1. (3) Show using definition of $\Theta$ that $\frac{1}{2}n^2 - 5n = \Theta(n^2)$

$$\frac{1}{2}n^2 - 5n \leq cn^2 \quad (1)$$

$$\frac{1}{2}n^2 - 5n \leq cn^2, \text{ for } c = 1 \text{ and } n_0 > 2 \quad (2)$$

$$\frac{1}{2}n^2 - 5n \geq cn^2 \quad (3)$$

$$\frac{1}{2}n^2 - 5n \geq cn^2 \text{ for } c = 0.1 \text{ and } n_0 > 11 \quad (4)$$

2. (5) For the following pair of functions indicate whether $f(n)$ is $O, \Omega, \Theta$ of $g(n)$:

- $n^k, c^n$ Answer $n^k = O(c^n)$
- $2^n, 2^{n/2}$ Answer $2^n = \Omega(2^{n/2})$
- $n^2, n \log n$ Answer $n^2 = \Omega(n \log^2 n)$

3. (4) Chapter 1, Problem 1 (True or False) 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 and $w$ is ranked first on the preference list on $m$. FALSE Counter example is the stable matching X-A, Y-B Z-C which does not have that property.

<table>
<thead>
<tr>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Y</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

4. (5) Chapter 2, Problem 1 c, d

- $100n^2$ double the input $100(2n)^2 = 400n^2$. The algorithms will run 4 times slower.
- $n \log n$ double the input $2n \log(2n) = 2n(\log n + \log 2) = 2n \log n + 2 \log 2$. The algorithms will run 2 times slower.

5. (5) Chapter 2, Problem 2 c, e

$100n^2 = 60 \times 60 \times 10^{10}$ and solve for $n$ gives us $n = 6 \times 10^5$.

$2^n = 60 \times 60 \times 10^{10}$ and solving for $n$ gives us $n = \log_2 3600 \times 10^{10}$ this is rounds up to $n = 13$.


Worst case running time for finding a minimum in $n$ dimensional array is going to take $n$ steps. Then we have to repeat that operation for the remaining $n - 1$ dim array, which is going to take $n - 1$ steps, etc. Hence the worst case running time is $n + (n - 1) + (n - 2) + \ldots + 2 = \Theta(n^2)$.
Practice Problems (not for grade)

1. Chapter 1, Problem 4

   The algorithm is very similar to Gale-Shapley algorithm; students are either committed or free and hospitals either have positions of they are full.

   While some hospital has available positions
   
   \[ h_i \text{ offers job to the next student on its preference list} \]
   
   if \( s_j \text{ is free} \) then \( s_j \text{ accepts the offer} \)
   
   else (\( s_j \text{ is already committed to a hospital } h_k \))
   
   if \( s_j \text{ prefers } h_k \text{ to } h_i \text{ then it remains with } h_k \)
   
   else \( s_j \text{ becomes committed to } h_i \)

   the number of available positions at \( h_k \) increases by one
   
   the number of available positions at \( h_i \) decreases by one

   This algorithm terminates in \( O(mn) \). Suppose that there \( p_i > 0 \) positions at the hospital \( h_i \). The algorithm terminates with all positions filled, if not that would contradict our assumption that number of students is greater than number of positions. To argue for stability, check for two types of instability and prove by contradiction appealing to the algorithm that then cannot happen. Very similar to the stale marriage case proof.

2. Chapter 2, Problem 3

   We know from the chapter that polynomials grow slower than exponentials so we can consider \( f_1, f_2, f_3, f_6 \) as one group and \( f_4 \) and \( f_5 \) as other group. For polynomials we can order them by their exponent so \( f_2 \) is before \( f_3 \) before \( f_1 \). \( f_6 \) grows faster than \( n^{2} \) but slower than \( n^{c} \) for some \( c > 2 \). We can insert \( f_6 \) in this order between \( f_3 \) and \( f_1 \). Exponentials can be ordered by their bases, so we can put \( f_4 \) before \( f_5 \). So the functions in the ascending order of growth rate

   \[ \sqrt{2n}, n + 10, n^2 \log n, n^{2.5}, 10^n, 100^n \]