Student Number

The University of Melbourne
Department of Computing and Information Systems

Mid-semester Test, Semester 1, 2019
COMP10001 Foundations of Computing

Writing Time: 45 minutes

This paper has 7 pages including this cover page.

There are 5 questions in the paper, for a total of 45 marks (one mark for each minute of test time).

- 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.
- Only material written in the lined spaces provided will be marked.
- 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.
- You should base all of your answers on Python3 (the version of Python that is used in Grok).
- Your answers can use any of the standard Python libraries, but be sure to call/import them correctly.

Authorised Materials: No materials are authorised.

Calculators: Calculators are not permitted.

Examiners' use only

1	2	3	4	5	Total	

Question 1

[9 marks]

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

(a) `["apples", "bananas", "cantaloupes"][-1][:3]`

____

(b) `bool([1, 3, 5] and "bods")`

____

(c) `f"{4+7} and {11*2} were racehorses"`

____

Question 2

[9 marks]

What are the final values of each of the variables below, after executing the following code:

```
def fun(i, j):
    return str(j) in str(i)

nums = (1, 1)
r = 3
total = 0
for i in range(3):
    total += fun(nums, r)
    p, q = nums
    nums = (q, p + q)
```

(a) `i`

____

(b) `nums`

____

(c) `total`

____

Question 3

[7 marks]

Rewrite the following function, replacing the `for` loop with a `while` loop:

```
def last_letter(word):
    last = 0
    for i in range(1, len(word)):
        if word[i] > word[last]:
            last = i
    return last
```

Question 4 [10 marks]

The following `create_deck` function initialises a deck of 52 playing cards in the form of a list of lists, where each entry is a string representing a card (e.g. `'AS'` is the ace of spades, and `'0D'` is the 10 of diamonds). A second function `replace_with_joker` replaces a single card in the original deck with a joker, at the indicated (zero-offset) position in the deck. When called as shown in the final three lines of code below, the printed output should be equivalent to the following:

```
[['AS', 'AH', 'AD', 'AC'], ['2S', '2H', '2D', '2C'],
 ['3S', '3H', '3D', 'joker'], ['4S', '4H', '4D', '4C'],
 ['5S', '5H', '5D', '5C'], ['6S', '6H', '6D', '6C'],
 ['7S', '7H', '7D', '7C'], ['8S', '8H', '8D', '8C'],
 ['9S', '9H', '9D', '9C'], ['0S', '0H', '0D', '0C'],
 ['JS', 'JH', 'JD', 'JC'], ['QS', 'QH', 'QD', 'QC'],
 ['KS', 'KH', 'KD', 'KC']]
```

Provide a single statement to insert into each of the numbered boxes to complete the code to achieve the intended behaviour. Note that your code should run at the indentation level indicated for each box.

```
VALUES = "A234567890JQK"
SUITS = "SHDC"

def create_deck():
    [      1      ]
    for i in VALUES:
        [      2      ]
        for j in SUITS:
            [      3      ]
        deck.append(row)
    return deck

def replace_with_joker(deck, pos):
    row_id = pos//4
    [      4      ]
    deck[row_id][val_id] = "joker"

deck = create_deck()
replace_with_joker(deck, 11)
print(deck)
```

(1) ______________________________

(2) ______________________________

(3) ______________________________

(4) ______________________________

Question 5 [10 marks]

Write the function `sum_eq_prod`, which takes a non-negative integer argument `num`, and returns `True` if the sum of the digits of `num` is the same as the product of the digits, and `False` otherwise. For example, the function call `sum_eq_prod(123)` should return `True` as $1+2+3 = 1 \times 2 \times 3 = 6$, whereas `sum_eq_prod(42)` should return `False` as $4+2 \neq 4 \times 2$.

Example calls to `sum_eq_prod` are:

```
>>> sum_eq_prod(123)
True
>>> sum_eq_prod(42)
False
>>> sum_eq_prod(2)
True
>>> sum_eq_prod(52111)
True
```

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.

⇜⋐⋐⋐⋐⋐⋐⋐ End of Test ⋑⋑⋑⋑⋑⋑⋑⇝