

Solutions

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

2.1

- target 4: returns the **first** match, **0**
- target 7: **-1**

2.2

- mid 4 (16): go left, high = 3; mid 1 (5): found
- returns **1**

2.3

- it must be **sorted**

2.4

- $100 \rightarrow 50 \rightarrow 25 \rightarrow 12 \rightarrow 6 \rightarrow 3 \rightarrow 1$ is 6 halvings, plus the last: **7**

3.1 A

- mid 3 (12), high = 2; mid 1 (6), high = 0; mid 0 (3), low = 1; low > high, stop

3.2 B

- binary search does not work on unsorted data; linear may check every element

3.3 C

- B and D return after checking only the **first** element; A compares references

3.4

(a)

```
public static int countAbove(int[] ids, int x)
{
    int count = 0;
    for (int id : ids)
    {
        if (id > x) count++;
    }
    return count;
}
```

- traverses every element
- tests `id > x` and counts
- returns the count after the loop

(b)

- low 0, high 7, mid 3 (125): $125 < 152$, low = 4
- low 4, high 7, mid 5 (140): $140 < 152$, low = 6
- low 6, high 7, mid 6 (152): found, returns **6**

(c)

- each comparison **halves** the part still to search
- so about **17** comparisons at most, against up to **100 000** for linear search