| Task | Expected response | Additional guidance | Max marks | |
|---|---|---|---|---|
| 2c | Call validate password function **from within** first procedure | | 1 | Implementation (15) |
| | Extract/identify first and last character | Accept use of pre-defined function or array position | 1 | |
| | Use of pre-defined function to convert first and last characters to ASCII | | 1 | |
| | Validate first and last characters | | 1 | |
| | Return password | | 1 | |
| | Parameters passed to second procedure | | 1 | |
| | ♦ Read data from file<br>♦ Assign to parallel arrays | | 2 | |
| | Add/append new member data to existing arrays | | 1 | |
| | Display all members, count of category and total members | ‘Oliver Wilson Adult’ showing as new member<br>Award 0 marks if password is displayed | 1 | |
| | Parameter category() passed to third procedure | | 1 | |
| | Counting occurrence algorithm:<br>♦ Intialise three count variables<br>♦ Increment appropriate count | | 2 | |
| | Program code is maintainable | Meaningful variable names and regular internal commentary should be relevant to task. | 1 | |
| | Matches steps 1, 2 and 3 of main algorithm | | 1 | |

| Task | Expected response | Additional guidance | Max marks | |
|---|---|---|---|---|
| 2d | Award 1 mark each for any two bullets:<br>Design tests to ensure that:<br>♦ the correct substrings are extracted/first and last characters correctly identified<br>♦ characters are correctly converted to ASCII code<br>♦ the first and last characters are each within the valid ASCII value range<br>♦ both first and last characters have to be valid to make the whole password valid<br>♦ the valid password is returned to main procedure | Answers should reference/give examples from candidate's code. | 2 | Testing (2) |
| 2e | Award 1 mark for an example from each category. This must be evidenced in candidate's code.<br>♦ Efficiency eg:<br>- use of appropriate constructs (loops, else if statements)<br>- use of appropriate data structures (parallel arrays)<br>- parameter passing (reusable modules, no global variables)<br>♦ Robustness eg:<br>- input validation<br>♦ Fit for purpose<br>This must refer back to functional requirements.<br>Program must run as expected | Accept not efficient/robust/fit for purpose responses with valid reasons/examples from code | 3 | Evaluatoin (3) |