COMP108 Algorithmic Foundations
Tutorial 6
w/c 10th March 2014

Name: _____

Hand this in to the demonstrator at the end of the tutorial even if you haven't finished it. You will get feedback in the next tutorial. Tutorial participation contributes to 5% of overall marks.

1. What is the time complexity (in Big-O notation) of the following pseudo code?

   $a = 1$, $b = n$, $c = \lfloor \frac{a+b}{2} \rfloor$
   while $a \leq b$ **do**
   begin
   Output c
   if c is odd **then**
   $a = c + 1$
   else // i.e., c is even
   $b = c - 1$
   $c = \lfloor \frac{a+b}{2} \rfloor$
   end

2. (a) Describe a divide-and-conquer algorithm to find the sum of the numbers in an array $A[]$ with n integers $A[1]..A[n]$.

(b) Explain why the time complexity $T(n)$ of your algorithm in (2a) can be described by the following recurrence.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 \times T(\frac{n}{2}) + 1 & \text{if } n > 1 \end{cases}$$

(c) Consider the function $T(n)$ defined in (2b) Prove that $T(n)$ is $O(n)$ by the *substitution method*, i.e., use mathematical induction.
We are going to prove that $T(n) \leq 2 \times n - 1$ for all $n \geq 1$.

Base case:
When $n = 1$, L.H.S. $T(1) =$ _____ R.H.S. = _____
Therefore, L.H.S. = R.H.S.

Induction hypothesis: Assume that the property holds for all integers $n' < n$, i.e., assume

$$T\left(\frac{n}{2}\right) \leq 2 \times \frac{n}{2} - 1$$

Induction step:
We want to prove $\boldsymbol{T(n) \leq 2 \times n - 1}$.
[The induction step can be proved by first using the recurrence to express $T(n)$ in terms of $T(\frac{n}{2})$, and then use the hypothesis.]

L.H.S. =

Therefore, L.H.S. $\leq$ R.H.S. and the property holds for n.
Conclusion: $T(n) \leq 2 \times n - 1$ for all positive integers n and therefore, $T(n)$ is $O(n)$.