Collaborators: Ian Ding, Isaac Sin

Question 1: Correctness of Alpha-Beta Pruning

Base Case: Let the base case be that s is a terminal state. This means that no pruning is done because there are no "deeper" branches of the tree. Since no pruning is ever done then $v' = v$ for this state. This means that if $\alpha \leq v \leq \beta$ then $v' = v$ is trivial. Otherwise if $v \leq \alpha$ and since $v' = v$ then $v \leq \alpha$. Furthermore if $v \geq \beta$ and since $v' = v$ then $v \geq \beta$.

Inductive Step: Let s be some non-terminal state. Assume that for all descendants of s down to the terminal states the statements hold. We will show that this implies that the statements hold for s as well.

We can start with the first statement. Lets assume that $\alpha \leq v \leq \beta$. When we run alpha beta pruning from s we will prune any branches of the tree which result in values $v' > \beta$ or $v' < \alpha$. However since minimax returns v it would have never chosen any of those branches in the first place. This means that at this point alpha-beta pruning will only choose between branches that minimax could have chosen. This means since the two algorithms follow the same logic at this point (either minimizing or maximizing choices) and all of the children have $v' = v$ (from our inductive hypothesis alpha-beta pruning will choose a branch such that $v' = v$ for s.

Next, lets assume that $v \leq \alpha$ and we will show that $v' \leq \alpha$. From our inductive hypothesis we know that for each child of s, where v_i is the value of alpha-beta pruning on child i, $v_i \leq \alpha$. This means at this point alpha-beta pruning can only choose from choices such that $v' \leq \alpha$, so whichever branch it chooses the algorithm will result in some value which is $v \leq \alpha$.

For the last statement we can apply the same logic as the previous example except in the opposite direction, by the symmetry of alpha-beta pruning we know that this statement is also true without loss of generality.

Question 2: CSP Reduction

Let X be all the variables of the CSP $X = \{X_1, X_2, X_3, ..., X_n\}$. And the domain for every variable X_i is D_i. Let C be a n-ary constraint that involves all the variables. Create a synthetic variable y. The domain of y, D_y is the cartesian product of all the domains of the variables in X. $D_y = D_1 \times D_2 \times ... \times D_n$. Each element of D_y is a tuple representing the combination of values that satisfy the n-ary constraint C. For each variable X_i create an binary constraint from that variable to y, this constraint ensures that if y takes a value $\{v_1, v_2, ..., v_n\}$ then X_i must take up the value of v_i. The set of binary constraints ensures that any assignment to X that satisfies that original n-ary constraint C is an element of D_y. And any valid assignment to y and the variables in X according to the binary constraints will satisfy C.