1. Consider matchings $M$ and $N$ in $G$ with $|M| > |N|$. We construct matching $M'$ and $N'$ in $G$ such that $|M'| = |M| - 1$ and $|N'| = N + 1$. All edges that are in both $M$ and $N$ we put in both $M'$ and $N'$. Now, if there exists an edge in $M$ such that none of its endpoints are endpoints on an edge in $N$, we move that edge to $N'$, while all the remaining edges that were in $M$ go to $M'$ and the remaining edges in $N$ go to $N'$. The two new matchings clearly satisfy the desired property.

However, if there does not exist an edge in $M$ such that none of its endpoints are endpoints of an edge in $N$, it must be the case that every edge in $M$ shares vertices with an edge in $N$. So besides all the edges that are in both $M$ and $N$, we have "clusters" of edges in $M$ and $N$ that look like the ones on page 8 of Lecture 12. Since $|M| > |N|$, there will be such a cluster that has more edges in $M$ than edges in $N$ (so in the first picture on page 8, the blue edges are the ones in $M$). We can then switch the edges in $M$ with the ones in $N$, i.e. we put the edges of $M$ in this cluster in $N'$ and the edges of $N$ in this cluster in $M'$. If we just move all other edges in $M$ to $M'$ and all other edges in $N$ to $N'$, the new matching $M'$ and $N'$ will have the desired property.

2. Biscuit applies the algorithm from class to find a maximum matching. Since no perfect matching exists, there exists at least one vertex that is not adjacent to an edge in the perfect matching. Now, Biscuit starts the game from that vertex. Adam will have to choose an edge that is not in the matching. But then Biscuit can choose an edge in the matching otherwise an augmenting path would exist. Then Adam has to choose an edge in the matching, while on the next turn Biscuit can again choose an edge in the matching since otherwise an augmenting path would exist. This continues, and on each turn Adam will have to choose an edge not in the matching while Biscuit can choose an edge in the matching. The game has to end on Adam’s turn since otherwise an augmenting path would exist.

3. If the size of the maximum flow is not a multiple of five, we will show that $e$ is saturated in every maximum flow of the network. Indeed, by the max flow-min cut theorem we have that the capacity of a minimum cut is also not divisible by 5. So the minimum cut must contain edge $e$. But by Claim 1 on page 15 of Lecture 13, given a maximum flow $f$ we must have that an edge $e$ in the minimum cut must be saturated in the maximum flow $f$.

4. We replace each edge with two antiparallel edges and assign unit capacity to every edge. We saw in class that in any flow network there exists a maximum flow $f$ in which for each pair of antiparallel edges $e$ and $e'$, either the flow through $e$ is zero, or the flow through $e'$ is zero. We can then run the Ford-Fulkerson algorithm for this modified directed graph. The maximum flow is equal to the maximum number of edge disjoint paths.

5. (a) We can modify the graph into an undirected graph as in problem 4. Just like in class, the maximum number of edge-disjoint paths equals the min number of edges needed to disconnect $s$ from $t$. Moreover, from the proof on page 12 on Lecture 14, the min cut is a set of disconnecting edges. We learned on pages 14-15 of Lecture 13 how to construct a min cut from the residual network of a maximum flow.

(b) We saw in HW 3 that there is a MST that contains $e$ iff $e$ is not the unique heaviest in any cycle that contains it. Then we can apply the same algorithm as in class, however, this time we ignore edges $e' \in E$ with $w(e') \geq w(e)$ (see page 14 of Lecture 15).