COMP1201 Assignment 2

(due 24th April, 4pm)

This assignment is worth 5% of the module marks. Submission is via the Handin system.

Q1 Consider the following hash function with separate chaining: it takes a non-negative integer $n < 100000$ and calculates its hash value as $(2d_1 + 3d_2 + 5d_3 + 7d_4 + 11d_5)\%47$, where % is modulo division, d_1 is the most significant digit, and d_5 is the least significant one (and the others are in decreasing significance order). Suppose we have 2000 numbers. Prove that there exists a number x between 0 and 46 for which we can find at least 43 numbers among the given 2000, whose hash value is exactly x. [2 marks]

Q2 Design a data type that supports the following operations: insert, delete the maximum, and delete the minimum (all in logarithmic time); and find the maximum and find the minimum (both in constant time). **Hint:** Use two heaps. Describe your solution in either pseudocode (preferred) or English. You can refer to helper functions from the lectures. Your answer should include a brief explanation why your data type achieves the desired complexity. [3 marks]

Q3 (a) Implement Quick-Union (without the weighted heuristic or path-compression) in Java. Use your implementation to create a graph with 256 vertices A (numbered 0 to 255). Connect the vertices $A[i]$ with $A[i + 1]$ for i ranging from 0 to (and including) 254 by increments of 2. Connect the vertices $A[j]$ with $A[j + 3]$ for i ranging from 0 to (and including) 200 by increments of 3. How many connected components are there in the resulting graph? What is the root of node 19, i.e. what is Find(19)? What is Find(112)? [3 marks]

(b) Now implement Weighted Quick-Union (without path compression), correctly keeping track of the size (i.e. the number of elements) of the sub-trees and always merging the smaller tree into the larger one during Union. Generate the same graph as before. What are the root elements of nodes 19 and 112? [2 marks]

(You should provide both implementations as part of your submission for this question.)

Q4 Explain how one can modify any sorting algorithm to have $O(n)$ best case time complexity. [1 marks]

Q5 Describe how an unstable sorting algorithm can be turned into a stable sorting algorithm. [1 marks]

Q6 Write down in pseudocode a recursive version of the Selection Sort algorithm and write down the recurrence relation for its worst-case running time. [3 marks]