX 105`: the address is $105 + 2 = 107$, so `ACC` = **43**

2.2

- (a) **immediate**, (b) **direct**
- (c) **indirect**, (d) **indexed**

2.3

- a **fact** states something that is unconditionally true
- a **rule** states something that is true **only if** one or more other conditions are satisfied

2.4

(a) **low-level**

(b) **object-oriented**

(c) **declarative**

3.1

- in **indexed** addressing the effective address is the operand plus the index register, so the instruction holds the array's **start** address and `IX` holds the subscript
- incrementing `IX` moves to the next element with the **instruction itself unchanged**, so one instruction inside a loop can read the whole array
- in **relative** addressing the target is an **offset from the current instruction** rather than an absolute address
- so wherever the program is loaded in memory, every jump still lands in the right place and nothing has to be recalculated: the code is relocatable

3.2

- the operand is the **address of an address**: the processor reads the contents of the given address and uses **that** as the address of the value it wants
- so it follows the operand **twice** rather than once
- it is needed for **pointers**, and wherever the location of the data is not known until run time – for example following a linked list, or reaching a value through a table of addresses

3.3

(a) `animal(bella, dog).` and `owns(chen, bella).`

- `animal(milo, cat).` and `owns(rani, milo).`
- names are constants and so start with a **small** letter; any consistent predicate names are accepted, provided the same ones are used in (b)

(b) `dogowner(X) IF owns(X, Y) AND animal(Y, dog)`

- one mark for the two conditions, one for joining them with the shared variable `Y`, which is what links the owner to their animal

(c) `dogowner(W)`

- the engine returns `chen`, the only owner whose animal has species dog

3.4

- in an **imperative** program the programmer writes a **sequence of instructions** that are executed in order and change the program's state: they say **how** the task is done
- in a **declarative** program the programmer writes **facts and rules** – what is known and what is wanted – and no order of execution at all
- the language's **inference engine** then works out how to satisfy the goal, so the programmer says **what** is required rather than how to get it