The University of Melbourne
Department of Computing and Information Systems

COMP90038 - Algorithms and Complexity
Sample Mid-semester test*: Semester 1, 2015

* This is only a sample test for your reference. Types of questions asked may differ. Also, the number of questions and the allocation of marks for each question may defer.

Identical examination papers: None
Exam duration: Forty-five (45) minutes
Reading time: Five (5) minutes
Length: This paper has TWO (2) pages including this cover page (double sided).

Authorized materials: No materials are authorized. Calculators are not permitted.

Instructions to invigilators: Students may not remove any part of the examination paper from the examination room. Students should be supplied with the exam paper and a script book, and with additional script books on request.

Instructions to students: *All questions should be answered* by writing a brief response or explanation on the lined pages in the script book. The reverse side of any page may be used to make rough notes, or prepare draft answers. Please write your student ID below and on your script book. When you are finished, place the exam paper inside the front cover of the script book.

- This test is worth 10 marks, which will count as 10% of your final grade.
- Unreadable answers will be deemed wrong.
- Start your answer to each question on a new page.
- Be sure to clearly number your answers.
- Use a blue or black pen (not a pencil).

Students ID number:

Examiners' use only:

Total [10]

Answer all the questions (10 marks)

Question 1 (*2 marks*):
Consider the following algorithm in pseudo-code.

```
ALGORITHM DOES_SOMETHING (A [0..n - 1, 0..n - 1])
    for i : = 0 to n - 2 do
        for j : = i +1 to n - 1 do
            if A[i, j] ≠ A[j, i]
            return false
        return true
```

i. What does this algorithm compute?
ii. What is its basic operation?
iii. How many times is the basic operation executed?
iv. What is the efficiency class of this algorithm? Show your working.

Question 2 *(0.5 mark)*:
State whether the following statement is true or false: "The worst case complexity of selection sort is $O(n \log n)$".

Question 3 *(0.5 mark)*:
State whether the following statements is true or false: "Merge sort is a stable algorithm".

Question 4 *(2 marks)*:

Here is the Master Theorem that states that if $f(n) \in \Theta(n^d)$ where $d >= 0$ in the recurrence equation: $T(n) = aT(n/b) + f(n)$, then

$T(n) \in \Theta(n^d)$ if $a < b^d$

$T(n) \in \Theta(n^d \log n)$ if $a = b^d$

$T(n) \in \Theta(n^{\log_b a})$ if $a > b^d$

a) To what family of algorithms does the Master Theorem apply?
b) An algorithm called **funsort,** at each iteration, examines every item in the file, then, splits the file into *three* parts, and recursively examines only *two* of the parts. Write a recurrence that describes the cost or running time, *T (n),* for this algorithm **funsort** in terms of the number of items *n.* Using the Master theorem, write the recurrence relation and find the efficiency class for the recurrence relation.

Question 5 *(2 marks)*:
When discussing sorting algorithms we talk about the behavior on files that are already sorted. Why are we interested in how long it takes to sort a file that is already sorted? Explain your answer *briefly*.

Question 6 *(3 marks)*:
Design an algorithm to check whether two given words are anagrams. That is, whether one word can be obtained by permuting the letters of the other. For example, the words *tea* and *eat* are anagrams; *team* and *meat* are anagrams. You may write your algorithm in pseudo-code or any programming language. Then, analyze the complexity of your algorithm in Big-Oh notation. Show your working.
