*Hash  
Cracking*

52093180  
CS4028 – Assessment 1  
University of Aberdeen

# Cracking Class

## Password Data Class

When an instance of Cracking is initiated with an array of hashes, a set called passwords is created for the class comprising the hashes structured as Password data objects. Password objects have values for hash, salt, password, cracked, and attempts. When initiated only hash and possibly salt will hold a value. Once a hashes password is found cracked will = True and the password and attempts value will be populated.

## \_crack(passwordStream):

As the logic is much the same for Tasks 1-3 I created a function that could be passed a stream of passwords that will then be iteratively hashed and compared against uncracked hashes in the passwords set. If the uncracked hashes are salted, each password in the stream will be individually salted then hashed and compared to its corresponding uncracked salted hash.

***Explain***

This means if passwords are salted, rather than a password being hashed once, it will be hashed as many times as there is uncracked passwords

# Tasks

## Task 01

Task01.py initiates Cracking with inputted hashes and calls bruteForce(). BruteForce() creates a password stream bruteForceSteam() which yields rebase(i) with an incrementing i on each call. Rebase() takes a base10 integer and converts it to the base of the provided alphabet - the default being base36 comprising integers and lowercase characters. bruteForceSteam() is passed to \_cracker() which then iterates through all natural number in base36 until the input hashes are cracked. The list of cracked passwords is then output. As attempted passwords are never repeated, they’re hashes are not stored.

## Task 02

## Task 03

NEED TO ALLOW HASH INPUT? WORKS for T1 But not T2?