Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

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

2.1

`f(5)` returns **6**

the global `x` is still **1**: output `6 1`

the `x` inside `f` is a new local name that shadows the global one

2.2

`s * 3`: `*` is not defined for a string and an int (or the result cannot go into an int)

the line and column, and the types found: “cannot multiply string by int”

2.3

```
t1 = b2 * 3
t2 = a1 + t1
count = t2
```

2.4

```
x = 6 + y
```

the multiplication is done once by the compiler, not every time the line runs

3.1

(a) `push a`, `push b`, `add`; `push c`, `push d`, `sub`; `mul`

(b) **3**, after `push d`

(c) `t1 = a + b`, `t2 = c - d`; `t3 = t1 * t2`

(d) the optimiser works on one simple form, and each machine needs only a back end for it

3.2

(a) **3**: `fact(3)`, `fact(2)`, `fact(1)`

(b) the return address and the parameters; the local variables and saved registers

(c) $100\,000 \times 64 = 6.4 \times 10^6$ B $= 6.4$ MB > 1 MB, so it overflows

3.3

(a) direct: $4 \times 3 =$ **12** compilers; IR: $4 + 3 =$ **7** parts

(b) direct: 3 new compilers; IR: 1 new front end

(c) it improves the code of every language on every processor at once

(d) features of one language may not fit the IR well, so some information or speed is lost

exercises.lol exercises.com.cn

(e) use a shared IR, ideally an existing one such as LLVM, which already has back ends and optimisers for common processors