Department of Computing and Information Systems
COMP20007 Design of Algorithms
Semester 1, 2013
SAMPLE Mid Semester Test WITH ANSWERS

1 Instructions

- You must have your student card on display during this test.
- The test will start at 5:30pm and finish at 6:00pm.
- The total time allowed for this test is 30 minutes.
- Answer all questions on this paper.
- Remember, -1 marks for incorrect answer in true/false (don't guess).

Question 1 [10 marks, minimum 0 marks]

Answer True or False for each of these statements. You will gain one mark for a correct answer, and **lose** one mark for an incorrect answer.

0.	$f(n) = 12 \times \log(n^2)$ is in $O(\log n)$	T
1.	$f(n) = 12 \times \log(n^2)$ is in $\Omega(\log^* n)$	T
2.	$f(n) = 1.8^n$ is in $O(n)$	F
3.	If $T(n) = 8T(n/2) + \Theta(1)$, then $T(n)$ is in $O(n)$	F
4.	If $T(n) = 2T(n/6) + \Theta(n)$, then $T(n)$ is in $\Omega(n^2)$	F
5.	Any function that is in $O(n^2)$ is also in $\Omega(\log n)$	F
6.	A program that has a worst case running time in $O(n)$ will always run faster than a program that has a worst case running time in $O(n \log n)$	F
7.	All comparison sorts require $O(n \log n)$ in the worst case	F
8.	Counting Sort for sorting n elements has worst case running time $\Theta(n)$	F
9.	DFS can process a graph with n vertices and m edges in $O(mn)$ time	T

Question 2 [4 marks]

Write pseudo code for a function that partitions an array A containing n elements in-place into odd and even numbers, with the odd numbers filling the left of the array and the even numbers filling the right hand end of the array.

```
// assumes A is not NULL
function partition(A, n)
    left = 0
    right = n-1
    while (left < right)
        if A[left] is even
            swap array[i] and array[right]
            Decrement right
        else
            Increment left
```

Question 3 [6 marks]

(a) What is the definition of a Strongly Connected Component (SCC) in a directed graph?

For each vertex in the SCC, there is a path to any other vertex in the SCC.

(b) Outline an algorithm for finding all SCCs give a graph $G(V, E)$. You may assume that you can call DFS without needing to describe it in detail.

INPUT: A directed graph $G(V, E)$
OUTPUT: G with each vertex labelled with its SCC number

1. Let G' be G with all the edges reversed.
2. Perform DFS on G' keeping track of post-numbers for each vertex.
3. While no vertices left
4. Choose the remaining vertex with highest post number from Step 2, v.
5. Label all vertices that can be reached during a DFS from v as a SCC.
6. Remove all the marked vertices from the graph.