Solutions

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

2.1

```
int sum = 0;
for (int m : marks) sum += m;
double avg = (double) sum / marks.length;
```

- sum every element
- divide by `marks.length` with a `double` cast

2.2

- `{5, 8, 2}`

2.3

```
boolean allPositive = true;
for (int x : v)
{
    if (x <= 0) allPositive = false;
}
```

- starts `true`
- tests every element against 0
- sets `false` when one fails, and never back to `true`

2.4

- `{5, 4, 3, 2, 1}`
- $5/2 = 2$ in integer division: **2** swaps; the middle element stays

3.1 B

- starting at a real element is correct whatever the values are

3.2 C

- `pos` is overwritten by each match, so it ends as the **last** index holding 8

3.3 A

- 22/4 in **integer** division is 5

3.4

```
public static void shiftRight(int[] arr)
{
    int last = arr[arr.length - 1];
    for (int i = arr.length - 1; i > 0; i--)
    {
        arr[i] = arr[i - 1];
    }
    arr[0] = last;
}
```

- saves the **last** element before the loop
- loops **backwards** from `length - 1` down to 1
- copies `arr[i - 1]` into `arr[i]`
- puts the saved value at index 0

3.5

```
public static int bestRun(int[] sales)
{
    int best = sales[0] + sales[1];
    for (int i = 1; i < sales.length - 1; i++)
    {
        int pair = sales[i] + sales[i + 1];
        if (pair > best) best = pair;
    }
    return best;
}
```

- starts `best` at a real pair, e.g. the first
- loop bound keeps `i + 1` in range
- compares each neighbouring sum with `best` and updates it
- returns `best` **after** the loop