CS1010S Programming Methodology
National University of Singapore

Makeup Practical Examination 1

19 Sept 2025

Time allowed: 1 hour

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **two** questions. The time allowed for solving this test is **1 hour** .
3. The maximum score of this test is **10 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the template `practical-template.py` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness using Glide and not depend only on the provided test cases. Do ensure that you submit your answers correctly by running the test cases at least once. Note that a **heavy penalty will be applied for code that cannot be interpreted due to SyntaxError**.
6. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.
7. While you may use any built-in function provided by Python, you may not import functions from any packages, unless otherwise stated and allowed by the question.
8. Adhere strictly to the given restrictions; failure to do so will result in a **score of 0 marks**.

GOOD LUCK!

Question 1 : Count Valid Brackets [5 marks]

INSTRUCTIONS: Please read the entire question carefully before attempting to solve the problem. Adhere strictly to the given restrictions; failure to do so will result in a score of 0 marks.

Restrictions.

- Use of any compound data structures, such as tuple, list, dict, set, etc is strictly prohibited.
- You must provide a **purely iterative solution** for this question. Neither the `count_matching_bracket` function nor any helper function you may write should be recursive in nature.

Implement a **iterative** function `count_matching_bracket` that takes a non-empty string input `s` consisting only of opening and closing round brackets (i.e., `'('` and `')'`). The function should return the total number of valid bracket pairs (i.e., `'()'` or `'( .. )'` where there may be other valid/invalid bracket in between) in the string.

For example:
If s = `'(())'`, the output would be 2 since there are two valid `'()'` in s.
If s = `'(()())'`, the output would be 3 since there are three valid `'()'` in s.
If s = `'(()()'`, the output would be 2 since there are only two valid `'()'` in s.

Sample execution:

```
>>> count_matching_bracket('(())')
2

>>> count_matching_bracket('()()')
2

>>> count_matching_bracket('(()()()())')
5

>>> count_matching_bracket('(()()()(()))')
6

>>> count_matching_bracket('(()()()(()')
4

>>> count_matching_bracket('((((')
0

>>> count_matching_bracket('))')
0
```

Question 2 : Count Adjacent Pairs [5 marks]

INSTRUCTIONS: Please read the entire question carefully before attempting to solve the problem. Adhere strictly to the given restrictions; failure to do so will result in a score of 0 marks.

Restrictions.

- You are **only allowed to use integers** to solve this question. You are not allowed to use `str(...)` function to cast integer into a string. Use of any compound data structures, such as tuple, list, dict, set, etc is strictly prohibited.
- You must provide a **purely recursive solution** for this question. Neither the `count_adjacent_pairs` function nor any helper function you may write should contain iterative statements such as `for` and `while`.

Write a **recursive** function `count_adjacent_pairs(n)` that takes a positive integer, `n` as input returns the number of times two equal digits appear next to each other.

For example: If `n` = `1221`, the output is `1` since there is only one pair of adjacent equal digits, `22`.
If `n` = `12233322111`, the output is `6` since there is two pairs of `11`, two pairs of `22`, and two pairs of `33`.

Please refer to the sample execution for more examples.

```
>>> count_adjacent_pairs(1122)
2

>>> count_adjacent_pairs(1221)
1

>>> count_adjacent_pairs(12223123)
2

>>> count_adjacent_pairs(12233322111)
6

>>> count_adjacent_pairs(111111111)
8

>>> count_adjacent_pairs(1)
0
```

— E N D   O F   P A P E R —