MSc in Artificial Intelligence
MRes in AI & Machine Learning

COMP70053
Python Programming

Exercises

Question 1

In `q1/q1.py`, complete the function `longest_word(word_list)` that returns the longest word in `word_list` and also the length of the longest word.

`word_list` is a `list` of `str`

The function should return a `tuple`, where the first item in the tuple is the longest word, and the second item is the length of this word.

If there is more than one word with the highest length, return the one that occurs first in the list.

Sample input and output

```
["I", "love", "Python", "programming", "the", "most"]
```

returns

```
(programming, 11)
```

Question 2

In `q2/q2.py`, complete the function `filter(string)` that takes a `string`, removes all words containing an `"e"` from the `string`, and returns the new string.

Sample input and output

```
"I swear by the moon and the stars in the sky"
```

returns

```
"I by moon and stars in sky"
```

Question 3

In q3/q3.py, complete the function `convert(seconds)` that converts `seconds` (an `int`) an hour/minute/second format.

The function should return a `tuple` `(hours, minutes, seconds)`.

Sample input and output

`38` returns `(0, 0, 38)`
`176` returns `(0, 2, 56)`
`3823` returns `(1, 3, 43)`

Question 4

In q4/q4.py, complete the function `simplify_fraction(numerator, denominator)` that simplifies a fraction `numerator/denominator` to its most simplified form.

The function should return a `tuple` `(numerator, denominator)`.

Sample input and output

`simplify_fraction(4, 8)` returns `(1, 2)`
`simplify_fraction(7, 9)` returns `(7, 9)`
`simplify_fraction(15, 5)` returns `(3, 1)`
`simplify_fraction(200, 300)` returns `(2, 3)`

Question 5

In `q5/`, you are provided with some files. You are given a `BankAccount` class with the read-only attribute `balance` and the methods `deposit(amount)` and `withdraw(amount)`.

`CashAccount` is a special type (inheritance) of `BankAccount` that allows users to deposit and withdraw money from a cash machine, but with a service charge for each cash withdrawal. The first four cash withdrawals of the month are free, but all subsequent withdrawals will each incur a £1.00 service fee. The fees will only be deducted by the bank at the end of each month, when the bank will compute and deduct the correct amount in one go using the method `deduct_fees()`. There is no charge for making a deposit.

Your task is to complete the `CashAccount` class in `q5/q5.py`. More specifically:

- Implement a `deduct_monthly_fees()` method that deducts the fees and reset the withdrawal transaction count for the month. Remember that the first four cash withdrawals are free.
- You may also need to override the `__init__()` constructor and the `withdraw()` method inherited from the `BankAccount` superclass, to be able to keep track of the withdrawal transaction count.

You may only modify `q5.py` and `test_q5.py`. You should **NOT** modify `BankAccount` in `bank_account.py`.

Run `test_q5.py` to test your class.

Sample input and output

The expected output after running `test_q5.py`:

```
Creating a new cash account in October with an initial balance of £1000.0.
User makes 10 deposits of £10 each in October...
Balance after the 10 deposits:  £1100.0
User makes 10 withdrawals of £100 each in October...
Balance after the 10 withdrawals:  £100.0
The bank deducts its service fee at the end of October
Balance after deduction:  £94.0
Starting balance in November:  £94.0
```

Question 6

Note: This is a question (out of two questions) from the 2019/20 exam. The style of this exam is different from your exam, but you might find this question a useful exercise.

You will implement several utility functions for a checkout system at a fruit market. The fruit market is represented as a nested dictionary, where each fruit has a price, and some fruit may or may not have a promotional offer. Specifically, the fruit market is represented as follows:

```
fruit_market = dict()
fruit_market['apple'] = dict()
fruit_market['apple']['price'] = 1.2
fruit_market['apple']['promo'] = dict()
fruit_market['apple']['promo']['nth'] = 2
fruit_market['apple']['promo']['name'] = 'Buy one apple get one free'
fruit_market['apple']['promo']['discount'] = 0

fruit_market['plum'] = dict()
fruit_market['plum']['price'] = 0.8

fruit_market['pear'] = dict()
fruit_market['pear']['price'] = 1.0

fruit_market['pineapple'] = dict()
fruit_market['pineapple']['price'] = 2.5
fruit_market['pineapple']['promo'] = dict()
fruit_market['pineapple']['promo']['nth'] = 2
fruit_market['pineapple']['promo']['name'] = 'Get 50 pence off two pineapples'
fruit_market['pineapple']['promo']['discount'] = 0.5

fruit_market['kiwi'] = dict()
fruit_market['kiwi']['price'] = 0.4

fruit_market['mango'] = dict()
fruit_market['mango']['price'] = 1.4
```

You are given a file `shopping.py` which contains the skeleton code with the functions that you will have to complete. You are also provided with a number of assertions for the functions that you need to implement. The assertions can be found in the function `main`. You should make sure that all the assertions successfully pass upon running the `shopping.py` script. This can be confirmed by receiving the message `Everything finished correctly.`

You need to implement:

1. `get_regular_cost_of_item` which, given a fruit name, returns its price as found in the `fruit_market`. `(5 marks)`

2. `check_item_has_promo` which returns `True` if there exists a promo for a given fruit name in the `fruit_market`, otherwise it returns `False`. `(5 marks)`

3. `is_cheap` which returns `True` if the price of an item is strictly less than `1.0` as found in the `fruit_market`, otherwise it returns `False`. `(5 marks)`

4. `show_cheap_items` which returns a list containing the cheap items from the `fruit_market`. `(5 marks)`

5. `items_with_promos` which, given a list of fruit names (which may or may not contain duplicates), returns a list (containing no duplicates) of items that have a promo in the `fruit_market`. `(6 marks)`

6. `total_cost_without_promos` which, given a list of fruit names (which may or may not contain duplicates), returns the total cost without taking into consideration any promos. For example, given the list of fruit `['apple', 'mango']`, the total cost without promos is `2.6` (`1.2 + 1.4`). For `['apple', 'mango', 'apple', 'pineapple']`, the total cost without promos is `6.3` (`1.2 + 1.4 + 1.2 + 2.5`). `(6 marks)`

7. `total_cost_for_item_promo_buy_n_get_one_free` which returns the total cost in situations where one item is to be given for free if `n` number of items have been bought. The function takes in the following as parameters: the cost of an item, the number of items purchased, and the number of items on which the promotional offer will be applied. For example, `total_cost_for_item_promo_buy_n_get_one_free(1.2, 3, 2)` should return `2.4` because, out of 2 items, you get one free, and you have purchased 3 items of the same type where the cost is `1.2`. `(6 marks)`

8. `total_cost_for_item_promo_buy_n_get_discount` which returns the total cost in situations where a discount is to be given if `n` number of items have been bought. The function takes in the following as parameters: the cost of an item, the number of items purchased, the number of items on which the discount will be applied, and the discount to be given. For example, `total_cost_promo_buy_n_get_discount(2.5, 3, 2, 0.5)` should return `7` because you have purchased 3 items of the same type where the cost is `2.5` and, for 2 items, you get a single discount of `0.5`. `(6 marks)`

9. `total_cost_with_promos_applied` which, given a list of fruit names (which may or may not contain duplicates), returns the total cost by taking into consideration all applicable promotional offers. `(6 marks)`

Important:

- You can assume that all items passed as parameters are valid (i.e. all items passed as parameters exist in the `fruit_market`).
- You can assume that if a fruit has a promotional offer, then it either has a discount or a free item offer but not both.
- More precisely, for a given fruit, if the discount is 0, then there is a promotional offer for which you buy `n` of those fruit and get one of them free.

Question 7

Note: This question has been modified from the 2019/20 exam to fit the style of your exam.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

In `q7/q7.py`, please complete the function `sudoku_board_valid(board)` that returns a `bool` indicating whether a given Sudoku `board` is valid.

`board` is a two-dimensional `list` of `str` representing a 9×9 grid. Each cell is filled with a digit from `"1"` to `"9"` or a `"."` representing a blank cell. You may assume that the cell will never be filled with anything other than these.

A Sudoku `board` is valid only if all the following conditions are true:

- Each row does not have any duplicated digits
- Each column does not have any duplicated digits
- Each 3x3 box does not have any duplicated digits

Sample input and output

Please see `q7/test_q7.py` for example valid and invalid boards.