**Coding Environment:**  
OS: Windows 10 (64-bit)  
Editor: Visual Studio Code   
Compiler: MinGW-w64 g++ 11.2.0   
Standard: C++11 (-std=c++11)

**Commands:**# Opening phase (minimax)  
g++ -o bin/MiniMaxOpening src/MiniMaxOpening.cpp src/classboard.cpp -std=c++11

# Opening phase (improved static estimation)  
g++ -o bin/MiniMaxOpeningImproved src/MiniMaxOpeningImproved.cpp src/classboard.cpp -std=c++11

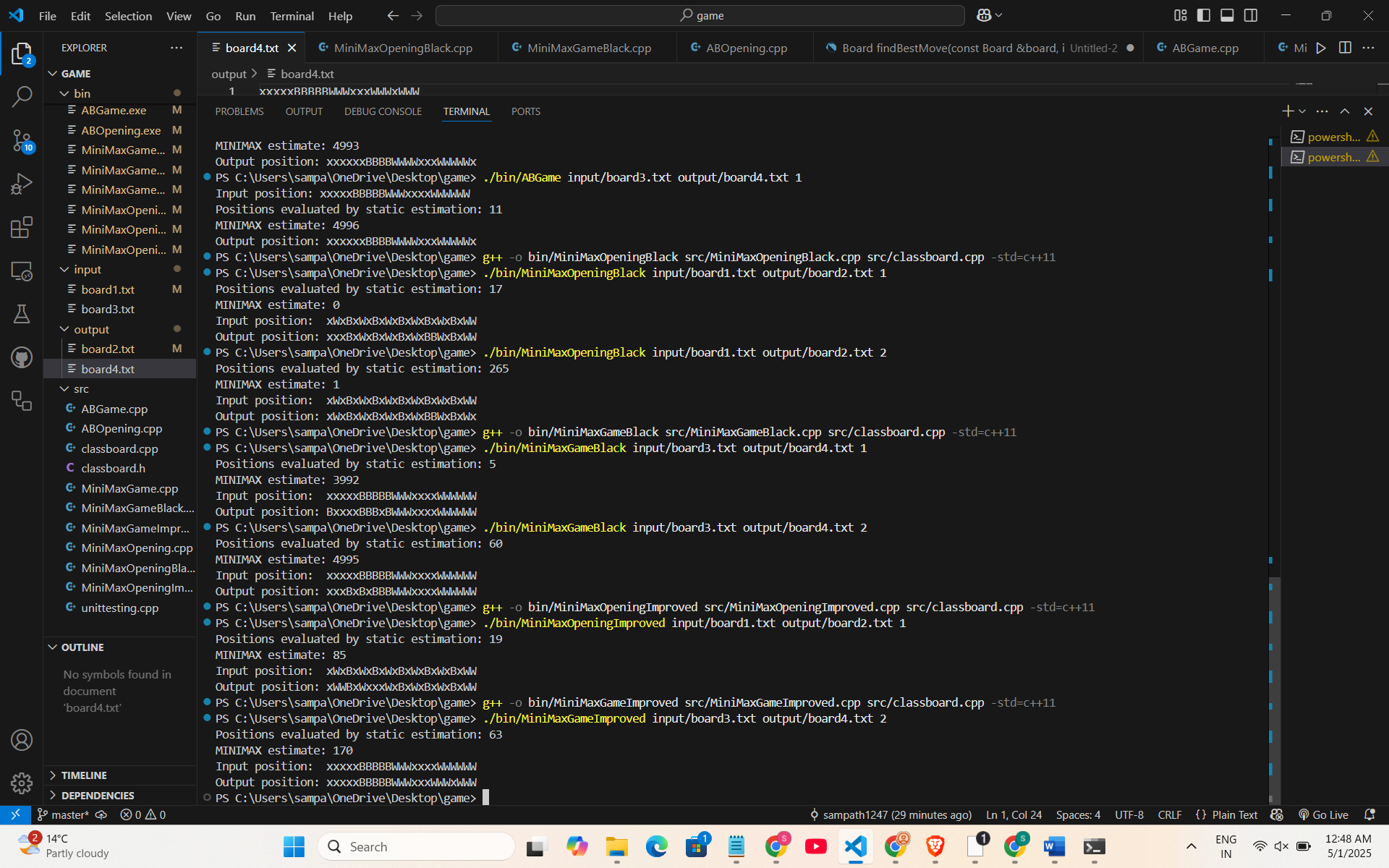
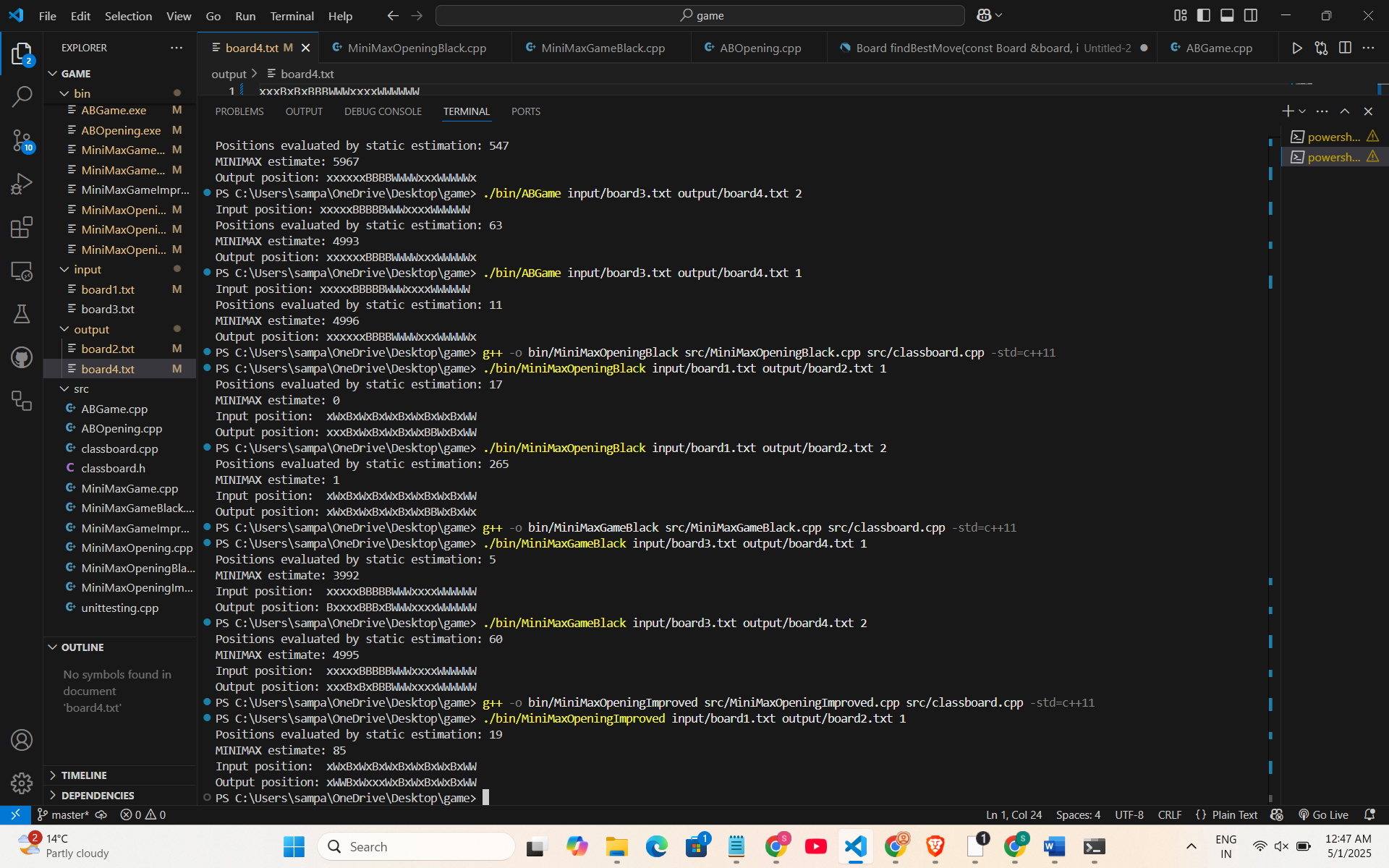
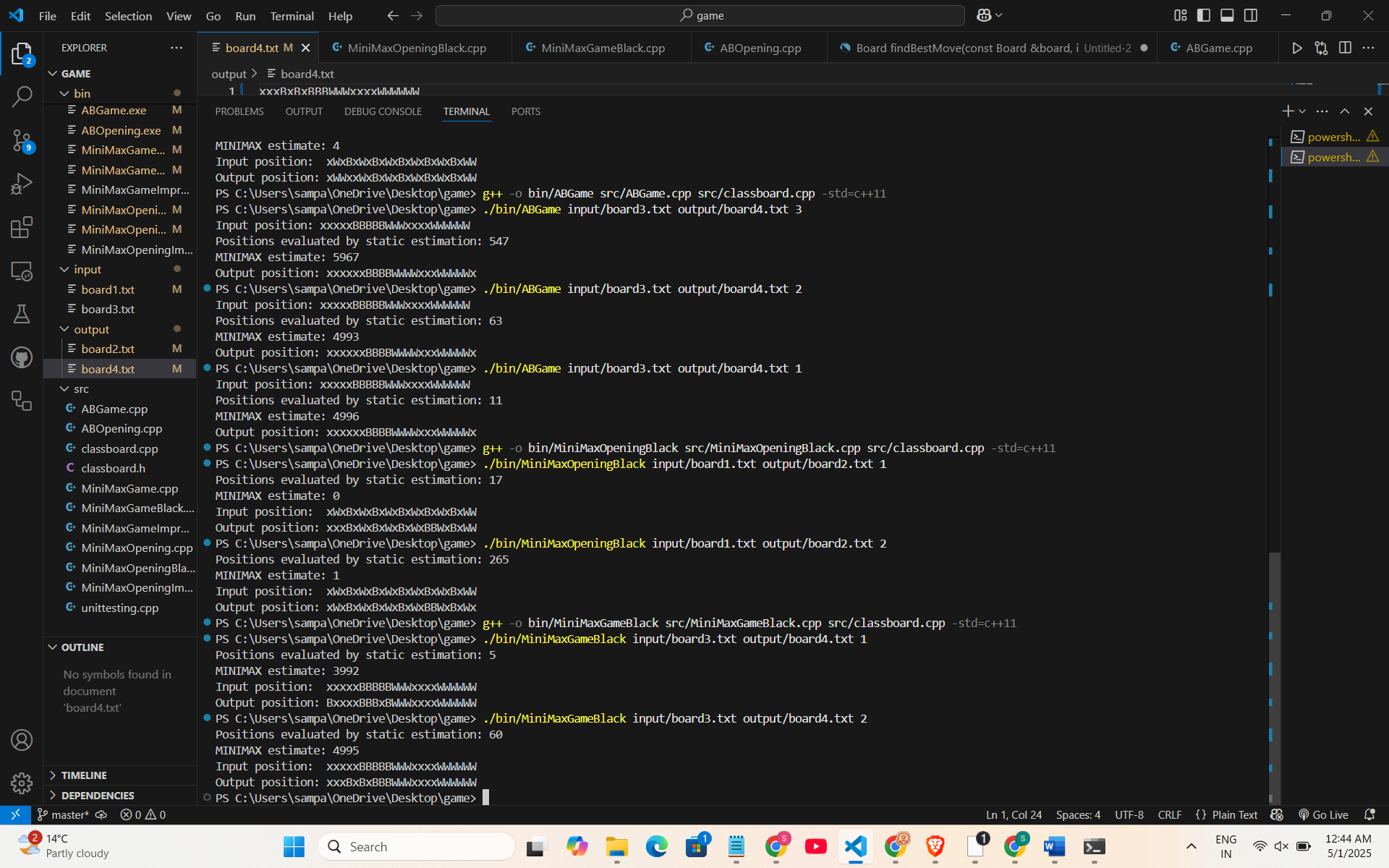
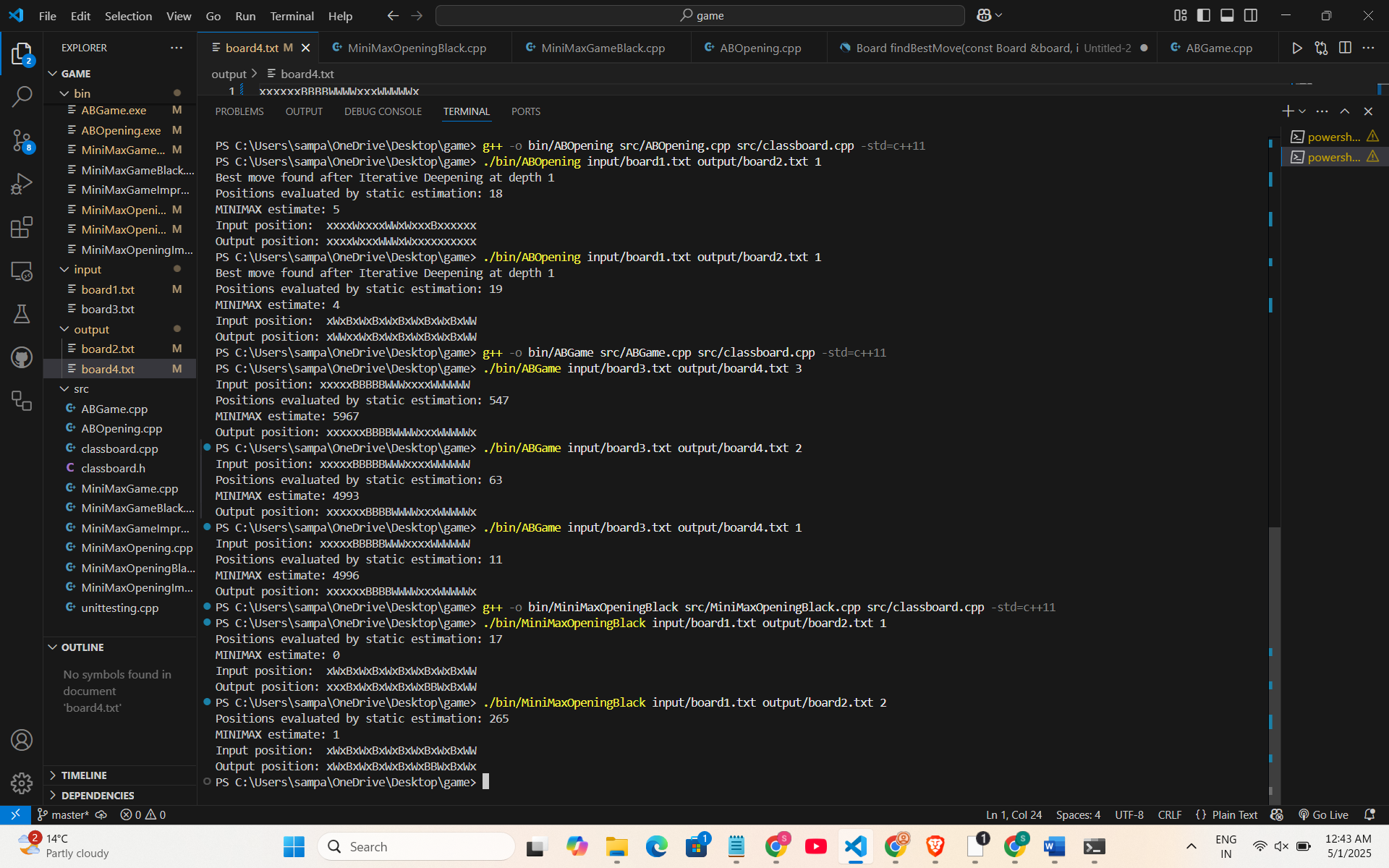
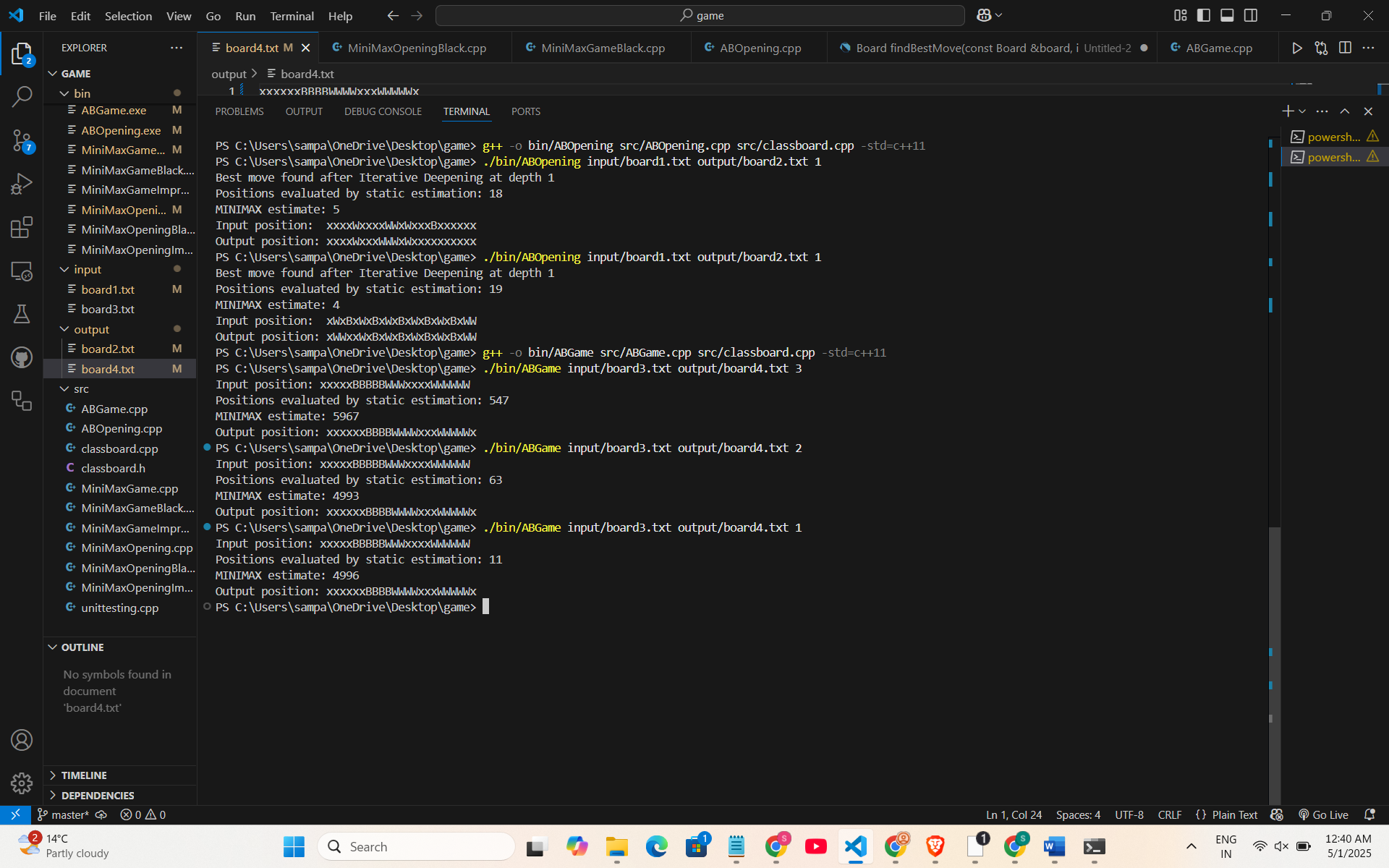
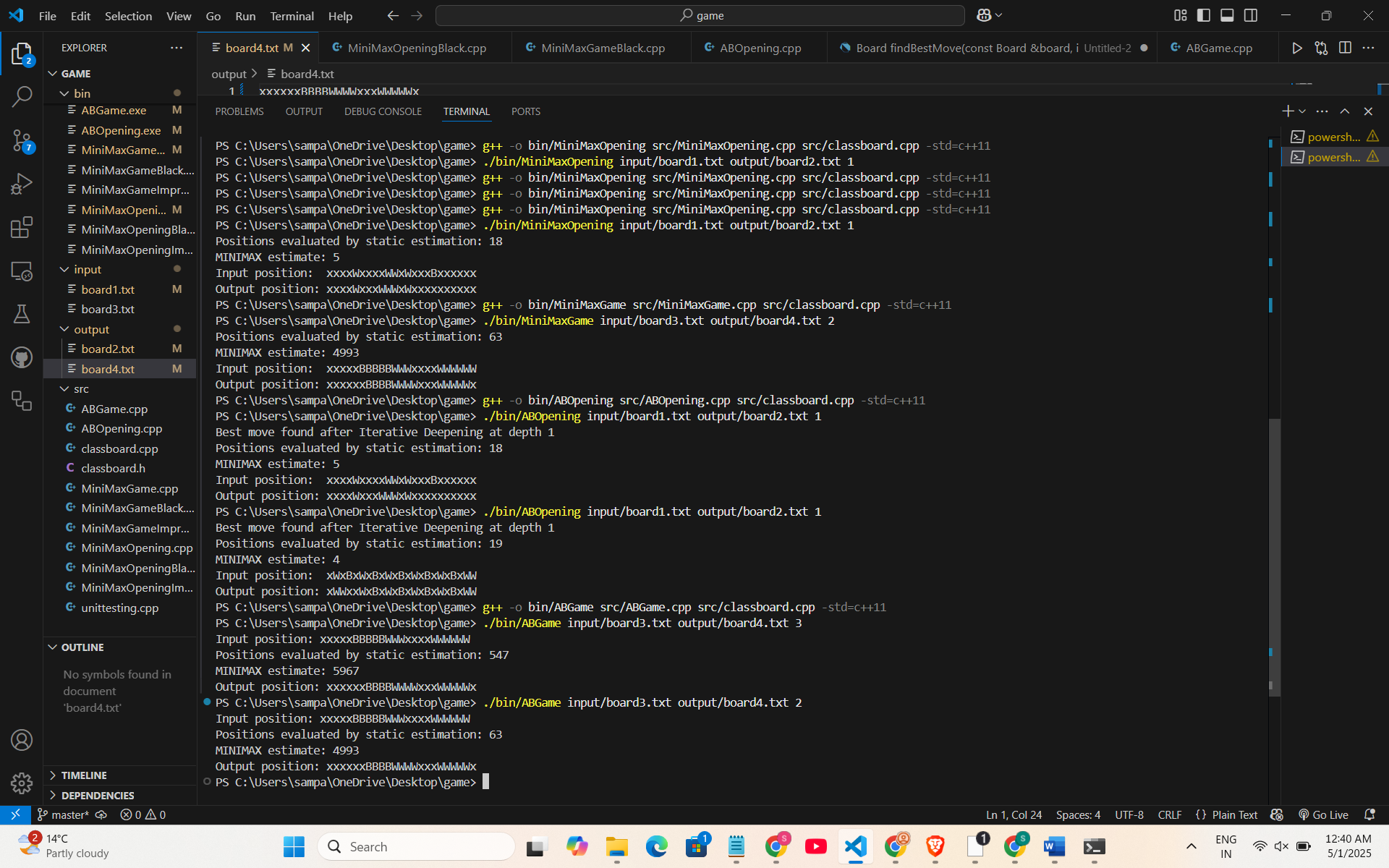
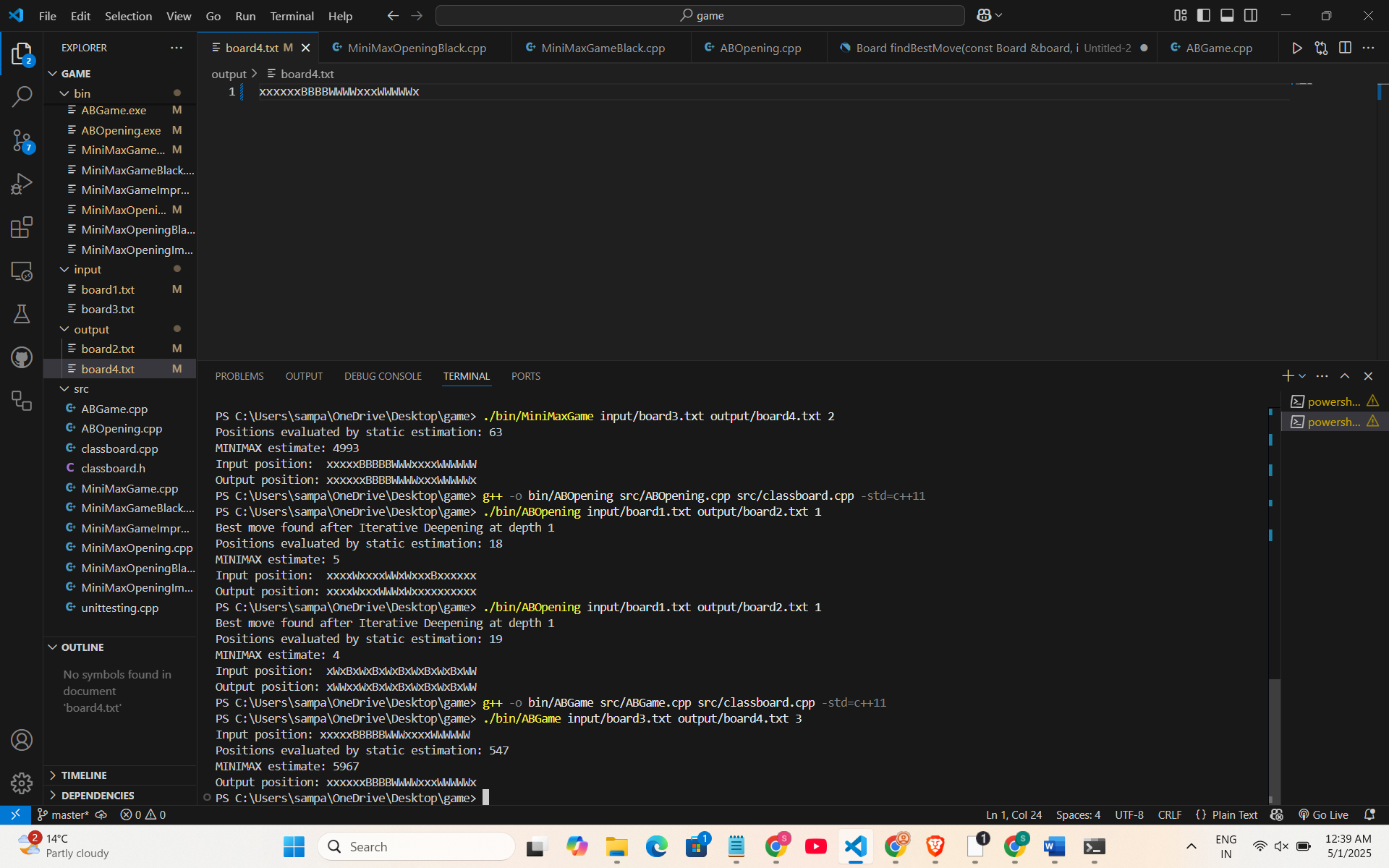
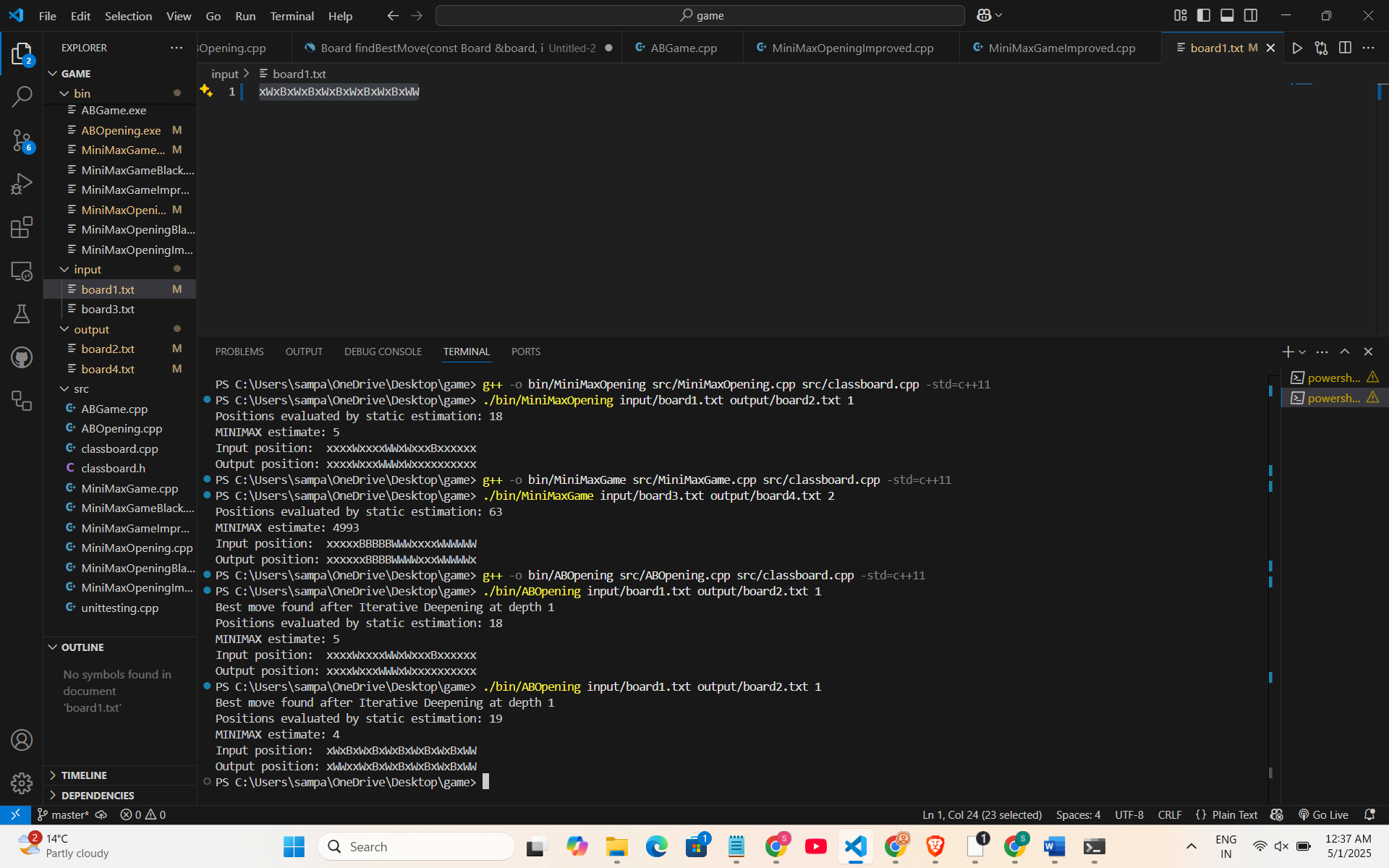
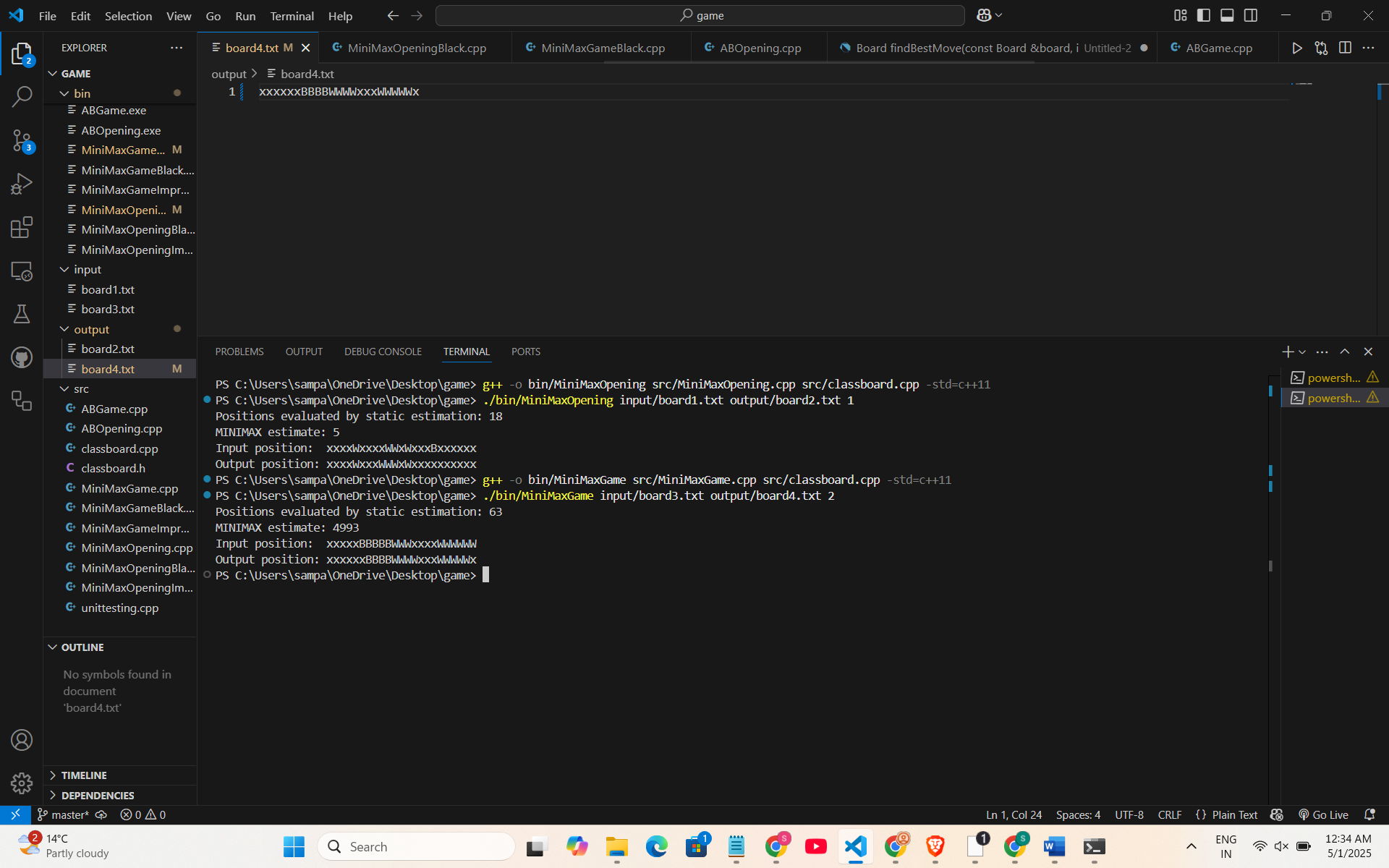
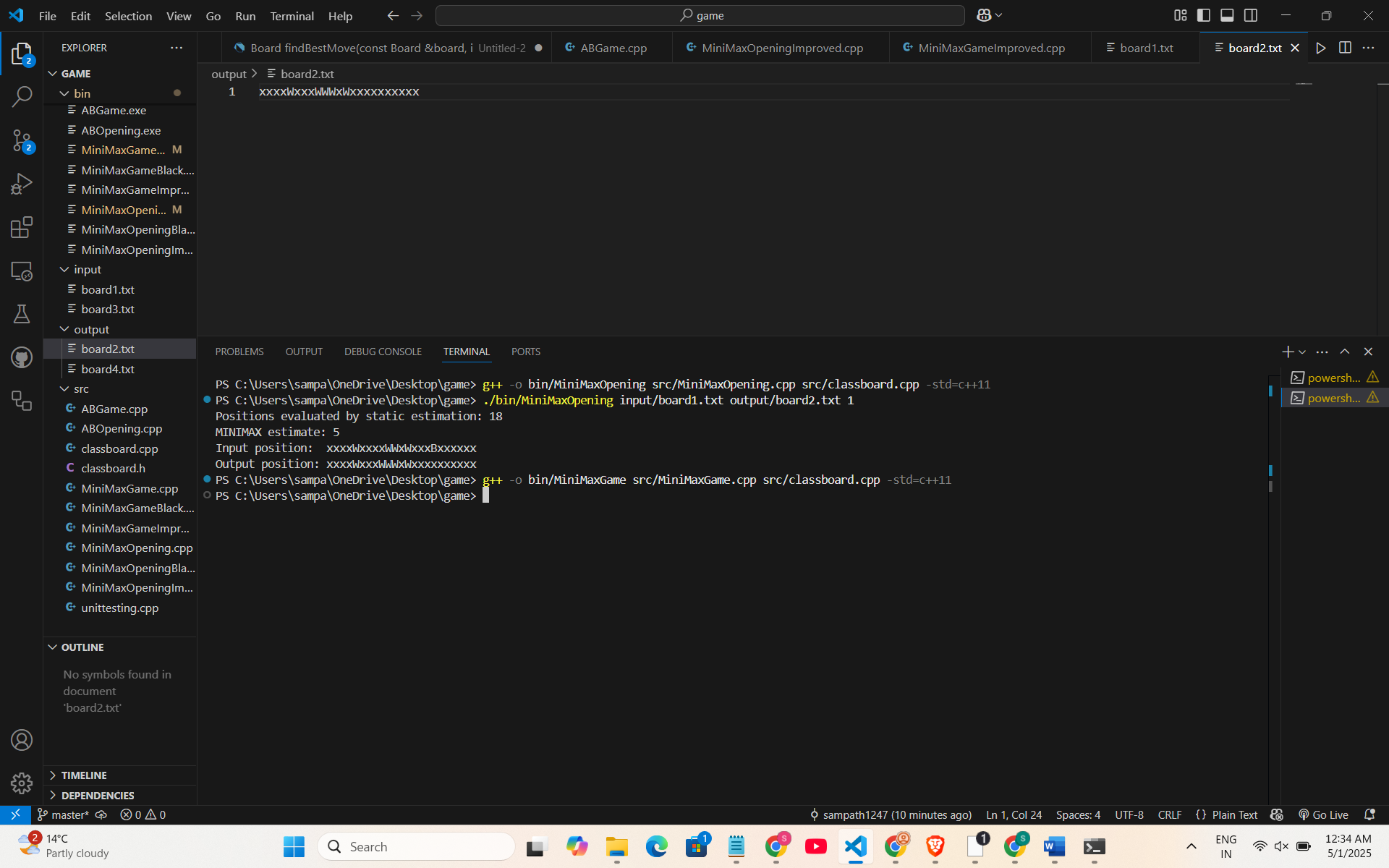
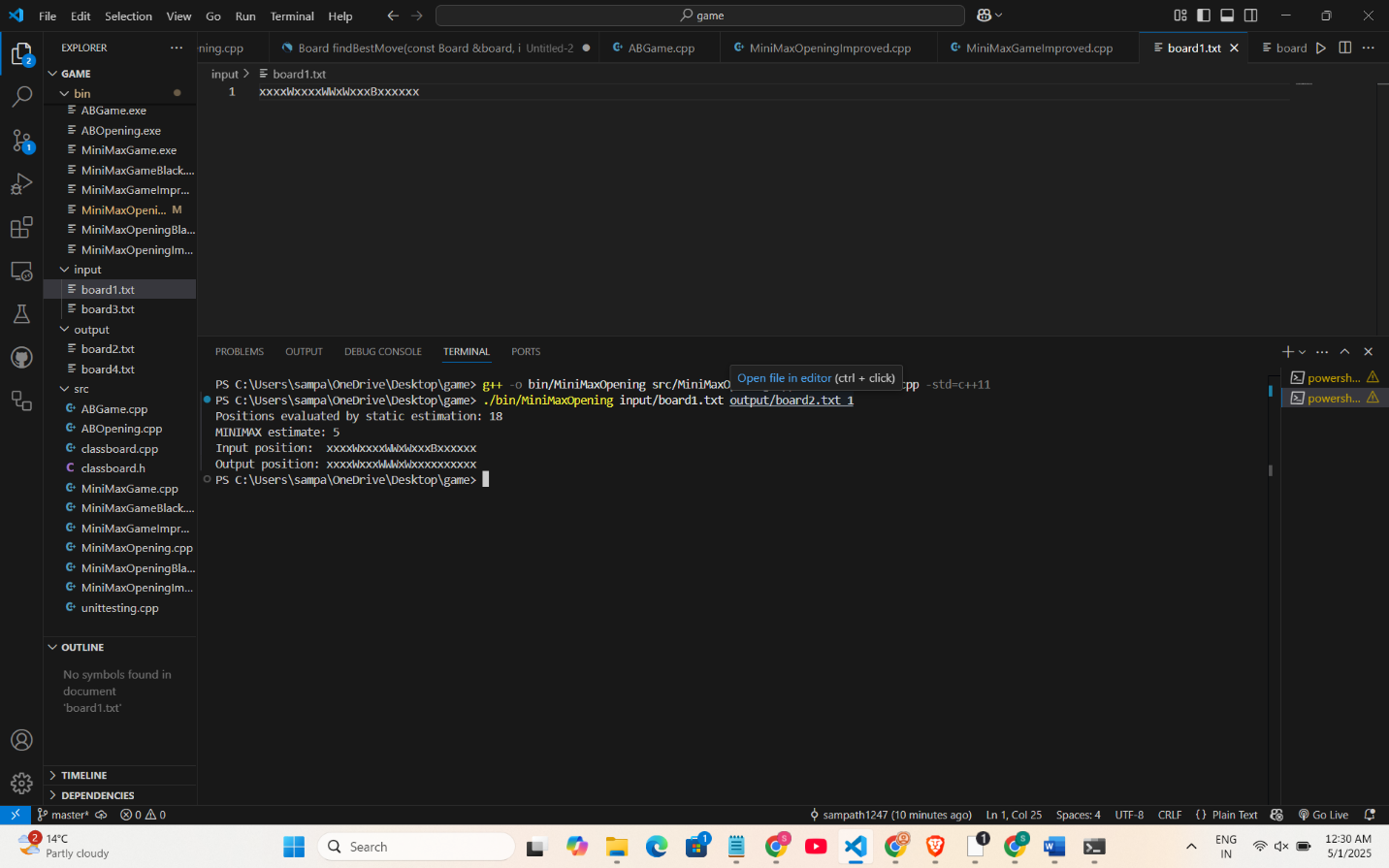
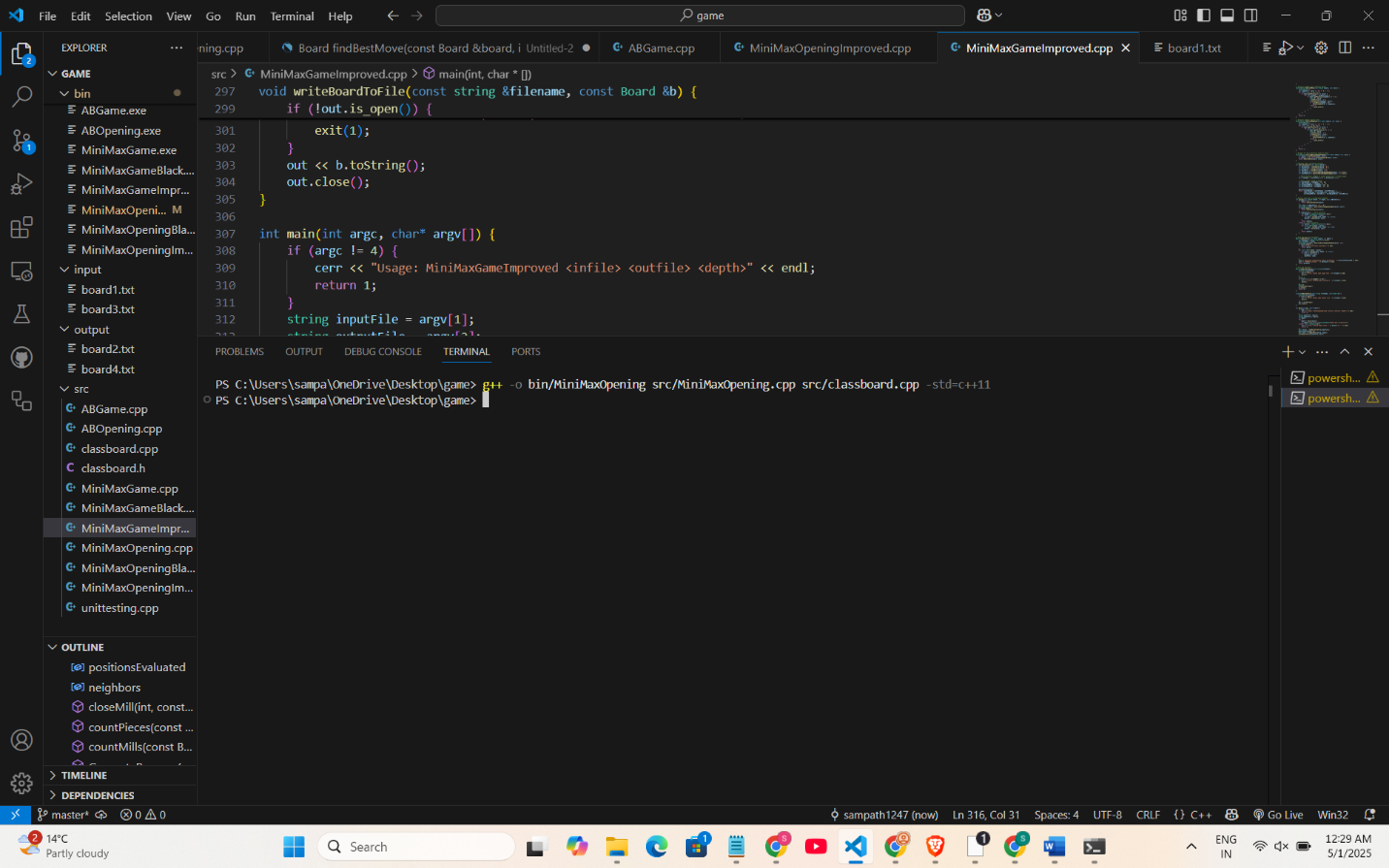
# Game phase (minimax)  
g++ -o bin/MiniMaxGame src/MiniMaxGame.cpp src/classboard.cpp -std=c++11

# Game phase (improved static estimation)  
g++ -o bin/MiniMaxGameImproved src/MiniMaxGameImproved.cpp src/classboard.cpp -std=c++11

# Alpha–Beta opening  
g++ -o bin/ABOpening src/ABOpening.cpp src/classboard.cpp -std=c++11

# Alpha–Beta game  
g++ -o bin/ABGame src/ABGame.cpp src/classboard.cpp -std=c++11

**Test Runs:**



**Explanation of the Improved Static Function:**

In the Nine Men's Morris game, the minimax algorithm searches the game tree up to a depth limit. At this point, it uses a static estimation function to assign a numerical score to the board state, indicating how favorable it is for the AI (White) versus the opponent (Black). A higher score favors the AI, while a lower score favors the opponent. The goal is to refine this function to capture strategic nuances beyond basic metrics.

The basic static estimation function typically calculates:

Piece count difference: (AI pieces - opponent pieces), reflecting material advantage.

Mobility difference: (AI legal moves - opponent legal moves), measuring flexibility.

For example:

evaluation = (whitePieces - blackPieces) + (whiteMoves - blackMoves)

This provides a starting point but overlooks key strategic elements like mill formation, which is central to Nine Men's Morris.

Add Game-Specific Heuristics

To improve the function, incorporate strategic factors specific to Nine Men's Morris:

Mill Formation: Add points (e.g., +15) for each mill White forms (three pieces in a row) and subtract points (e.g., -10) for each mill Black forms or threatens to form in the next move.

Blocking Opportunities: Add value (e.g., +5) if White blocks a potential Black mill.

Mobility: Reward White’s legal moves and penalize Black’s, but adjust its importance based on the game phase.

Piece Positioning: Optionally, give extra weight to pieces in key positions (e.g., corners or central spots), though this is less critical than mills.

These additions make the function more aligned with the game’s winning conditions—forming mills and restricting the opponent.

Weight the Heuristics

Assign weights to balance the factors:

evaluation = (10 × whitePieces) - (8 × blackPieces) +

(5 × whiteMoves) - (5 × blackMoves) +

(15 × whiteMills) - (10 × blackMillThreats)

Weights can be tuned through testing to prioritize mills (a high-impact factor) over piece count or mobility, reflecting their strategic value.

Phase-Aware Estimation

Nine Men's Morris has distinct phases—opening, midgame, and endgame—each requiring different priorities:

Opening/Midgame: Emphasize mobility and setting up potential mills (e.g., higher weight on moves).

Endgame (when a player has ≤3 pieces): Focus on piece count and existing mills (e.g., increase mill and piece weights).

Example logic:

if (blackPieces <= 3 || whitePieces <= 3) { // Endgame

evaluation = (20 × whitePieces) - (15 × blackPieces) + (20 × whiteMills) - (15 × blackMills);

} else { // Midgame

evaluation = (10 × whitePieces) - (8 × blackPieces) + (5 × whiteMoves) - (5 × blackMoves) +

(15 × whiteMills) - (10 × blackMillThreats);

}

This phase-aware approach ensures the AI adapts its strategy as the game progresses.

Normalize the Score

To ensure consistency across evaluations, normalize the score to a range like -100 to +100. For example, cap extreme values or scale the result. This step aids debugging and move comparison but can be optional for initial implementation.

The improved static estimation function enhances the basic version from the handout by moving beyond a simple piece count difference to include strategic heuristics tailored to Nine Men's Morris. By factoring in mill formation, mobility, and phase-specific weights, it better captures the game’s core mechanics—forming mills to remove opponent pieces and controlling the board. For instance, rewarding White’s mills and penalizing Black’s mill threats allows the AI to prioritize game-winning moves, while the phase-aware design ensures it adapts to the shifting importance of mobility and material as the game progresses. This results in smarter, more competitive decision-making compared to the handout’s basic approach, which lacks such depth and adaptability.