1 P1 Solutions

1.1 (i) Defining SMP as a Computational Problem

Input:

- Two sets of participants (often referred to as *men* and *women*, though in practice they could be any two disjoint sets), each of size n.
- Each participant in the first set provides a *strict preference ordering* over all participants in the second set.
- Each participant in the second set provides a *strict preference ordering* over all participants in the first set.

Output:
A *stable matching*, which is a one-to-one pairing between the two sets such that **no blocking pair** exists.

A **blocking pair** is a man m and a woman w who are not paired with each other but who **both** prefer each other to their current partners in the proposed matching. If such a pair exists, the matching is **not stable**.

1.2 (ii) Example of the Stable Marriage Problem

Consider a small instance with $n = 2$. We have two men (M_1 and M_2) and two women (W_1 and W_2). Their preference lists are as follows:

Men's Preferences

- M_1: prefers W_1 over W_2.
- M_2: prefers W_1 over W_2.

Women's Preferences

- W_1: prefers M_1 over M_2.
- W_2: prefers M_2 over M_1.

We want to find a stable matching. Let's check possible matchings:

Matching 1: (M_1, W_1) and (M_2, W_2). No blocking pair exists, so **Matching 1** is stable.

Matching 2: (M_1, W_2) and (M_2, W_1). Since M_1 and W_1 prefer each other to their current partners, they form a blocking pair, making this matching unstable.

Thus, the only stable matching is:

$$(M_1, W_1) \quad \text{and} \quad (M_2, W_2).$$

Algorithm 1 Gale-Shapley Algorithm

Input: Two sets, Men and Women, each with strict preference lists.
Output: A stable matching.
for each man m in Men **do**
  m is free and has not proposed to anyone yet
end for
for each woman w in Women **do**
  w is free
end for
while there exists a free man m who has not proposed to every woman **do**
  Let w be the highest-ranked woman in m's preference list to whom he has not yet proposed
  if w is free **then**
    (m, w) become engaged
  else
    Let m' be the man currently engaged to w
    if w prefers m' to m **then**
      m remains free
    else
      m' becomes free
      (m, w) become engaged
    end if
  end if
end while
return the set of engaged pairs

2 P2 Solutions

2.1 (i) Gale–Shapley Algorithm in Pseudocode

2.2 (ii) Time Complexity Analysis

In the worst case, each man could propose to every woman, leading to at most $O(n^2)$ proposals. Checking preferences takes $O(1)$ time with preprocessing. Thus, the overall complexity of the Gale–Shapley algorithm is $O(n^2)$.

3 P3 Solutions

3.1 Gale–Shapley Algorithm in Python

```
def gale_shapley(men_preferences, women_preferences):
    rank_of_man = {w: {m: i for i, m in enumerate(pref_list)}
                   for w, pref_list in women_preferences.items()}

    free_men = list(men_preferences.keys())
    next_proposal_index = {m: 0 for m in men_preferences}
    current_engagements = {}

    while free_men:
        m = free_men.pop(0)
        w = men_preferences[m][next_proposal_index[m]]
        next_proposal_index[m] += 1

        if w not in current_engagements:
            current_engagements[w] = m
        else:
            m_current = current_engagements[w]
            if rank_of_man[w][m] < rank_of_man[w][m_current]:
                current_engagements[w] = m
                free_men.append(m_current)
            else:
                free_men.append(m)

    return {m: w for w, m in current_engagements.items()}

if __name__ == "__main__":
    men_preferences = {'M1': ['W1', 'W2'], 'M2': ['W1', 'W2']}
    women_preferences = {'W1': ['M1', 'M2'], 'W2': ['M2', 'M1']}
    matching = gale_shapley(men_preferences, women_preferences)
    print("Stable Matching:", matching)
```

This confirms that the stable matching is (M_1, W_1) and (M_2, W_2).