Unity Montreal - Developer Quiz

# Question 1:

* I prefer Option 1.
* Pros of Option 1: Simple, high level, easy to read, short, fast when compiler optimizer is on, less memory used
* Cons of Option 1: Slow without compiler optimization enabled
* Pros of Option 2: Fast without compiler optimization due to a larger for-loop body
* Cons of Option 2: Hard to read, complex, more memory used

# Question 2:

* There is redundancy among the POST API’s update.
* A response code of 200 usually means success, not error.

# Question 3:

* The used salt is the same for all passwords and is very short.
* The salt is intialized as a field literal in the class.
* The new password is not checked against password requirements.
* The encryption method MD5 is known for having significant vulnerabilities.

# Question 4:

* Additional documentation is appreciated.
* Specify what the function is supposed to do and what the arguments are.
* Change the function into a non-static function in the user class.

# Question 5:

* The “update” method is using “fromAccount.credits” instead of a $inc operator, which may lead to setting the wrong credits when a parallel transfer is occuring. (Similar situation for the “toAccount.credits”)
* The function does not check if the “findOne” method has found no account or one account from many others with the same name.

# Question 6

* There is no safety check on whether the ‘amount’ is legal, eg. not NaN.
* The balances belonging to the instance may not have been initialized.
* ‘Account.prototype’ can be undefined.

# Question 7

* Use good indices
* Simplify the queries, eg. avoid complicated SQL queries
* Add redundant data to tables (denormalization) to avoid joins
* Improve the hardware