National University of Singapore
School of Continuing and Lifelong Education

TIC1002: Introduction to Computing and Programming II
Semester II, 2020/2021

Tutorial 5
Abstract Data Types

1. We will be implementing an `NUSModule` ADT, and use a vector of NUSModule objects to calculate a CAP score.

   The `NUSModule` ADT needs to be able to store i) the module code, ii) the number of modular credits that it is worth, iii) the grade obtained for the module, and iv) whether it has been declared S/U.

   All completed modules are given a letter grade from A+ to F, but it can also be declared as an S/U grade at a later time (assume all modules can be declared as S/U). Whether a module gets an S or U is determined by the grade as shown in the table below.

Grade	A+	A	A-	B+	B	B-	C+	C	D+	D	F
Point	5.0	5.0	4.5	4.0	3.5	3.0	2.5	2.0	1.5	1.0	0.0
S/U	S	S	S	S	S	S	S	S	U	U	U

   (a) Decide on an internal representation of `NUSModule` and provide an implementation.

   (b) Decide what constructor, accessors, predicates and printers are necessary or useful to have, and provide their implementation. You may assume that the code, credits and grade will not be modified after creation.

   (c) Implement a function `credits_obtained` that calculates the total modular credits obtained, given a vector of NUSModules. Note that modules with F or U grade will count as 0 credits.

   (d) Suppose you are given a map that maps a grade to its point value.

   Implement a function `calculate_cap` that calculates the CAP, given a vector of NUSModules along with this grade point map.

   Note that modules with S or U grade will not be included in the CAP computation.

2. In lecture we discussed two different implementation of Queue: 1) using a vector and 2) using two stacks. Intuitively, we know that implementation 2 is more efficient than implementation 1, but let us try to quantify it.

   Come up with a sequence of queue operations that

   (a) provides the best case performance for implementation 1.

   (b) provides the worst case performance for implementation 1.

   (c) provides the best case performance for implementation 2.

   (d) provides the worst case performance for implementation 2.

For each of the operations above, show the time complexity for both implementation.

3. For this question, we will try our hand at using templates.

   We have discussed in lecture that there is no built-in function to obtain a sequence of keys or values, unlike other modern programming languages. But no worries, what is not provided, we can do ourselves.

   We can implement a function `get_keys` that takes in a map, and returns a vector containing the keys, and `get_values` that returns a vector containing the values.

   Here are the possible ways we can implement such a function:

   - `void get_keys(vector<K> *v, const map<K, T> m);`
   - `void get_keys(vector<K> &v, const map<K, T> m);`
   - `vector<K> & get_keys(const map<K, T> m);`
   - `vector<K> get_keys(const map<K, T> m);`

   (a) Discuss the differences of each of the implementation.

   (b) Should the map also be passed as a pointer, reference or by value?

   (c) Finally, decide on one and provide an implementation for `get_keys` and `get_values`.