This handout is intended as a supplement to Monday’s lecture. Assume that we have \( n \) residents \( R_1, R_2, \ldots, R_n \) and \( n \) medical school residency positions \( M_1, M_2, \ldots, M_n \) where \( n \) can be any positive integer. Each \( M_i \) gives a preference list that gives the residents in an order giving \( M_i \)'s preferences. Each \( R_j \) must be in each list exactly once. Likewise, each \( R_j \) gives a preference list that gives the medical school positions in order of \( R_j \)'s preferences where each \( M_i \) is in each list exactly once.

Let a matching be an assignment between residents and medical schools so that each resident is matched with a single position and each position is matched with a single resident. A matching is unstable if there exists a resident \( R_j \) and a position \( M_i \) that are not paired to each other would both prefer each other to their current pairing. Our goal is to creating a matching that is stable. Here is a proposed method to achieve this goal. In this we assume that initially everyone is unmatched.

```
Make-Stable-Matching
1    Repeat
2    Let \( M_i \) be any unmatched medical school
3    \( M_i \) proposes to first \( R_j \) on its preference list
4    If \( R_j \) is unmatched
5        Match \( M_i \) and \( R_j \)
6        Remove from \( R_j \)'s preference list all \( M \)'s ranked lower than \( M_i \)
7    Else if \( R_j \) is matched to \( M_k \) where \( M_i \) is preferred to \( M_k \)
8        Match \( M_i \) and \( R_j \)
9        Remove from \( R_j \)'s preference list all \( M \)'s ranked lower than \( M_i \)
10   Now \( M_k \) is unmatched
11   Else (\( R_j \) is matched with \( M_k \) where \( M_k \) is preferred to \( M_i \))
12      Do nothing (so \( M_i \) remains unmatched)
13   \( M_i \) removes \( R_j \) from its preference list
14   Until every \( M_i \) is matched
```

Our proof that this algorithm is correct uses the following three lemmas about operation of the Make-Stable-Matching.

**Lemma 1** If \( M_i \)'s prefers \( R_\ell \) to \( R_j \) (i.e. \( R_\ell \) is before \( R_j \) in \( M_i \)'s preference list) then \( M_i \) will propose to \( R_\ell \) prior to proposing to \( R_j \).

**Proof:** This follows directly from line 3 since \( M_i \) always proposes to the first resident on its preference list. \( \square \)
Lemma 2 If $R_j$ is unmatched then it accepts any proposal made.

Proof: From the condition on line 4, it immediately follows that an unmatched resident accepts any proposal.

Lemma 3 If $R_j$ is matched to $M_k$, then $R_j$ accepts a proposal from $M_i$ if and only if $R_j$ prefers $M_i$ over $M_k$.

Proof: We first argue that if $R_j$ accepts a proposal from $M_i$ (breaking the match with $M_k$) then $R_j$ prefers $M_i$ to $M_k$. We use an indirect proof here. Namely, we argue that if $R_j$ does not prefer $M_i$ over $M_k$ then then $R_j$ does not accept a proposal from $M_i$. This follows directly from the condition on Line 11.

Next we argue that if $R_j$ prefers $M_i$ over $M_k$ then then $R_j$ will accept a proposal from $M_i$. This follows directly from the condition on line 7.

We now prove that our algorithm, Make-Stable-Matching satisfies the following two conditions: (1) it will terminate with a matching (i.e. each $M$ and $R$ is paired) and (2) the matching is stable.

Theorem 1 The algorithm Make-Stable-Matching will terminate with a matching.

Proof: The key observation used to argue that Make-Stable-Matching terminates is that each time through the repeat loop at least one item is removed from someone’s preference list. We now use a proof by contradiction to argue that a medical school position cannot reach the end of its preference list without being paired. Suppose not. Let $M_i$ be a medical school position that is not paired after reaching the end of its preference list. Since each resident is paired with a single medical school position and the number of medical schools positions is the same as the number of residents, some resident, say $R_j$, must be unpaired. By Lemma 2, $R_j$ has not received any proposals. This contradicts that $M_i$ has reached the end of its list since its list included $R_j$. Thus, each medical school position (and hence each resident) will be paired.

Theorem 2 The matching output by Make-Stable-Matching is stable.

Proof: We use a proof by contradiction. Suppose the matching output is not stable. Then we have pairings $M_i \rightarrow R_j$ and $M_k \rightarrow R_\ell$ where $M_i$ and $R_\ell$ would both prefer each other to their current pairs. That is, the original preference lists for $M_i$ and $R_\ell$ have the following general structure:

$M_i : \cdots R_\ell \cdots R_j \cdots$

$R_\ell : \cdots M_i \cdots M_k \cdots$

For $M_i$ and $R_j$ to be paired by Make-Stable-Matching, $M_i$ must have proposed to $R_j$. Hence, by Lemma 1, at some earlier point, $M_i$ proposed to $R_\ell$. Suppose that $R_\ell$ rejects the proposal from $M_i$. Then, by Lemma 3, $R_\ell$ is paired with a medical school position it prefers to $M_i$ and hence prefers to $M_k$. And if $R_\ell$ accepts the proposal from $M_i$, then $R_\ell$ is paired with a medical school position, namely $M_i$, that it prefers to $M_k$. Since, in both cases, $R_\ell$ is paired with a medical school position it prefers over $M_k$, it follows from Lemma 3 that $R_\ell$ will reject a later proposal from $M_k$ contradicting that $R_\ell$ and $M_k$ are paired in the matching that was output. Hence, a non-stable matching could not occur.