Problem 1 [9 marks - 3 per part]

Consider the running time of Quick Sort, Merge Sort, Heap Sort, and Insertion Sort to answer the following questions:

A. What is an advantage of Quick Sort over Merge Sort?
B. What is an advantage of Insertion Sort over Merge Sort?
C. What is an advantage of Heap Sort over Quick Sort?

Problem 2 [12 marks - 6 per part]

Write pseudocode for each of the following problems, and justify the running times of your solutions:

A. Given an array of $n + 2$ integers, all of which are between 1 and $n$, containing exactly two duplicates (that is, two numbers that appear exactly twice), give a $O(n)$ time algorithm for finding the duplicated elements.

B. Given an array of $n + 5$ integers, with exactly 5 duplicates, find the duplicated elements in time $O(n \lg n)$.

Problem 3 - [41 points - 8+5+8+15+5]

Consider the following problem:

We are given a rod of length $m$ for some positive integer $m$, and positive integers $s_1, s_2, \ldots, s_k$ where $s_1 = 1$ and $s_i < m$ for all $2 \leq i \leq k$, representing valid sizes for smaller rods.

Our task is to cut the rod of length $m$ into pieces, so that each piece is of length $s_i$ for some $1 \leq i \leq k$, and the number of pieces is minimized.

Example: Given a rod of length 52, and positive integers 1, 5, and 10 for the valid sizes of smaller rods, the optimal solution is to divide the rod of length 52 into five pieces of length 10 and two pieces of length 1.
A. Write pseudocode for a greedy algorithm for the above problem. Hint: Make the first cut as big a possible, followed by next largest possible cut, etc.

B. Analyze the running time of the above algorithm.

C. Prove that this algorithm is not optimal if the valid sizes of smaller rods are 1, 3, and 4.

D. Come up with a dynamic programming algorithm to this problem that would be all optimal for all instances of this problem. Justify your solution.

E. Analyze the running time of your dynamic programming solution.

Problem 4 [6+10+10 = 26 marks]

Consider the following (not necessarily best) implementation of an algorithm that finds the largest element in an array.

```python
function max-element(A)
    if n = 1
        return A[1]
    val = max-element(A[2...n])
    if A[1] > val
        return A[1]
    else
        return val
```

A. Give the best case running time of this algorithm using Θ-notation. Justify your answer.

B. Give a recurrence relation $T(n)$ for the running time of this algorithm in the worst case.

C. Prove that $T(n) \in O(n)$ using the substitution method.

Problem 5 [6+6=12 marks]

Solve the following recurrences using the Master Theorem.

A. $T(n) = 2T(n/4) + n \lg n$

B. $T(n) = 5T(n/3) + 2^n$