Comments:

Q1:

2/3 of the students took this question, and the results, on the whole, varied from good, to excellent to outstanding, and the average is very high - over 70%. Almost all students completed the question, very few got below 50%, and some of the answers showed great invention and the development and description of complex algorithms. Well done everyone!

A few points -
(a) almost all gave sort and iterate from both ends, but some considered factors and complementary factors (and some had clever ideas on finding these).
(b) majority element: if sorted, and there is a majority element then it must be in the middle of the list! Variations on this, and also the use of hash counting, and clever linear double-pass algorithms all got full marks.
(c) choosing k distinct elements at random. The most common mistake (over half the solutions) was not to notice that repeated choice from a fixed list may not terminate when duplicates are rejected!

Question 2 on Sorting was popular: 142 students took the question.

On the whole, the question was answered well by the large majority.
Parts (a) and (b) were answered well.

A fairly common mistake in part (c) was missing some key informationout of the description of the merge operation used in merge sort, particularly that the two input lists to be merged must themselves be sorted.

Part (d) was challenging for most. This concerned using a techniqueto make quicksort stable (for integer keys) by transforming the keys based on their initial list index. It was high school mathematics to find a transformation back to the original key values after sorting, but this transformation does not work if it includes the need to know the initial index (as this information will be lost during sorting). 3 out of 4 marks were awarded if it was noticed that the initial index needs to be stored for use, and giving some plausible way of doing that. Full marks were awarded for the very few answers that observed that the initial index value is not needed, and should not be used, in the transformation back.

Q3 Complexity

Statistics:
121 students took the question. The mean mark was 12.74 / 20 = 64% with a standard deviation of 3.48 marks.

100 of 121 students got 50% or more. 111 of 121 got 40% or more. 1 student scored full marks.

Commentary:

Part (a) tested understanding of big-Oh notation and the relative growth rates of common functional forms. It was answered well generally. However, a quite high portion of students did not know that "log(log(n))" is a slower growing
function than "log(n)".

(b) This tested more about big-Oh, big Theta, and big Omega, and the use of common terms like "exponential", "polynomial" to describe functions. Many students got full marks. A common error was thinking that "n log n" was "linear" (= BigTheta(n)).

(c) This tested whether students understood the advantages of asymptotic notationfor describing algorithm complexity. Some good answers were given, indicating sound understanding, but some students struggled to express themselves precisely
enough and so dropped some marks.

(d) This part tested the ability of students to analyse some pseudocode and provide a simple correctness argument and complexity argument. This part was harder for most students, and very few obtained full marks. Usually some correct observations about the code were made, but these were not put into a logical, sound argument.