CS577
Homework #7, Problem #4
Sreeram Danda

#4 The Algorithm

```
partner_pairing(list L, list R)
  |   for(all p  in L and R)
  |     |   p.partner = free
  |     |   if (p.group == L)
  |     |     |   Q.add(p)
  |   while(Q is not empty)
  |     |   let current = Q.pop()
  |     |   for(i = 0, i < current.preferences.length, i++)
  |     |     |   let possible_partner = current.preferences[i];
  |     |     |   if(possible_partner.partner == free)
  |     |     |     |   current.partner = possible_partner
  |     |     |     |   break;
  |     |     |   else if(possible_partner.partner != free)
  |     |     |     |   if(possible_partner.prefrences[current] <
  |     |     |     |   possible_partner.prefrences[possible_partner.partner])
  |     |     |     |     |   possible_partner.partner.partner = free;
  |     |     |     |     |   Q.add(possible_partner.partner);
  |     |     |     |     |   possible_partner.partner = current;
  |     |     |     |     |   current.partner = possible_partner;
```

#4 Prove |Q| is Non-Increasing

There are three cases that need to be considered when x gets tentatively paired up.

#1: x partner is free prior to being partnered with x. In this case, we consider x paired and remove x from the queue.

#2: x partner y is not free but partnered with someone x' who they do not prefer over x. In this case, the x' and y are broken up and x is partnered to y. This means that x' is once again free and must be added back to the queue, however, since x is partnered; they are removed from Q. Thus, the size of Q does not change.

#3: x's partner y is not free and y preferes x' over x. In this case, x continues attempting to pair with others in their list. Thus, the size of Q remains unchanged until x gets paired.

#4 Prove Algorithm Equivalence

Remembering who is broken up does not change the algorithm. This is because for each $x \in L$ a potential pattern must have their preferences checked as well. If x has been previously broken up with said y, then it must be the case that y prefers x' over x and thus they will never be paired in either algorithm.

#4 Prove Termination

For each x that is considered they have n possible partners. For each partner under each iteration of the for loop the number of possible partners decrease by one if x is not partnered. This means if x is

continually not partnered, the number of possible partners decreases to 1. At this point, it is not possible for both x and the possible partner to prefer someone else. This is because all the people must be partnered and if all previous x are perfectly matched, then the possible partner cannot fall into a condition where both people prefer others over each other as if x prefers someone else they must have offered to partner prior to partnering with the current possible partner. This means x gets assigned to this current possible partner. This repeats until all x are assigned to a partner. Since, the subproblem terminates for each x, the algorithm must terminate as it is just *n* iterations of the subproblem for each x.

#4 Prove Time Bound

The runtime complexity of this algorithm is $O(n^2)$. Each x partner has n possible partners and there are n x's. This means in the worst-case scenario each x must attempt to partner with all n possible partners. Thus, there are n^2 attempts to partner in the worst-case.

#4 Prove Correctness

When the algorithm terminates, there cannot be a $x \in L$ and a $y \in R$ that are both not partnered. This statement is true because that x must have offered to partner the y at some point during the matching. If that partnership was broken then y must have been matched with some x'. Now in terms of preference, it is not possible for an x and a y to prefer each other over their current partners at the end of the algorithm. If x prefers y, then there must have been a prior time where x offered to partner with y. At that point, y could have accepted or rejected. If y accepted and x and y are not partnered at the end of the algorithm, then x and y must have broken up as y gets offered to partner with someone they prefer more than x, meaning y doesn't prefer x over their current partner. Similarly, if y did not get partnered with x from the start, then y must have been partnered with some x' that they preferred over x. Thus there cannot be $x \in L$ and a $y \in R$ that are partnered and both prefer an alternative x' and y'.