Pledge: *I pledge my honor that I have abided by the Stevens Honor System.* -Eric Altenburg

---

1: . . . Use these ideas to write down a complete proof of Zermelo's theorem. You're also welcome to use *other* ideas (Zermelo's theorem can be proved in more than one way.)

Proof. A finite game with two players can be thought of as a finite tree endowed with a root (which corresponds to the starting position) and a set of terminal nodes each of which is labeled as a win for Alice (Player 1) or a win for Bob (Player 2). At every vertex in this tree, this naturally determines a subtree which can be thought of as a game in its own right. From this idea, we can work backwards from each terminal node labeling each vertex x with either *Alice* or *Bob* depending on whether Alice or Bob has a winning strategy for the game that corresponds to the subtree at x, and depending on whose turn it is to play. To begin this process, we note that each terminal node is labeled *Alice* or *Bob* to signify who won the game. The following steps can be taken to label each vertex in the tree with a player's name.

1. Starting from the root, progress down the tree until you find a vertex x that does not have label on it but has each of its children labeled with *Alice* or *Bob*.
2. Depending on whose turn it is at x, we will check to see if x has any children with the label of the said player.
3. If there is a child of x that has a label of the said player, then we can label x with the player's name because that means the player can force a win at that position by moving to a child node with their name. If not, then x will be labeled with the opponent's name because no matter what move they make, they will move to a child node labeled with the opponent's name which means only the opponent can force a win at that position.

Steps 1-3 are then repeated until every vertex—including the root—in the tree has a label of *Alice* or *Bob* on it. Now when the game is played, starting from the root of the tree if the vertex is labeled *Bob*, then Bob would have a winning strategy for the game. This is because on Bob's turn, he can always move to a child node that has the label of *Bob* regardless of how Alice decides to play her turn. Because of the way each vertex is labeled, Bob will always have one or more child nodes that he can move to, and if he keeps doing this, he will reach a terminal node labeled *Bob* which would signify a win for him. The same follows for Alice if it is her move and the current vertex is labeled after her. Generally speaking, if the current vertex has the same label as the player's name, then they have a winning strategy at that move. □

2: Prove that if a game is allowed to have draws, then one player has a winning strategy or both players have drawing strategies.

Proof. Continuing with the same idea of representing a finite two player game as a finite tree with a root as the starting position, and a set of terminal nodes each of which is labeled as a win for Alice (Player 1), win for Bob (Player 2), or *Draw* if the game ends in a draw. At every vertex in this tree, this naturally determines a subtree which can be thought of as a game in its own right. From this idea, we can work backwards from each terminal node labeling each vertex x *Alice*,

Bob, or *Draw* depending on whether a player has a winning strategy or can force a draw for the game that corresponds to the subtree at x, and depending on whose turn it is to play. To begin this process, we note that each terminal node is labeled *Alice*, *Bob*, or *Draw* to signify how the game ended. We can modify the previous procedure to label each vertex as follows.

1. Starting from the root, progress down the tree until you find a vertex x that does not have a label on it but has each of its children labeled with *Alice*, *Bob*, or *Draw*.
2. Depending on whose turn it is at x, we will check to see if x has any children with the label of the said player or *Draw*.
3. If there is a child of x that has a label of the said player, then we can label x with the player's name because that means the player can force a win at that position by moving to a child node with their name. If none of the children are labeled with the player's name but at least one is labeled *Draw*, then we can label x with *Draw* since the player does not have a winning move to make, so the next best outcome for them would be to force a draw in which no one would win. If none of the children are labeled with the player's name or *Draw*, then x will be labeled with the opponent's name because no matter what move the player makes, they will move to a child node labeled with the opponent's name which means only the opponent can force a win at that position.

Steps 1-3 are then repeated until every vertex—including the root—in the tree has a label of *Alice*, *Bob*, or *Draw* on it. Now when the game is played, starting from the root of the tree if the vertex is labeled *Bob*, then Bob would have a winning strategy for the game. This is because on Bob's turn, he can always move to a child node that has the label of *Bob* regardless of how Alice decided to play her turn. Because of the way each vertex is labeled, Bob will always have one or more child nodes that he can move to, and if he keeps doing this, he will reach a terminal node labeled *Bob* which would signify a win for him. The same follows for Alice if it is her move and the current vertex is labeled after her. If the current vertex is labeled with *Draw*, then both Alice and Bob should move to a child node labeled *Draw* since none of those nodes will have a label with their name on it depending on whose turn it is. Moving to a child node that is not labeled *Draw* will give their opponent a winning strategy since they can only move to a vertex with their opponent's name as the label. Generally speaking, if the current vertex has the same label as the player's name, then they have a winning strategy at that move, but if it is labeled as *Draw*, then they will both have a drawing strategy. □

3: Consider a version on Nim that is just like regular Nim except the player who takes the last token *loses*. Describe a winning strategy for this version of Nim, and prove your answer.

Proof. It is important to note that if the game has the Nim positions initially balanced, the opponent will have the winning strategy so the player must wait until their opponent makes a mistake this way the player can balance the Nim positions. Additionally, as it was shown in class we know that on a balanced Nim position, there is only one possible move and it is to unbalance it. If the Nim position is already unbalanced, then there is a move that a player can make to balance it.

To begin the winning strategy for a player, let Alice (Player 1) make the first move to balance the Nim position. Since the current state of the game is balanced, Bob (Player 2) has no choice but

to unbalance it. Because it is unbalanced, Alice can then take the balancing move. This balancing from Alice and unbalancing from Bob will continue until all the heaps are of size 1. From here, Alice must make sure to leave an odd number of heaps with a size of 1 to force a win. A win can be forced from this position because we know that the first player to pick a token from the odd-number of heaps will choose the token from the last heap as well subsequently causing them to lose. Therefore, to win, a player must leave an odd number of heaps all of size 1 this way their opponent will take the first token guaranteeing they will also take the last token as well. □