Midterm Marking Scheme - Part 2 Question 1 - Out of 12 marks

Function header -> 3 marks:
- "def" -> 0.5 marks
- descriptive function name (like "encrypt") -> 0.5 marks
- parameter (a string) -> 0.5 marks
- descriptive parameter name (like "plainMessage") -> 0.5 marks
- docstring -> 0.5 marks
- docstring well describes what the function does and returns -> 0.5 marks

Function body -> 9 marks:
- Mapping vowels to numbers (encrypted vowels): 2 marks
- There is an accumulator variable -> 0.5 marks
- Accumulator variable initialize to "" (empty string) -> 0.5 marks
- Each char of plain message is "processed" (perhaps in a "for" loop) -> 1 mark
- Lower/upper case of char is dealt with -> 1 mark
- Each character, if a vowel, is mapped to its corresponding encrypted characters (a number) -> 1 mark
- And "put" into the accumulated cipher -> 1 mark
- Any non-vowel character finds itself in the accumulated cipher -> 1 mark
- "return" statement -> 0.5
- returning the accumulator variable (i.e., encrypted message aka cipher) -> 0.5

Midterm Marking Scheme - Part 2 Question 2 - Out of 20 marks

Header Comment Block -> 2 marks:
- Filename -> 0.5 marks
- Description of program -> 0.5 marks
- Author -> 0.5 marks
- Date -> 0.5 marks

Sensible comments throughout the program -> 1 mark

Body of program -> 17 marks:
- Display vending machine welcome -> 0.5 marks
- If above is done matching the wording and layout of the sample runs -> +0.5 marks
- Display vending machine menu (item and cost) -> 1 mark
- If above is done matching the wording and layout (order) of the sample runs -> +1 mark
- If above is done without repeating the code -> +2 marks

Display menu prompt -> 0.5 marks
- If above is done matching the wording and layout of the sample runs -> +0.5 marks

Display each item + (y/n) -> 1 mark
If above is done matching the wording and layout of the sample runs -> +1 mark
- If above is done without repeating the code -> +1 mark
- Read user input for each item -> 1 mark
- And take care of lower/upper case of user input -> 1 mark
- Take care of user input that is neither "y/Y" nor "n/N" -> 1 mark
- Initialize accumulator variable (running cost or sum or total) to 0 -> 1 mark
- If user input "y/Y", accumulate the sum of selected item -> 2 marks
- Print the cost of selected items -> 1 mark
- If above is done matching the wording and layout of the sample runs -> +1 mark
- If tax correctly computed and added to total -> BONUS 1 mark - should we give 0.5 marks if tax wrongly computed?