**Anton Sax (askz6)**

**CS5400-SP2018 Game1 Grading Rubric**

**Action & state generation: 70.00%**

Your code doesn't examine pieces that are between the king and enemy pieces capable of moving more than one space, but this is invalid as the piece "in the middle" could potentially capture the opponent piece that could otherwise place the king in check. The current castling logic only requires a single space between the king and the final location to be empty, when all of these spaces should be empty (you should "and" these expressions together instead of using "or").

**Search Algorithm: 100.00%**

A little inefficient because you can continuously search over pieces that don't have moves, but otherwise fine.

**Programming Practice: 50.00%**

There is a very large amount of redundant code. Your code structure would benefit drastically from completely separating the logic that identified if the king is in check. I would advise specifically checking for the cases where pieces can check the king instead of generating all "engaged spaces." In addition to this, it would be far better to simulate each move and see if the king is in check after the move has been made.

**Code reliability: 100.00%**

Good.

**Additional feedback:**

Unless you're calculating engaged spaces for a heuristic later on, I would strongly advise replacing this functionality with a function that can identify if a single space on the board is under attack by the enemy team (is there a piece within the line of sight capable of attacking the current space or a knight that can attack the current space). This approach is simpler, more robust, and should improve your code structure quite a bit.

**Total = 75.00%(Action & state generation) + 5.00%(Search Algorithm) + 10.00%(Programming Practice) + 10.00%(Code reliability)**

**Total: 72.50%**