Problem 1. [10 points]
Solve the following recurrences. Assume \( T(n) = 1 \) for \( n < 2 \). Express your solutions in Big-Oh notation, and show your work.

a) \( T(n) = 4T(n/2) + n \)
b) \( T(n) = T(n/2) + 1 \)
c) \( T(n) = T(n - k) + n, \ k > 0 \)

Problem 2. [5 points]
Decide whether you think the following statements are true or false. If it is true, give a short explanation. If it is false, give a counterexample.

1. In every instance of the Stable Matching Problem, there is a stable matching containing a pair \((m, w)\) such that \( m \) is ranked first on the preference list of \( w \) and \( w \) is ranked first on the preference list of \( m \).

2. Consider an instance of the Stable Matching Problem in which there exists a man \( m \) and a woman \( w \) such that \( m \) is ranked first on the preference list of \( w \) and \( w \) is ranked first on the preference list of \( m \). Then in every stable matching \( S \) for this instance, the pair \((m, w)\) belongs to \( S \).

Problem 3. [5 points]
(a) Suppose algorithm A has worst-case running time \( O(n) \) and algorithm B has worst-case running time \( O(n^2) \). What can you say about the relative performance of the algorithms?
(b) Repeat part a for the worst-case running times \( \Theta(n) \) and \( \Theta(n^2) \).

Problem 4. [10 points] - Proof Module - Submit Separately in class
Use mathematical induction to show that when \( n \) is an exact power of 2, the solution of the recurrence

\[
T(n) = \begin{cases} 
2, & \text{if } n = 2 \\
2 \cdot T(n/2) + n, & \text{if } n = 2^k, \ k > 1
\end{cases}
\]

is \( T(n) = n \lg n \). (Hint: Your induction should proceed on \( k \), not on \( n \)).

Problem 5. [10 points]
Insertion sort can be expressed as a recursive procedure as follows. In order to sort \( A[1..n] \), we recursively sort \( A[1..n-1] \) and then insert \( A[n] \) into the sorted array \( A[1..n-1] \).

(a) Write pseudocode for this recursive version of Insertion sort
(b) Give a recurrence for the running time of the algorithm
(c) Solve that recurrence to find the Big-Oh running time of the algorithm

Problem 6. [10 points] - Coding question
In order to asymptotically analyze the running time of an algorithm, one has to think about how the data will be represented and manipulated in an implementation of that algorithm, so as to bound the number of computational steps it takes. In this problem, you will use arrays to implement the different high-level operations in the Gale-Shapley algorithm presented in Chapter 1, and shown in Figure 1.

Write a Java function to implement the following Gale-Shapley algorithm pseudocode with a worst-case
Initially all $m \in M$ and $w \in W$ are free

While there is a man $m$ who is free and hasn’t proposed to every woman

Choose such a man $m$
Let $w$ be the highest-ranked woman in $m$'s preference list to whom $m$ has not yet proposed
If $w$ is free then
$(m, w)$ become engaged
Else $w$ is currently engaged to $m'$
If $w$ prefers $m'$ to $m$ then
$m$ remains free
Else $w$ prefers $m$ to $m'$
$(m, w)$ become engaged
$m'$ becomes free
Endif
Endif
Endwhile

Return the set $S$ of engaged pairs

Figure 1: The Gale-Shapley algorithm.

running time complexity of $O(n^2)$ that follows the input and output specifications below.

Your function should have the following definition:

```java
public int[] GaleShapley(int[][] m-pref, int[][] w-pref)
```

**Input Types:**
Two 2D arrays of Integer; one for the men’s preferences and the other for women’s preferences. You could define them as int[][] m-pref, int[][] w-pref.

**Output Types:**
The output format is a single Integer array; in which the value in index $i$ represents the women ID that is matched with the man with ID $i$.

**Notes:**
Since we are programming, the first man is referred to as man 0, first woman is referred to as woman 0. Also, assume all inputs are valid, you don’t have to verify them. In other words, length of m-pref == length of w-pref, and the preference subarrays contain every person from the opposite list once.

**Example 1:**
m-pref = [[0, 1, 2], [0, 2, 1], [2, 1, 0]]
w-pref = [[1, 2, 0], [2, 0, 1], [0, 1, 2]]
Outputs [1, 0, 2], representing man 0 with woman 1, man 1 with woman 0, man 2 with woman 2

**Example 2:**
m-pref = [[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
w-pref = [[0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5], [0, 1, 2, 3, 4, 5]]
Outputs [0, 1, 2, 3, 4, 5]