Solutions

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

2.1

- one name for all the values, reached by index
- one loop processes them all, however many there are

2.2

- (i) **count**: add 1 when `rain[i] == 0`
- (ii) **maximum**, keeping its index

2.3

- e.g. which product sold best; or how much money was taken
- needs sales recorded per product; or the price of each item

3.1 C

- the array holds steps only; distance needs each step's length

3.2 B

- compares each value with the one **before** it, starting at 1 so no index is out of bounds; A reaches `d[d.length]`

3.3 B

- 20 students at one school do not represent all teenagers

3.4

(a)

- add every value in `loans` to a total
- divide the total by `loans.length`, as a `double`

(b)

```
int busy = 0;
for (int n : loans)
{
    if (n > 200) busy++;
}
```

- `busy` starts at 0
- traverses every element
- adds 1 when the value is over 200

(c)

- `loans` holds only the **total** per day, not the age of the borrowers
- the library must record each loan's borrower age or group (child/adult)