Solutions

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

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

2.1

FIRST(E) = {`(`, `id`}

FIRST(E') = {`+`, ε}

FIRST(T') = {`*`, ε}

2.2

FOLLOW(E') = FOLLOW(E) = {`)`, `$`}

FOLLOW(T') = FOLLOW(T) = {`+`, `)`, `$`}

2.3

`E → T E'`, `E' → + T E'` | ε

`T → F T'`, `T' → * F T'` | ε

2.4

shift `id`, reduce F, reduce T

shift `*`, shift `id`, reduce F

reduce T → T * F, reduce E → T, accept

3.1

(a) E: `id`, `(` → T E'; E': `+` → + T E'; `)`, `$` → ε; T: `id`, `(` → F T'; T': `*` → * F T'; `+`, `)`, `$` → ε; F: `id` → id; `(` → (E)

(b) no cell has two rules

(c) `def Eprime(): if next == '+': match('+'); T(); Eprime(); else: return` (ε), which is right when next is `)` or `$`

3.2

(a) shift: `*` binds more tightly, so `E * E` must be built first

(b) reduce `E * E` first

(c) the grammar file declares precedence and associativity, e.g. `%left '+'` then `%left '*'`; the tool uses them to pick shift or reduce in each conflict

3.3

(a) on an error, skip tokens until a synchronising token such as `;` or `}`, pop the parser state back to a matching rule, and continue

(b) stop-at-first: "line 1: expected a number or `(` before `;`"; panic mode reports the same, skips to `;`, and parses line 2 without errors

(c) later messages may be caused by the first error, not by real mistakes, which confuses beginners

(d) show the first error clearly, with the line, column and the expected token; after recovery show at most two more, and mark them "may be caused by the first"

exercises.lol exercises.com.cn