Sam Beckmann
Connect 4 Write up

Alpha-Beta in game-tree search:

The use of alpha-beta pruning in a minimax search will indirectly lead to better search results. At the same level of search, minimax with or without alpha-beta will produce the same result. However, using alpha-beta will allow this result to be achieved faster, cutting out the evaluations of certain boards. As a result of this time save, additional levels of depth can be calculated for the same computational cost, ensuring greater accuracy.

Designing an evaluations function:

For the evaluation function, I eventually decided on just iterating through the list of possible win configurations and evaluating each one based on how many black and white tokens were in it. A configuration with no tokens or both black and white tokens did not affect the evaluation. A configuration with 1 token was worth one point for the respective side, 2 tokens was worth 5 points, 3 tokens was worth 10 points, and 4 tokens was worth infinity. Positive points were awarded for black, negative points for white.

I considered adding in heuristics for future boards, such as if three token slots had the fourth slot available to place a token, but ultimately decided the computational time would be better spent going to more levels with a simpler evaluation function, meaning the boards would actually be evaluated, instead of merely being estimated by heuristics.

As for the values of 1, 5, 10, they were chosen rather arbitrarily. I considered running the program with different values against your implementation and see if I could find a configuration that fared better, but was unable to get your compiled project to work on my ANSI C environment.