

Programming Assessment

by Iwan Thomas

Submission date: 24-May-2026 11:20AM (UTC+0100)

Submission ID: 284502217

File name: 66124_Iwan_Thomas_Programming_Assessment_727104_425386844.docx (105.52K)

Word count: 261

Character count: 1373

1a.

Test Case	Observed output	Error Type	Explanation
A	Error	Syntax	Incorrect casing on variable
B			
C			
D			

```
if zone_counter <= numberofZones:
    ^^^^^^^^^^^^^^^^^
NameError: name 'numberofZones' is not defined. Did you mean: 'numberOfZones'?
```

Error occurs in all cases

1b.

Test Case	Observed output	Error Type	Explanation
A	Error	Syntax	Underscores present where they shouldn't be
B			
C			
D			

```
total_spaces=Spaces_Per_Zone
    ^^^^^^^^^^^^^^^^^
NameError: name 'Spaces_Per_Zone' is not defined. Did you mean: 'SpacesPerZone'?
```

3a.

Test Case	Observed output	Error Type	Explanation
A	Error	Logic	free_spaces undefined
B			
C			
D			

```
if free_spaces > max_free_spaces:
    ^^^^^^^^^^^^^
NameError: name 'free_spaces' is not defined. Did you mean: 'max_free_spaces'?
```

This error was corrected by defining the 'free_spaces' variable. The variable was defined as free_spaces=0.

Test Case	Observed output	Issue Resolved?	Explanation
A	Occupancy 4 100% warning	yes	Algorithm reached end state, no errors. All variables correctly defined.
B	Occupancy 3 100% warning		Insufficient capacity
C	error	User is instructed to only use 1 or 0	Cannot accept string (%)
D	Occupancy 3 100% warning		Reached capacity

```

1
2   numberOfZones = int(input("Enter number of zones:"))
3   SpacesPerZone = int(input("SpacesPerZone:"))
4   zone_counter=1
5   max_free_spaces=1
6   zone_with_max_free=-1
7   free_spaces=0
8   while zone_counter <= numberOfZones:
9       if zone_counter <= numberOfZones:
10          space_counter=1
11          occupied_count=0
12          total_spaces=SpacesPerZone
13          if space_counter <= SpacesPerZone:
14              while space_counter <= SpacesPerZone:
15                  OccupancyStatus= int(input("Enter occupancy status, 1 or 0:"))
16                  if OccupancyStatus == 1 or OccupancyStatus == 0:
17                      if OccupancyStatus == 1:
18                          occupied_count=occupied_count+1
19                          space_counter=space_counter+1
20                      else:
21                          print("Invalid Status")
22          else: free_spaces=total_spaces-occupied_count
23          occupancy_percent=(occupied_count/total_spaces)*100
24          print(f"Occupied count:{occupied_count}, Occupancy percentage:{occupancy_percent}%")
25          if occupancy_percent >80:
26              print("WARNING, Occupancy over 80%")
27          if free_spaces > max_free_spaces:
28              max_free_spaces=free_spaces
29              zone_with_max_free=zone_counter
30              zone_counter=zone_counter+1
31      else:
32          print(f"Zone with most free spaces is {max_free_spaces}, "
33              f"and the number of free spaces is {free_spaces}")

```

Script with all syntax, logic and spelling errors corrected.

Improving Logic

The example use of nested if statements makes the script unwieldy, and difficult to edit due to an obscure decision tree structure. Additionally, this example requires that its loop doesn't terminate, else it would stop the program prematurely.

Reflection

This script makes use of nested if and else statements, comparing multiple variables and indefinite loops, with the most challenging being the aligning of parent if statements with corresponding child statements.

The first test case in a table was always useful in identifying problems in the script, and was used to root out syntax and logic errors.

Running real test cases helped to confirm and identify suspected errors in the script, as multiple rounds of testing and correction were required to generate the desired output.

Programming Assessment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

70/100

PAGE 1

PAGE 2

PAGE 3

GRADING FORM: ACCA4020 RUBRIC 2026

IWAN THOMAS

70

TASK 1A [20]

Correct implementation of flowchart (even with errors) [8] Code runs (even if incorrect output) [5] All test cases attempted [4] Screenshots/evidence provided [3]



The flowchart is mostly implemented as given, and test cases are executed. Evidence of some errors (e.g., iSpelling errors, invalid input, or incorrect

17

outputs) is shown, although screenshots or outputs could be clearer.

TASK 1B [15]

Correct outputs recorded [5] Identification of error types [5] Clear explanation of errors [5]



Some errors identified, but important issues, such as equal-zone logic is missing

11

TASK 2A [20]

Errors correctly fixed [10] Improved logic (loops, conditions) [6] Input validation added [4]



Most errors are fixed, including validation and general logic improvements. However, handling of the equal zone scenario is not well explained

15

TASK 2B [20]

All test cases re-run [6] Outputs clearly recorded [6] Comparison with old results [4] Explanation of fixes [4]



Most test cases are re-tested with generally correct outputs. Improvements are evident, though comparisons or explanations could be more detailed

15

TASK 3 [10]

Understanding of difficulty [5] Impact on design/efficiency [5]



Basic discussion of challenges. Limited explanation of the impact on efficiency or design, with little detail on alternative approaches

5

TASK 4 [15]

Identifies challenges [5] Examples from assignment [5] Reflection on testing/learning [5]



Reflection lacks depth and does not clearly link to the assignment tasks. Limited discussion of errors or learning

7