

Past-paper walk-through

2D Array Creation and Access ■ 4.11

eXercise

Questions in this set

- 2025 Q4
- 2024 Q4
- 2019 Q4
- 2014 Q3

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- **//** separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = “or words to that effect” – the wording need not match.

Question 4

2025 ■ Q4

This question involves reasoning about a number puzzle that is represented as a two-dimensional array of integers. Each element of the array initially contains a value between 1 and 9, inclusive. Solving the puzzle involves clearing pairs of array elements by setting them to 0. Two elements can be cleared if their values sum to 10 or if they have the same value. The puzzle is considered solved if all elements of the array are cleared. You will write the constructor and one method of the `SumOrSameGame` class, which contains the methods that manipulate elements of the puzzle.

```
k+kdpublic+w k+kdclass n+ncSumOrSameGame
p
+w      k+kdprivate+w k+ktinto[o]o[o]+w npuzzlep;

+w      c+cm/**
c+cm      * Creates a twodimensional array and fills it with random integers,
c+cm      * as described in part (a)
c+cm      * Precondition: numRows  0; numCols  0
c+cm      */
+w      k+kdpublic+w n+nfSumOrSameGamep(k+ktint+w nnumRowsp,+w k+ktint+w nnumColsp)
+w      p+w c+cm/* to be implemented in part (a) */+w p
+w      c+cm/**
c+cm      * Identifies and clears an element of puzzle that can be paired with
```

Question 4

(a) Write the constructor for the `SumOrSameGame` class. The constructor initializes the instance variable `puzzle` to be a two-dimensional integer array with the number of rows and columns specified by the parameters `numRows` and `numCols`, respectively. Array elements are initialized with random integers between 1 and 9, inclusive, each with an equal chance of being assigned to each element of `puzzle`.

When an element of the two-dimensional array is accessed, the first index is used to specify the row and the second index is used to specify the column.

Complete the `SumOrSameGame` constructor.

```
c+cm/**
c+cm * Creates a twodimensional array and fills it with random integers,
c+cm * as described in part (a)
c+cm * Precondition: numRows > 0; numCols > 0
c+cm */
k+kdpublic+w n+nfSumOrSameGamep(k+ktint+w nnumRowsp,+w k+ktint+w nnumColsp)
```

Question 4

puzzle before call to clearPair	Method Call	puzzle after call to clearPair	Return Value	Explanation
0 7 9 0 7 4 1 6 8 4 1 8	clearPair(0, 1)	0 0 9 0 0 4 1 6 8 4 1 8	true	The value 7 in row 0, column 1 is matched with the value 7 in row 1, column 0.
1 2 3 4 5 6 7 8 5 4 1 2	clearPair(2, 2)	1 2 3 4 5 6 7 8 5 4 1 2	false	There is no element in row 2 or a later row to pair with the value 1.
8 1 0 5 0 4 3 6 3 4 5 8	clearPair(1, 1)	8 1 0 5 0 0 3 0 3 4 5 8	true	The value 4 in row 1, column 1 is matched with the value 6 in row 1, column 3. It could also have been matched with the value 4 in row 2, column 1.
1 7 9 2 6 5 4 4 4	clearPair(0, 2)	0 7 0 2 6 5 4 4 4	true	The value 9 in row 0, column 2 is matched with the value 1 in row 0, column 0.

Question 4

2025 ■ Q4

This question involves reasoning about a number puzzle that is represented as a two-dimensional array of integers. Each element of the array initially contains a value between 1 and 9, inclusive. Solving the puzzle involves clearing pairs of array elements by setting them to 0. Two elements can be cleared if their values sum to 10 or if they have the same value.

The puzzle is considered solved if all elements of the array are cleared.

You will write the constructor and one method of the `SumOrSameGame` class, which contains the methods that manipulate elements of the puzzle.

```
k+kdpublic+w k+kdclass n+ncSumOrSameGame
p
+w      k+kdprivate+w k+ktinto[o]o[o]+w npuzzlep;

+w      c+cm/**
c+cm      * Creates a twodimensional array and fills it with random integers,
c+cm      * as described in part (a)
c+cm      * Precondition: numRows  0; numCols  0
c+cm      */
+w      k+kdpublic+w n+nfSumOrSameGamep(k+ktint+w nnumRowsp,+w k+ktint+w nnumColsp)
```