1. **The Assignment Problem**

   (a) **Implementation**

   Auction Algorithm in Python gives the following results.

   - O1 matches A9
   - O2 matches A4
   - O3 matches A8
   - O4 matches A3
   - O5 matches A6
   - O6 matches A10
   - O7 matches A7
   - O8 matches A5
   - O9 matches A2
   - O10 matches A1

   The object value is 847.

   (b) **Random problems and assignment values**

   Auction Algorithm in Python gives the following results for different number of agents.

   - 1000 trials, 2 agents of M 100 in 0:00:00.039298
     max objective: 197; average per agent: 61.885
   - 1000 trials, 4 agents of M 100 in 0:00:00.084385
     max objective: 382; average per agent: 74.06925
   - 1000 trials, 8 agents of M 100 in 0:00:00.301748
     max objective: 757; average per agent: 84.292875
   - 1000 trials, 16 agents of M 100 in 0:00:01.076487
     max objective: 1546; average per agent: 91.346125
   - 1000 trials, 32 agents of M 100 in 0:00:03.781173
     max objective: 3120; average per agent: 95.5595
   - 1000 trials, 64 agents of M 100 in 0:00:13.309321
     max objective: 6310; average per agent: 97.90771875
   - 1000 trials, 128 agents of M 100 in 0:00:46.555478
     max objective: 12721; average per agent: 99.0917890625
   - 1000 trials, 256 agents of M 100 in 0:02:45.659303
     max objective: 25547; average per agent: 99.67412890625

   We observe that the average per agent increases. Because we randomize the matrix for preference values, according to probability, it is more likely to obtain a larger objective value when we have more agents and more combinations.

   (c) **Timing**

   For different maximum values, Auction algorithm in Python gives the following results.

   - 1000 trials, 256 agents of max 10 in 0:02:41.212464
   - 1000 trials, 256 agents of max 100 in 0:02:34.458246
   - 1000 trials, 256 agents of max 1000 in 0:04:38.710950
   - 1000 trials, 256 agents of max 10000 in 0:09:26.463212
   - 1000 trials, 256 agents of max 100000 in 0:15:54.299111
   - 1000 trials, 256 agents of max 1000000 in 0:23:17.291773
   - 1000 trials, 256 agents of max 10000000 in 0:24:35.980179

The linear programming approach using Pulp gives the following results:

- 100 trials, 256 agents of max 10 in 0:12:15.812380
- 100 trials, 256 agents of max 100 in 0:13:19.055558
- 100 trials, 256 agents of max 1000 in 0:13:20.868162
- 100 trials, 256 agents of max 10000 in 0:13:14.210086
- 100 trials, 256 agents of max 100000 in 0:13:13.487019
- 100 trials, 256 agents of max 1000000 in 0:13:34.497067
- 100 trials, 256 agents of max 10000000 in 0:14:06.734649

We observe that the computing time increases quickly in respect of maximum value $M$ for auction algorithm but grows smoothly for the linear programming tool. Thus, we can conclude that $M$ has a greater impact on computing time in Auction Algorithm than using linear programming approximation. According to the textbook, the linear program method takes $O(n^3)$ while the Auction Algorithm $O(mn^3)$ with worse case complexity of

$$\frac{max_{(i,j)}v(i,j) - min_{(i,j)}v(i,j)}{\epsilon}$$

where $\epsilon$ remains the same. The results are as expected.

2. **What problem?**

This is the LP solving of sorting a list of given numbers. $x_{ij}$ is 1 if the $i$th number is placed at $j$th position, and 0 otherwise.

The constraints indicate that it is a one-to-one matching problem. Additionally, index $j$ in the expression of objective value shows that element positions affect results. It reaches its maximum if and only if we place all numbers in an increasing order by exchange argument.

3. **Different objectives in stable matching**

The linear program is to minimize

$$\sum_{\forall i,j} \frac{x_{ij} * (A_{ij} + B_{ji})}{2n}$$

subject to the constraints

$$\sum_i x_{ij} = 1, \forall j$$
$$\sum_j x_{ij} = 1, \forall i$$
$$x_{ij} \geq 0, \forall i, j$$

where $i$ and $j$ range from 1 to $n$, the total number of matching pairs. $A$ and $B$ are two $n * n$ matrices that encode the preference values of men and women, respectively.

4. **Manipulation through permutation**

The Gale-Shapley results are:

- M1 matches W1
- M2 matches W2

- M3 matches W3

First, Man 1 proposes to Woman 1, who accepts because she is single. Then Man 2 proposes to Woman 2, who is single and therefore accepts. After that, Man 3 proposes to Woman 1 and gets rejected because her current match is more desirable. Then Man 3 proposes to Woman 2, rejected as well. Eventually he proposes to Woman 3 and is accepted.

Woman 2 can misrepresent her preferences and end up with a more preferred partner by reporting M1 > M3 > M2. First, Man 1 and Woman 1, Man 2 and Woman 2 are matched. Man 3 proposes to Woman 1, gets rejected, and then proposes to Woman 2. According to her fake preference, Man 3 ranks higher than Man 2, so Man 2 becomes single. Then Man 2 proposes to Woman 1, who abandons Man 1. Man 1 then proposes to Woman 2 and gets accepted. In Woman 2's true preference, Man 1 ranks higher than Man 3.

5. **Invariance of those matched**

Extending the notion of stable matching to be "no blocking pairs" to the new problem, then everyone marries only those that are before his/her "being single" preference. A man only proposes to women better than the choice of being single, and a woman handles proposals the same way. Then we can show that the resulted matching is stable when algorithm terminates using contradiction.

Suppose the result $S$ is not stable, then there must be at least one person that has one of the cases below in another matching $S'$ with distinct set of unmatched people:

- Case 1: A man $m$ who marries to $w$ in $S$ turns single in $S'$.

  Because in some matching $m$ is able to marry $w$, achievable $w$ ranks $m$ higher than "being single." Assuming that other pairs stays the same, $w$ will accept $m$'s proposal if she is single, so $w$ must have married to another man $m'$ at the end. This leads to the contradiction just as in the original stable marriage problem.
- Case 2: A woman $w$ who marries to $m$ in $S$ turns single in $S'$.

  Because in some matching $w$ marries $m$, $m$ ranks $w$ higher than "being single." Assuming that other pairs stays the same, $m$ would propose to $w$. Then $w$ accept $m$ until another man $m'$ comes, if any. In any circumstance $w$ does not stay single in $S'$, which contradicts our assumption.