import sys

class NimGame:

    def \_\_init\_\_(self, num\_red, num\_blue, version='standard', first\_player='computer', depth=None):

        self.num\_red = num\_red

        self.num\_blue = num\_blue

        self.version = version

        self.current\_player = first\_player

        self.depth = depth

    def is\_game\_over(self, red, blue):

        if self.version == 'standard':

            return red == 0 or blue == 0

        else:

            return red == 0 or blue == 0

    def score(self):

        return self.num\_red \* 2 + self.num\_blue \* 3

    def human\_move(self):

        while True:

            try:

                num\_red = int(input("Enter number of red marbles to remove: "))

                num\_blue = int(input("Enter number of blue marbles to remove: "))

                if 0 <= num\_red <= self.num\_red and 0 <= num\_blue <= self.num\_blue and (num\_red + num\_blue) > 0:

                    self.num\_red -= num\_red

                    self.num\_blue -= num\_blue

                    break

                else:

                    print("Invalid move. Try again.")

            except ValueError:

                print("Invalid input. Try again.")

    def computer\_move(self):

        move = self.minimax(self.num\_red, self.num\_blue, True)

        num\_red, num\_blue = move[1], move[2]

        self.num\_red -= num\_red

        self.num\_blue -= num\_blue

        print(f"Computer removes {num\_red} red marbles and {num\_blue} blue marbles.")

    def minimax(self, red, blue, maximizing):

        if self.is\_game\_over(red, blue):

            if self.version == 'standard':

                return (-(red \* 2 + blue \* 3), 0, 0)

            else:

                return ((red \* 2 + blue \* 3), 0, 0)

        if maximizing:

            best\_value = float('-inf')

            best\_move = (0, 0)

            for move in self.get\_possible\_moves(red, blue):

                new\_red, new\_blue = red - move[0], blue - move[1]

                value = self.minimax(new\_red, new\_blue, False)[0]

                if value > best\_value:

                    best\_value = value

                    best\_move = move

            return (best\_value, best\_move[0], best\_move[1])

        else:

            best\_value = float('inf')

            best\_move = (0, 0)

            for move in self.get\_possible\_moves(red, blue):

                new\_red, new\_blue = red - move[0], blue - move[1]

                value = self.minimax(new\_red, new\_blue, True)[0]

                if value < best\_value:

                    best\_value = value

                    best\_move = move

            return (best\_value, best\_move[0], best\_move[1])

    def get\_possible\_moves(self, red, blue):

        moves = []

        if self.version == 'standard':

            if red >= 2: moves.append((2, 0))

            if blue >= 2: moves.append((0, 2))

            if red >= 1: moves.append((1, 0))

            if blue >= 1: moves.append((0, 1))

        else:

            if blue >= 1: moves.append((0, 1))

            if red >= 1: moves.append((1, 0))

            if blue >= 2: moves.append((0, 2))

            if red >= 2: moves.append((2, 0))

        return moves

    def play\_game(self):

        while not self.is\_game\_over(self.num\_red, self.num\_blue):

            print(f"Red Marbles: {self.num\_red}, Blue Marbles: {self.num\_blue}")

            if self.current\_player == 'human':

                self.human\_move()

                self.current\_player = 'computer'

            else:

                self.computer\_move()

                self.current\_player = 'human'

        print("Game over!")

        print(f"Final score: {self.score()}")

if \_\_name\_\_ == '\_\_main\_\_':

    num\_red = int(input("Enter the number of red marbles: "))

    num\_blue = int(input("Enter the number of blue marbles: "))

    version = input("Enter the game version (standard/misere): ")

    first\_player = input("Enter the first player (computer/human): ")

    game = NimGame(num\_red, num\_blue, version, first\_player)

    game.play\_game()

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated