CS3217

Problem Set 1

Matric No
A0140036X

Name
Joel Lim Jing

Total Score: **20** /40

Item	Score	Remarks
Problem 1	1 /2	-1: For not explaining that hashing provides fast O(1) access to the key stored in the dictionary
Problem 2.1	6 /6	
Problem 2.2	6 /6	
Problem 2 subtotal	12 /12	
Problem 3.1	2 /8	You could have used your Stack DS here. -2: Non-lazy implementation of DFS generator. -2: No implementation provided for the case when init fails, e.g. when graph is empty, or when start node is not found in the graph. -2: Incorrectly handled the case when key is not found (in fact it is returned as the first DFS element!)
Problem 3.2	8 /8	You could have used your Queue DS here. -0: Non-lazy implementation of DFS generator. -0: No implementation provided for the case when init fails, e.g. when graph is empty, or when start node is not found in the graph. -0: Incorrectly handled the case when key is not found (in fact it is returned as the first BFS element!)
Problem 3.3	1 /2	- Optional chaining part was accepted as an answer. - "Truely an error" is vague.
Problem 3 subtotal	11 /18	
Problem 4	4 /8	- Covered the bare minimum, could have written more test cases for the DSes to cover more corner cases. E.g. interweaving pushing and popping. E.g. test more types of graph. E.g. the case of non-existent nodes.
Bonus from reflection	1 /1	
Coding style deduction	-5	Please see inline comments. -1: not keeping internal properties private. -1: unnecessary print statement -1: should instantiate at properties instead of init. -1: incorrect spacing -2: prefer guard statement Style -6, capped at -5
Late penalty	-4	
Tests passed (/47)	43	BFS and DFS failed: - initialisation test (failable init) - empty graph

Problem Set 1

Matric No	Name
A0140036X	**Joel Lim Jing**

Total Score: **20** /40

Item	Score	Remarks
Comments		- Congrats in completing your first PS! □ - Please submit your work on time in the next few PSes! - In the event that you really exceed the deadline, you may fully utilise the remaining day to refine your work. - Please embrace the use of `guard` and `guard ... let` statements. - Strive to write cleaner and elegant code! - Make use of functional programming too, such as "forEach..." - Should read the Swift documentation to find the best method to suit your need.