Lab Assignment: Implementing Alpha–Beta Pruning in Nim

Objective:

The goal of this lab is to implement the Alpha–Beta pruning algorithm for a two-player deterministic game (Nim). Apply adversarial search concepts to improve the efficiency of the minimax algorithm.

Game Description (Nim):

- The game starts with a certain number of heaps (piles) of stones.
- Two players take turns.
- On each turn, a player must:
  - Select one heap.
  - Remove at least one stone (up to all stones) from that heap.
- The player who makes the last move (takes the final stone) wins the game.

Requirements:

1. Represent the game state as a tuple of integers (e.g., (3, 4, 5) means 3 heaps with 3, 4, and 5 stones).
2. Write a function to generate all possible successor states from a given state.
3. Implement the Alpha–Beta pruning algorithm on top of the minimax framework to evaluate game states:
   - alpha = the best value found so far for the maximizing player.
   - beta = the best value found so far for the minimizing player.
   - Prune (skip exploring) branches when alpha ≥ beta.
4. Define the terminal state (no stones left).
   - Utility value: +1 if the maximizing player (MAX) wins, -1 if the minimizing player (MIN) wins.
5. The program should:
   - Take an initial state defined directly in the code (e.g., (3,4,5)).
   - Run Alpha–Beta pruning from that state.
   - Print the best move for MAX.
   - Show how many nodes were explored vs. pruned.

Deliverables:

- A Python program that:
  - Implements Alpha–Beta pruning for Nim.
  - Prints the explored moves, pruned branches, and the optimal move from the initial state.
  - Demonstrates efficiency improvements compared to plain minimax (fewer nodes explored).

Sample Run:

Initial State: (3, 4, 5)

MAX explores (3, 4, 5)
 Considering move: (3, 4, 5) → (2, 4, 5)
 Considering move: (3, 4, 5) → (3, 3, 5)
 [Pruned branch at (3, 4, 4) because alpha >= beta]

Best Move for MAX: (3, 4, 5) → (3, 4, 4)
Outcome: Winning position
Nodes Explored: 57
Nodes Pruned: 18

Evaluation Criteria:

- Correctness of alpha–beta pruning implementation.
- Clarity of code (readable functions, comments).
- Output showing explored moves, pruned branches, and best move.
- Comparison with plain minimax in terms of nodes expanded (bonus).