Student Number _____

The University of Melbourne
School of Computing and Information Systems

Final Examination, Semester 2, 2019
COMP10001 Foundations of Computing

Reading Time: 15 minutes. **Writing Time:** 2 hours.

This paper has 19 pages including this cover page.

Instructions to Invigilators:

Students must write all of their answers on this examination paper. Students may not remove any part of the examination paper from the examination room.

Instructions to Students:

There are 10 questions in the exam worth a total of 120 marks, making up 50% of the total assessment for the subject.

- All questions should be answered by writing a brief response or explanation in the lined spaces provided on the examination paper.
- It is not a requirement that all the lined spaces be completely filled; answers should be kept concise. Excessively long answers or irrelevant information may be penalised.
- Only material written in the lined spaces provided will be marked.
- The reverse side of any page may be used for notes or draft answers.
- Your writing should be clear; illegible answers will not be marked.
- Extra space is provided at the end of the paper for overflow answers.
  Please indicate in the question you are answering if you use the extra space.
- Your answers should be based on Python 3 (the version that Grok uses), and can use any of the standard Python libraries.

Authorised Materials: No materials are authorised.

Calculators: Calculators are not permitted.

Library: This paper may be held by the Baillieu Library.

Examiners' use only										
1	2	3	4	5	6	7	8	9	10	Total

Blank

Part 1: Algorithmic Thinking

Question 1

[12 marks]

Evaluate the following expressions, and provide the output in each case.

(a) `'snout'[2:]`

(b) `sorted(['fog', 'cog', 'dog'])`

(c) `('now'[-1] + 'south'[-1] + 'hurry'[-1])`

(d) `sorted({'owls': 'good', 'more owls': 'not so good'}.values())[1]`

(e) `'nic' not in sorted('cinema')`

(f) `[i%2 for i in range(0, 10, 2)]`

Blank

Question 2

[12 marks]

Rewrite the following function, replacing the `for` loop with a `while` loop, but preserving the remainder of the original code structure:

```
def all_vowels(word):
    vowels = 'aeiou'
    seen = ''
    for c in word:
        if c in vowels:
            seen += c
    return sorted(seen) == list(vowels)
```

Blank

Question 3

[12 marks]

In project 1, you considered preferential voting, in which a valid vote required a voter to list all candidates once in their preferred order.

The following function `orderings(candidates)` is intended to recursively generate all possible valid votes for a given set of n candidates.

```
1  def orderings(candidates):
2
3      if not candidates:
4          return [candidates]
5
6      vote == []
7      for i in range(len(candidates)):
8          current = candidates(i)
9          remainining = candidates[i] + candidates[i+1:]
10         for o in orderings(remaining)
11             vote.append([candidates] + o)
12
13     return vote
```

However, there are several errors in the given function definition. Identify exactly **three (3)** errors and specify: (a) the line number where the error occurs; (b) the type of error, as *syntax, logic* or *runtime*; and (c) how you would fix each error, in the form of corrected code.

Blank

Question 4

[12 marks]

To "obfuscate" text means to make it hard to read. Obfuscating text by substituting or modifying certain characters, can make it more difficult for a casual reader to understand, while still leaving it readable by someone familiar with the rules of modification.

The following function is intended to obfuscate a (lowercase) string of text according to the following rules:

1. First, all consecutive duplicate letters are replaced with a single letter; for example, 'hello' becomes 'helo' and 'mississippi' becomes 'misisipi'.

2. Next, some characters are replaced with numbers/symbols, according to the substitution dictionary below; e.g., 'a' becomes '@' and 'e' becomes '3'.

3. Finally, given the updated string after applying rules 1 and 2, convert each character at an even-numbered index (i.e., index positions 0, 2, 4, etc) in this string to uppercase; e.g., the string 'doubt' becomes 'DoUbT'.

For example:

```
>>> obfuscate_text('keeping secrets is wise')
'K3P!Ng $3cR3T$ !$ W!$3'
```

Assume you are given the dictionary `subs` as follows:

```
subs = {
        'a': '@',
        's': '$',
        'i': '!',
        'e': '3',
        'l': '1'
        }
```

As presented, the lines of the function are out of order. Put the line numbers in the correct order and introduce appropriate indentation (indent the line numbers using the columns in the answer table provided to show how the corresponding lines would be indented in your code).

```
1  obs_text = ''
2  i = 0
3  if short_text[i] in subs:
4  while i < len(short_text):
5  else:
6  obs_text += short_text[i]
7  obs_text += short_text[i].upper()
8  obs_text += subs[short_text[i]]
9  elif i%2 == 0:
10 i += 1
11 return obs_text
12 def obfuscate_text(text):
13 prev_char = ''
14 if c == prev_char:
15 short_text = ''
16 continue
17 short_text += c
18 for c in text:
19 prev_char = c
```

Blank

Answer:

Blank

Part 2: Constructing Programs

Question 5

[10 marks]

The following function is intended to read in a csv file containing a list of different widgets, and the number of units of each that were sold in a given year, and write out a new csv file which adds an additional column containing the proportion of total sales represented by each widget.

Proportions should be displayed as percentages, shown to two decimal places.

For example, if the file `data.csv` contains the following:

```
labels,counts
a,4
b,3
c,5
d,1
```

After running `add_proportions('data.csv', 'new_data.csv')`, the file `new_data.csv` will contain:

```
labels,counts,proportions
a,4,30.77
b,3,23.08
c,5,38.46
d,1,7.69
```

Provide code to insert into each of the numbered boxes in the code below to complete the function as described. Note that your code will be evaluated at the indentation level indicated for each box.

```
import [ 1 ]

def add_proportions(csv_filename, new_filename):

    with open(csv_filename) as csv_file:
        reader = csv.DictReader(csv_file, skipinitialspace=True)
        header = reader.fieldnames
        data = list(reader)

    count_sum = 0
    for row in [ 2 ]
        count_sum += int(row['count'])

    for row in data:
        prop = int(row['count']) / count_sum * 100
        row['proportion'] = round(prop, 2)

    new_header = header + [ 3 ]

    with open( [ 4 ] ) as new_file:
        writer = csv.DictWriter(new_file, new_header)
        writer.writeheader()
        for row in [ 5 ]
            writer.writerow(row)
```

Blank

(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

Blank

Question 6

[10 marks]

The aim of this question is to write *a single Python statement* that generates a given error or exception. Write a single Python statement that generates each of the following errors and exceptions, assuming it is executed in isolation of any other code.

(a) `IndexError: list index out of range`

(b) `TypeError: 'tuple' object does not support item assignment`

(c) `AttributeError: 'str' object has no attribute 'len'`

(d) `KeyError: 0`

(e) `TypeError: unsupported operand type(s) for +: 'int' and 'str'`

Blank

Question 7

[18 marks]

Write a function `alternate(word)` that rearranges the characters of the given lowercase alphabetic string in `word` such that vowels and consonants appear in alternating positions. The ordering of vowels and consonants should otherwise remain unchanged.

When no alternating ordering is possible (e.g., because there are too many vowels, or too many consonants) your function should return `None`.

When there are equal numbers of vowels and consonants, the rearranged word should begin with whichever letter comes earlier in the alphabet.

For example:

```
>>> alternate('tools')
tolos
>>> alternate('ambulance')
mabulanec
>>> alternate('headache')
ehadaceh
>>> alternate('football')
None
```

Blank

Blank

Blank

Part 3: Conceptual Questions

Question 8: Algorithmic Problem Solving [12 marks]

(a) In the context of the analysis of algorithms, briefly explain what is meant by an *exact algorithmic approach*?

[6 marks]

(b) Is *binary search* an example of a "divide-and-conquer" algorithm? Briefly explain why or why not.

[6 marks]

Blank

Question 9: Applications of Computing [12 marks]

(a) Briefly describe **two** applications of *artificial intelligence*.

[6 marks]

(b) Briefly explain the concept of *anomaly detection* and *how it can be used in network security*.

[6 marks]

Blank

Question 10: HTML and the Internet [10 marks]

Complete this HTML page:

```
<![      1      ] html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<[      2      ] http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>COMP10001: The Final Exam</title>
</head>
<(*\blank[3]@*)>
<h1>COMP10001: The Final Exam</h1>
<p>Starring:
<[      3      ]>
  <li>You!</li>
  <li>Chris, Nic, Marion</li>
  <li><(a [      4      ]="./images/pikachu.gif" alt="Pikachu!"/></li>
</ul>
</p>
</body>
</[      5      ]>
```

by filling in the numbered blanks based on selecting from among the following candidate strings:

- a
- body
- DOCTYPE
- ul
- html
- img
- meta
- ol
- src
- td

Blank

(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

Blank

This is blank space for further answers should you need it. Please ensure that you label the answers in this area carefully, and that you indicate on the corresponding question page that your answer can be found here.

Blank

Blank

— End of Exam —

Blank