These exercises are worth 2.5% of your final grade and are due at 5pm, Friday, August 17, 2012.

In this lab you are required to implement a number of sorting algorithms. You should download the interface Sort.java from the unit webpage and implement the specified methods in a class called Sorter.java. Your sort class should also maintain a variable to count the number of array assignments performed by each algorithm, as described in the interface Sort.java. Implement the following algorithms:

1. **procedure** INSERTION-SORT(A)
   
   ```java
   for j ← 2 to length[A]
   do key ← A[j]
       ▷ Insert A[j] into the sorted sequence A[1...j−1]
       i ← j − 1
       while i > 0 and A[i] > key
       do A[i + 1] ← A[i]
           i ← i − 1
       A[i + 1] ← key
   ```

   1 mark

2. **procedure** QUICKSORT(A, p, r)
   
   ```java
   if p < r
       then q ← PARTITION(A, p, r)
           QUICKSORT(A, p, q − 1)
           QUICKSORT(A, q + 1, r)
   ```

   **procedure** PARTITION(A, p, r)
   
   ```java
   x ← A[r]
   i ← p − 1
   for j ← p to r − 1
       do if A[j] ≤ x
           then i ← i + 1
   return i + 1
   ```

   2 marks
3. procedure MERGE-SORT(A, p, r)
   if p < r
     then q ← ⌊(p + r)/2⌋
     MERGE-SORT(A, p, q)
     MERGE-SORT(A, q + 1, r)
     MERGE(A, p, q, r)

procedure MERGE(A, p, q, r)
  n₁ ← q − p + 1; n₂ ← r − q
  allocate arrays L[1...n₁ + 1] and R[1...n₂ + 1]
  for i ← 1 to n₁
do L[i] ← A[p + i − 1]
  for j ← 1 to n₂
do R[j] ← A[q + j]
  L[n₁ + 1] ← ∞; R[n₂ + 1] ← ∞
i ← 1; j ← 1
  for k ← p to r
do if L[i] ≤ R[j]
    then A[k] ← L[i]
      i ← i + 1
    else A[k] ← R[j]
      j ← j + 1

1 mark

Record the CPU time taken by each algorithm for various inputs. Compare the output of the UNIX `time` command with the number of array assignments performed by each algorithm (Hint: use `man time` to find out how to interpret the values output by `time`). Repeat the tests for MergeSort and QuickSort and verify the theoretical rate of growth for these algorithms. You may also wish to compare against the running times implemented in in the Java class, Arrays.

Note that as an automated marker is used you will not be able to include print statements, java.util classes etc in your implementation. However, a test class `SortTest.java` is available for your convenience.

Submit only the file `Sorter.java` to https://secure.csse.uwa.edu.au/run/cssubmit. There is an automated script that will compile and run your code, and estimate your final mark. This feedback ensures that you will not lose marks for small errors, but is no substitute for your own thorough testing.