MODULE R EXAM TECHNIQUE · R6

The 15-mark scenario question: mark scheme

21 marks · Cambridge 0478 syllabus: Paper 2 scenario question (section 3, assessment details), 7.4 (standard methods: totalling, counting, finding maximum), 7.5(a) (validation checks), 7.8 (identify and correct errors), 7.9 (write algorithms using pseudocode or program code), 8.1.3 (input and output), 8.1.4 (selection, iteration, totalling, counting), 8.1.8 (maintainable programs: identifiers and comments), 8.2.1–8.2.3 (1D and 2D arrays, reading and writing with iteration)

How to use. Questions 1 and 2: one mark per bullet point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Question 3 is marked as a whole using the two level grids below (best fit), then the AO2 and AO3 marks are added. Ignore minor syntax slips throughout. Example answers are one possible response, not the only one.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 8.2.1, 8.2.2	The number of bikes hired from the third station // from the station in `StationName[3]` … on day 5	**1**	Both the station and the day are needed. Do not accept station 5 on day 3. Do not accept "the total for station 3" or "the name of station 3".
1(b) 0478 7.4	R2: **Totalling** R3: **Counting** R4: **Finding the maximum**	**3**	One mark per correct row. A row with more than one tick scores 0.
2 0478 7.8, 7.5(a), 8.1.3, 8.1.8	One mark for each problem identified **with** a matching correction. Max 2. Any two from: • No prompt / message before the input (line 03 // line 07), so the user is not told what to enter … **correction:** add an `OUTPUT` before each `INPUT` asking for the number of hires, e.g. `OUTPUT "Enter hires for ", StationName[Station], " on day ", Day` • The validation uses an IF, so the value is only checked once // a second invalid value (e.g. 301 after −5) is accepted and stored … **correction:** use a condition-controlled loop (REPEAT … UNTIL // WHILE … DO … ENDWHILE) so the value is input again until it is from 0 to 300 • The error message on line 06 is not suitable // does not say what is wrong or what the valid range is … **correction:** output a meaningful message, e.g. `"Invalid: enter a number from 0 to 300"` • There are no comments, so the code is not explained … **correction:** add comments (`//`) to explain the loops and the validation	**2**	Max 2. The problem and its correction must match; a problem with no correction scores 0. Credit any other valid problem and correction. Do not credit: saying the condition on line 04 is wrong (it is correct); saying the loop bounds are wrong; saying the arrays must be declared (the question says they need not be); "it should use a different array".

Question 3 (15 marks): requirements, data structures and level grids

QUESTION	ANSWER	MARKS	GUIDANCE
3 0478 Paper 2 scenario question; 7.4, 7.5(a), 7.9, 8.1.3, 8.1.4, 8.1.8, 8.2.1–8.2.3	**Requirements (techniques)** **R1** Input the number of hires for each station for each day, with prompts, and store it in `Hires[]` (nested iteration, input with prompts, storing in a 2D array); validate each value is from 0 to 300 inclusive and require re-entry until valid (range check inside a condition-controlled loop) **R2** Total the hires for each station and store each total in `StationTotal[]` (nested iteration, totalling, storing in a 1D array); output each station's name with its total (output with messages) **R3** Count the stations whose total is less than 150 (iteration, selection, counting); output the count with a message **R4** Find the highest weekly total (iteration, selection, finding the maximum); output the highest total and the name of every station with that total (iteration and selection over the totals, output with messages) **Data structures required** (names must be used as given) 1D array `StationName[1:12]` (station names, already stored) 2D array `Hires[1:12, 1:7]` (hires for each station on each day) 1D array `StationTotal[1:12]` (weekly total for each station) Variables chosen by the candidate, e.g. loop counters, the value input, the count, the highest total	**15**	Pseudocode, Python, Visual Basic or Java are all accepted. Any other programming language scores 0. Syntax is not assessed: mark the logic. Built-in functions of the chosen language (e.g. `max()`, `sum()` in Python) may be used to meet a requirement. Arrays indexed from 0 are accepted when used consistently. Ignore any declarations. The given array names must be used; do not penalise a different case (e.g. `stationtotal`). Common ways to lose marks: validating with a single IF (checks once only); outputting only one station when several share the highest total; resetting a total or count inside the wrong loop; inputs with no prompts; no comments; using names other than the given arrays.

AO2: apply knowledge and understanding (max 9). Mark by best fit.

Level	Description
7–9	All the requirements R1 to R4 are covered by appropriate techniques; the data structures store all the data that is needed.
4–6	More than one appropriate technique is applied to the scenario; some appropriate data structures are used to store some of the data.
1–3	At least one appropriate technique is used; some data is stored, but not always in an appropriate way.
0	No creditable response.

AO3: provide solutions to problems (max 6). Mark by best fit.

Level	Description
5–6	The program is commented throughout and uses meaningful identifiers. It is in a logical order, is accurate and meets all the requirements R1 to R4. Minor syntax errors are ignored.
3–4	The program has some relevant comments and most identifiers are sensible. It contains some errors but performs most of the requirements.
1–2	The program has no relevant comments, but some identifiers are sensible. It is illogical or inaccurate in many places, but attempts at least one requirement.
0	No creditable response.

Total for Question 3 = AO2 mark (max 9) + AO3 mark (max 6) = **15**.

Question 3: example 15-mark answer (Cambridge pseudocode)

One possible answer, sectioned by requirement. Other logically correct answers, and answers in Python, Visual Basic or Java, gain full marks.

```
// R1: input and validate the hires for every station for every day
FOR Station ← 1 TO 12
    FOR Day ← 1 TO 7
        // prompt names the station and the day so the user knows what to enter
        OUTPUT "Enter the number of bikes hired from ", StationName[Station],
          " on day ", Day, " (0 to 300)"
        INPUT NumberHired
        // range check: keep asking until the number is from 0 to 300
        WHILE NumberHired < 0 OR NumberHired > 300 DO
            OUTPUT "Invalid: the number must be from 0 to 300. Enter it again"
            INPUT NumberHired
        ENDWHILE
        // store the valid number in the 2D array
        Hires[Station, Day] ← NumberHired
    NEXT Day
NEXT Station
```

```
// R2: total the hires for each station and output each total
FOR Station ← 1 TO 12
    // start each station's total at zero
    StationTotal[Station] ← 0
    FOR Day ← 1 TO 7
        StationTotal[Station] ← StationTotal[Station] + Hires[Station, Day]
    NEXT Day
    OUTPUT StationName[Station], " total hires this week: ", StationTotal[Station]
NEXT Station
```

```
// R3: count the stations with fewer than 150 hires in the week
QuietCount ← 0
FOR Station ← 1 TO 12
    IF StationTotal[Station] < 150
      THEN
        QuietCount ← QuietCount + 1
    ENDIF
NEXT Station
OUTPUT "Stations with fewer than 150 hires this week: ", QuietCount
```

```
// R4: find the highest weekly total
HighestTotal ← StationTotal[1]
FOR Station ← 2 TO 12
    IF StationTotal[Station] > HighestTotal
      THEN
        HighestTotal ← StationTotal[Station]
    ENDIF
NEXT Station
OUTPUT "Highest weekly total: ", HighestTotal
// output every station with the highest total, in case more than one has it
OUTPUT "Station(s) with this total:"
FOR Station ← 1 TO 12
    IF StationTotal[Station] = HighestTotal
      THEN
        OUTPUT StationName[Station]
    ENDIF
NEXT Station
```

Total: 21 marks