## 7 Algorithm design and problem-solving continued

| Candidates should be able to: | Notes and guidance |
|---|---|
| 4 Understand standard methods of solution | • Limited to:<br>– linear search<br>– bubble sort<br>– totalling<br>– counting<br>– finding maximum, minimum and average values |
| 5 (a) Understand the need for validation checks to be made on input data and the different types of validation check | • Including:<br>– range check<br>– length check<br>– type check<br>– presence check<br>– format check<br>– check digit<br>– the purpose of each validation check and writing algorithms to implement each validation check |
| (b) Understand the need for verification checks to be made on input data and the different types of verification check | • Including:<br>– visual check<br>– double entry check |
| 6 Suggest and apply suitable test data | • Limited to:<br>– normal<br>– abnormal<br>– extreme<br>– boundary<br>• Extreme data is the largest/smallest acceptable value<br>• Boundary data is the largest/smallest acceptable value and the corresponding smallest/largest rejected value |
| 7 Complete a trace table to document a dry-run of an algorithm | • Including, at each step in an algorithm:<br>– variables<br>– outputs<br>– user prompts |
| 8 Identify errors in given algorithms and suggest ways of correcting these errors | |
| 9 Write and amend algorithms for given problems or scenarios, using: pseudocode, program code and flowcharts | • Precision is required when writing algorithms, e.g. `x > y` is acceptable but `x is greater than y` is not acceptable<br>• See section 4 for flowchart symbols<br>• See section 4 for pseudocode |