Solutions

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

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

2.1

`0100`

4 moves, then it reads the blank and halts

2.2

every program is a finite string, so the programs can be listed P1, P2, … (countable)

the yes/no functions cannot be listed: flipping the diagonal gives a function not in any list

so there are more functions than programs, and most have no program

2.3

decidable: search the DFA's graph for a path to an accepting state

undecidable: the halting problem

decidable: run M for 1000 steps and see

2.4

"ever prints hello" is about what the program does, and some programs do and some do not (non-trivial)

so by Rice's theorem it is undecidable

3.1

(a) $D(M)$: run $H(M, M)$; if it says "halts", loop forever; if it says "loops", halt

(b) if $D(D)$ halts, then H said it loops, so it does not halt; if it loops, H said it halts, so it halts: a contradiction either way

(c) no such H exists: the halting problem is undecidable

(d) cell (i, j) says whether program i halts on input j; D flips the diagonal, so it differs from every row i at column i, and cannot be in the table

3.2

(a) P: run M on w, then print `done`; M itself must never print `done` (rename its output)

(b) a checker for "prints done" would answer the halting question for any M, w; halting is undecidable, so the checker cannot exist

(c) give approximate answers: sometimes say "not sure", or time out

3.3

(a) "ever sends contacts" is about what the app does, and is non-trivial, so by Rice's theorem it is undecidable

(b) a cautious static analysis flags every app that *might* leak: false alarms; testing apps on real phones only sees the paths it runs: missed leaks

(c) 500 flagged; 200 real leaks caught and **300** false alarms

(d) use the cautious checker, and send flagged apps to human reviewers; also block access at run time with permissions, so a missed leak still needs the user's consent

exercises.lol exercises.com.cn