Specific marking instructions

Task 1 — software design and development

Task	Expected response	Max mark	Additional guidance
1a	♦ apropriate drink selected (milkshake, smoothie or fruit juice) ♦ validate fruit name or a correct description of the validation ♦ check if you want to enter another fruit	**2**	Award 1 mark for each bullet. Maximum 2 marks. Take care that candidate doesn't repeat/reword processes already given: ♦ check no more than 6 fruits entered ♦ select mystery fruit option ♦ calculate total number of fruits
1b	♦ conditional loop ♦ input of fruit inside loop ♦ ensure input is at least 4 characters	**3**	
1c	♦ the fruits you entered were(.) lemon pear peach orange the mystery fruit is(.) mango ♦ (fruit) juice	**2**	List of fruits may be written vertically or horizontally.

Task	Expected response	Max mark	Additional guidance
1d	**Initialise variables (2 marks)** ♦ array storing ten mystery fruits ♦ empty array to store users fruits ♦ counter = 0 ♦ decision **Input validation (3 marks)** ♦ conditional loop with correct condition ♦ input of all fruits within validation loop ♦ error message displayed inside loop **Fruit loop (4 marks)** ♦ add fruit entered to array inside the loop ♦ increment counter inside the loop ♦ ask user for decision regarding another fruit entry inside the loop ♦ end conditional loop when user enters no or counter = 6 **Mystery fruit (1 mark)** ♦ generate random number between 0 and 9 **Generating outputs (5 marks)** One mark each for the following outside the fruit loop: ♦ display array of fruits entered ♦ display a random mystery fruit ♦ add 1 to counter ♦ if structure with correct conditions ♦ correct drink messages associated with conditions	**15**	Ensure condition matches the type of loop used. Accept either nested if or if-else if-else structures.
1e	**Efficiency:** • comment on efficiency or inefficiency of own code **Robustness:** • comment on one aspect of robustness of own code **Readability:** ♦ readability — comment on one aspect of readability in the candidate's own code	**3**	In efficiency and robustness, candidates should not give a generic answer that could apply to any program. Answers must contain examples from the candidate's code. Evaluation of readability must contain an element of evaluation rather than simple statements of terms. For example "I have used white space to highlight structures in my program" not "I have used white space". The candidate's code must also show evidence of this for a mark to be awarded.