National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2016/2017

Discussion Group Exercises Week 2

Tips

- You can drag tabs around and create a custom layout that suits you. Be sure to try it! Establishing a good workflow is very useful and important.
- Save often when working in the Source IDE! Be careful not to submit an outdated version of your missions. We will only entertain requests for unsubmission in exceptional cases.

Problems:

1. Suppose we're designing a point-of-sale and order-tracking system for a new burger joint. It is a small joint and it only sells 4 options for combos: Classic Single Combo (hamburger with one patty), Classic Double With Cheese Combo (2 patties), and Classic Triple with Cheese Combo (3 patties), Avant-Garde Quadruple with Guacamole Combo (4 patties). We shall encode these combos as 1, 2, 3, and 4 respectively. Each meal can be *biggie-sized* to acquire a larger box of fries and drink. A *biggie-sized* combo is represented by 5, 6, 7, and 8 respectively, for combos 1, 2, 3, and 4 respectively.

   Write a function named `biggie_size` which when given a regular combo returns a *biggie-sized* version.

2. Write a function named `unbiggie_size` which when given a *biggie-sized* combo returns a non-*biggie-sized* version.

3. Write a function named `is_biggie_size` which when given a combo, returns true if the combo has been *biggie-sized* and false otherwise.

4. Write a function named `combo_price` which takes a combo and returns the price of the combo. Each patty costs $1.17, and a *biggie-sized* version costs $.50 extra overall.

5. An order is a collection of combos. We will encode an order as each digit representing a combo. For example, the order 237 represents a Double, Triple, and *biggie-sized* Triple.

   Write a function named `empty_order` which takes no arguments and returns an empty order which is represented by 0.

6. Write a function named `add_to_order` which takes an order and a combo and returns a new order which contains the contents of the old order and the new combo. For example, `add_to_order(1, 2) -> 12`.

7. Write a function named `last_combo` which takes an order and returns the last combo. For example, `last_combo(321) -> 1`

8. Write a function named `other_combos` which takes an order and returns a new order without the last combo. For example, `other_combos(321) -> 32`