Palmer, Deuce
Theory

The 5 conditions in formal logic

1. The set S must be a subset of A x B (cartesian product of A and B)

$$S \subset A \times B$$

2. All elements in set A have a corresponding element in set B such that the pair (a, b) is an element of S

$$\forall a \in A, \exists b \in B, (a,b) \in S$$

for all Elements a in A, there exists an Element b in B, where the pair (a,b) is an Element of S (stable match).

3. All elements in set B have a corresponding element in set A such that the pair (a, b) is an element of S

$$\forall b \in B, \exists a \in A, (a,b) \in S$$

for all Elements b in B, there exists an Element a in A, where the pair (a,b) is a Element of S (stable match).

4. There are no two pairs (a, b) and (a', b') in S such that b' is higher preference for a than b, and a is higher preference for b' than a' (in this case, since a and b' both prefer each other to their assigned match, they will choose to break their match in favor of a match (a, b')) $\forall (a,b)(a',b') \in S, \sim(a.\text{prefersOver}(b',b) \wedge b'.\text{prefersOver}(a,a'))$

for all possible pairs (a,b) and (a',b') that are Elements of S (all stable matches), a does NOT prefer b' over b OR b' does NOT prefer a over a'.

5. There are no two pairs (a, b) and (a', b') in S such that a' is higher preference for b than a, and b is higher preference for a' than b' (again, in this case b and a' would break their assigned match in favor of a match (a', b)

$$\forall (a,b)(a',b') \in S, \sim(b.\text{prefersOver}(a',a) \wedge a'.\text{prefersOver}(b,b'))$$

for all possible pairs (a,b) and (a',b') that are Elements of S (all stable matches), b does NOT prefer a' over a OR a' does NOT prefer b over b'.

2 Proofs by Contradiction

1. If both sets contain n elements, The Gale-Shapley algorithm always results in n pairs.

if P then Q

$P \rightarrow Q$

$\sim(P \rightarrow Q) = P \wedge \sim Q$

$\sim(P \rightarrow Q)$ = Both sets contain n elements, and the Gale-Shapley algorithm results in less than n pairs

We have a contradiction

If both sets have n elements and the algorithm results in less than n pairs, then elements are unpared

We know that all elements will be paired based on conditions 2 & 3 showing us that each element in B has a corresponding match in A.

The algorithm implementation involves a loop that will run until each element in A and each element in B are no longer unpaired, therefore the algorithm must not be completed and will run until unpared elements are paired since conditions 2 & 3 state they will always have a match

$\sim(P \rightarrow Q) \rightarrow C$

$\therefore P \rightarrow Q$

2. The resulting pairs are stable; as in, there are no unstable pairs when the algorithm finishes.

~P = The resulting pairs are unstable; as in, there exists unstable pairs when the algorithm finishes.

We have a contradiction

If pairs are unstable, it means that condition 4 or 5 does not hold true.

If condition 4 or 5 are false, elements will voluntarily switch pairing.

If elements will voluntary switch, the algorithm is NOT completed and when it does complete the pairs will be stable

$\sim P \rightarrow C$

$\therefore P$

Pseudocode that verifies stability of matches

First, the last two conditions above will be verified across all possible combinations of pairs in matches. For each possible combination of pairs (a, b), (a', b'), it will be verified that they will not voluntarily switch. This will be done by two "if" statements. The first of which follows condition 4 and will make sure a doesn't prefer b' over b, and that b' doesn't prefer a over a'. This will be done with a "prefersOver" function which can be called for a single Element and will take 2 elements and return a boolean depending on whether or not the first is of higher preference than the second. This will be accomplished by looking through the Element's preference list to see which one appears first. The same will be done for condition 5 to verify that b doesn't prefer a' over a and that a' doesn't prefer b over b'. If the conditions are valid for all possible combinations of pairs, then the matches will be proven stable.

```
Bool StableMatchSet::matchesAreStable() {
        For loop (the length of matches) {
                For loop (the length of matches) {
                        If (Elements don't equal each other) { //creates all possible combinations
                without comparing the same Element to itself
                                If (a prefers b' over b && b' prefers a over a') {
                                        return false
                                }
                                If (b prefers a' over a && a' prefers b over b') {
                                return false
                                }
                        }
                }
        }
        return true
}
```