National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2016/2017

Sidequest 8.2
Cheryl's Birthday

Release date: 06 March 2017
Due: 16 April 2017, 23:59

Required Files

- sidequest08.2-template.py

Background

Albert and Bernard just became friends with Cheryl, and they want to know when her birthday is. Cheryl gives them a list of 10 possible dates. Cheryl then tells Albert and Bernard separately the month and the day of her birthday respectively.

May 15	May 16	May 19
June 17	June 18	
July 14	July 16	
Aug 14	Aug 15	Aug 17

Passing by, you overhear the conversation between Albert and Bernard.

Albert	I don't know Cheryl's birthday, but I know that Bernard doesn't know too.
Bernard	At first I didn't know when Cheryl's birthday was, but I know now.
Albert	Then I also know when Cheryl's birthday is.

Using these constraints, you realize that you can create a program that can filter out the invalid cases, and identify Cheryl's birthday.

You begin by arranging the possible birthdays into a neat table, and observing that there are some dates which are unique. If Bernard is given a unique `day`, he will know what Cheryl's birthday is immediately. Similarly, if Albert is given a `month` which only has a single birthday listed for that month, he will also know Cheryl's birthday immediately.

	14	15	16	17	18	19
May		×	×			×
June				×	×	
July	×		×			
Aug	×	×		×		

Administrivia

For this sidequest, `birthday` will refer to a `month` and `day` tuple pair. The term `month` refers to the month of the `birthday` while the `day` refers to the day of the `birthday`.

For example, for the `birthday ("May", "15")`, `month` is represented by the string `"May"` and `date` is represented by the string `"15"`.

All the possible `birthdays` are stored as a tuple of tuples. The `possible_birthdays` for this scenario is provided in the template file.

This mission consists of **three** tasks.

Task 1: Unique dates and months (3 Marks)

You would like to find out if a given `day` or `month` is unique for any arbitrary tuple of possible birthdays.

(a) The function `unique_day` takes in a `day` and a tuple of `possible_birthdays` and returns `True` if the `day` is unique for the particular set of possible birthdays. Otherwise, it returns `False`. Implement `unique_day`. (1 Mark)

(b) The function `unique_month` takes in a `month` and a tuple of `possible_birthdays` and returns `True` if the `month` is unique for the particular set of possible birthdays. Otherwise, it returns `False`. Implement `unique_month`. (1 Mark)

(c) The function `contains_unique_day` takes in a `month` and a tuple of `possible_birthdays` and returns `True` if the `month` contains a unique `day` for the particular set of possible birthdays. Otherwise, it returns `False`.

Implement `contains_unique_day`. (1 Mark)

For example if the `possible_birthdays` is `(("May", "16"), ("May", "17"), ("June", "16"))`. Then,

- `17` is a **unique day** because `17` only appears once across all the months
- `June` is a **unique month** because it only has one day whereas `May` has two
- `May` contains a **unique day** `("May", "17")`
- `June` does not contain a unique because `June` only has the **repeated day** `16`

Task 2: Setting up the constraints (6 marks)

(a) *I don't know Cheryl's birthday, but I know that Bernard doesn't know too.*
Analyzing the first statement from the conversation, you realize that Albert is really saying that his given month is **not** unique. Also, for his given month, all the days are **not** unique either, so Bernard cannot know the birthday either.

The function `statement1` takes in a `birthday` and a tuple `possible_birthdays` and return `True` if the `month` of the `birthday` is not unique and does not contain a unique `day` for the particular set of possible birthdays. Otherwise it returns `False`. Implement `statement1`. (2 marks)

(b) *At first I didn't know when Cheryl's birthday was, but I know now.*
Analyzing the second statement from the conversation, you realize that Bernard has a unique `day` from the remaining possible birthdays. The function `statement2` takes in a `birthday` and a tuple, `possible_birthdays` and returns `True` if the `day` of the `birthday` is unique for the particular set of possible birthdays. Otherwise, it returns `False`. Implement `statement2`. (2 marks)

(c) *Then I also know when Cheryl's birthday is*
Analyzing the third statement from the conversation, you realize that Albert has a `month` from the remaining possible birthdays. The function `statement3` takes in a `birthday` and a tuple, `possible_birthdays` and returns `True` if the `month` of the `birthday` is unique for the particular set of possible birthdays. Otherwise it returns `False`. Implement `statement3`. (2 marks)

Task 3: And now I know her birthday, too! (1 mark)

Implement the function `get_birthday` that takes in a tuple, `possible_birthdays` and returns a tuple containing the `birthdays` that are still valid after being filtered through the constraints. If done correctly, the results should be a single element tuple consisting of Cheryl's `birthday` only.

(Hint: What are the remaining birthdays possible after imposing the constraint given by `statement1`? You might find `filter` useful here.)