BÀI TẬP LAB 3
Môn: Cơ sở trí tuệ nhân tạo (HK2 2024-2025) – Tuần 8

I. Thông tin sinh viên

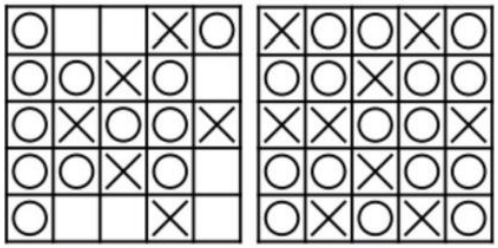
Họ và tên: Phan Ngọc Hà
MSSV: 23880020

II. Bài làm

Bài 1:

Đề bài:

- Đọc hiểu và trình bày lại theo cách hiểu của mình, diễn giải kỹ hơn ở mỗi bước và ánh xạ đến kiến thức lý thuyết đã được học.
- Dựa trên code có sẵn để cài đặt chạy trên máy của mình.
- Điều chỉnh code để chấp nhận input bất kỳ (trạng thái ban đầu bất kỳ). Ví dụ, đã có sẵn một số bước đã thực hiện.
- Thực hiện trên input có giới hạn lớn hơn như 10x10 hay 20 x 20 với luật chơi có thể thay đổi như 5 ô liên tiếp, …



Player 2 wins Player 1 wins

Bài làm:

Bài 1:

Ý 1:

Khai báo các biến và hàm cần thiết:

```
from math import inf as infinity
from random import choice
import platform
import time
from os import system

HUMAN = -1
COMP = +1
board = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0],
]
```

Khai báo biến:
*HUMAN là 1: đại diện cho con người
*COMP là +1: đại diện cho máy tính
*Ma trận dạng 3x3: thể hiện bảng chơi caro

```
def evaluate(state):
    if wins(state, COMP):
        score = +1
    elif wins(state, HUMAN):
        score = -1
    else:
        score = 0

    return score
```

Hàm tính điểm ứng với trạng thái hiện tại:
Hàm evaluate truyền vào param state là trạng thái hiện tại của bàn cờ.
Gọi tới hàm wins và kiểm tra để:
*gán score = + 1 : nếu máy tính thắng
*gán score = -1: nếu con người thắng
*gán score = 0: nếu hòa
Trả về score

```python
def wins(state, player):
    win_state = [
        [state[0][0], state[0][1], state[0][2]],
        [state[1][0], state[1][1], state[1][2]],
        [state[2][0], state[2][1], state[2][2]],
        [state[0][0], state[1][0], state[2][0]],
        [state[0][1], state[1][1], state[2][1]],
        [state[0][2], state[1][2], state[2][2]],
        [state[0][0], state[1][1], state[2][2]],
        [state[2][0], state[1][1], state[0][2]],
    ]
    if [player, player, player] in win_state:
        return True
    else:
        return False
```

Hàm kiểm tra thắng trò chơi:
Hàm wins truyền vào 2 param
*state: trạng thái của bàn cờ hiện tại
*player: người chơi hoặc máy tính
Ứng với 3 khả năng thắng:
* Ba hàng [X X X] hoặc [O O O]
* Ba cột [X X X] hoặc [O O O]
* Hai đường chéo [X X X] hoặc [O O O]
Đưa tất cả các trường hợp thắng này vào mảng win_state
Nếu [player,player,player] thuộc mảng thì return true, nghĩa là đã có người thắng

```python
def game_over(state):
    return wins(state, HUMAN) or wins(state, COMP)
```

Hàm kiểm tra trò chơi kết thúc:
Nếu người hoặc máy thắng thì trò chơi kết thúc, nghĩa là hàm wins trả về True thì hàm game_over trả về true

```python
def empty_cells(state):
    cells = []

    for x, row in enumerate(state):
        for y, cell in enumerate(row):
            if cell == 0:
                cells.append([x, y])

    return cells
```

Hàm empty_cells(state) có nhiệm vụ tìm và trả về danh sách tọa độ của các ô trống (ô có giá trị bằng 0) trong một ma trận state.
* enumerate(state) giúp duyệt qua từng hàng (row) của ma trận state, đồng thời lấy chỉ số hàng x cột y.
*Nếu giá trị của ô bằng 0, Thêm vị trí [x, y] vào danh sách cells.nghĩa là nó trống.

```python

def valid_move(x, y):
    if [x, y] in empty_cells(board):
        return True
    else:
        return False


def set_move(x, y, player):
    if valid_move(x, y):
        board[x][y] = player
        return True
    else:
        return False
```

Hàm valid_move truyền vào một tọa độ (x,y) sau đó kiểm tra có trống không (nằm trong danh sách mà hàm empty_cell trả về). Nếu trống thì trả về true

Hàm set_move truyền vào một tọa độ (x,y) và player sau đó kiểm tra nếu bước đi là valid nghĩa là đi trong ô trống thì gán tọa độ trong board ứng với player, rồi trả về true. Ngược lại trả về false.

Hàm sử dụng thuật toán minimax:

```python
def minimax(state, depth, player):
    if player == COMP:
        best = [-1, -1, -infinity]
    else:
        best = [-1, -1, +infinity]

    if depth == 0 or game_over(state):
        score = evaluate(state)
        return [-1, -1, score]

    for cell in empty_cells(state):
        x, y = cell[0], cell[1]
        state[x][y] = player
        score = minimax(state, depth - 1, -player)
        state[x][y] = 0
        score[0], score[1] = x, y

        if player == COMP:
            if score[2] > best[2]:
                best = score  # max value
        else:
            if score[2] < best[2]:
                best = score  # min value

    return best
```

Hàm minimax truyền vào state, depth, player:
*Nếu player hiện tại là Computer thì tìm MAX nên khai báo -infinity,
*Nếu player hiện tại là Human thì tìm MIN nên khai báo +infinity

Điều kiện dừng đệ quy:
nếu depth == 0 (đạt đến độ sâu tối đa) **hoặc** trò chơi đã kết thúc (game_over(state) == True), ta đánh giá trạng thái bàn cờ bằng evaluate(state).

*Duyệt tất cả các ô trống và thực hiện nước đi
*Gọi đệ quy để tìm giá trị tốt nhất cho trạng thái tiếp theo, với:
depth - 1: Giảm độ sâu tìm kiếm. Thử từng vị trí
-player: Đổi lượt chơi (ví dụ, nếu player = COMP = +1, lượt sau sẽ là HUMAN = -1).
*Trả lại trạng thái ban đầu (state[x][y] = 0) sau khi thử nước đi.
*Cập nhật vị trí của nước đi: Lưu lại vị trí `(x,y)` của nước đi hiện tại.

*Chọn nước đi tối ưu dựa trên Minimax
*Nếu **lượt đi của máy (COMP)**, ta **chọn giá trị lớn nhất** (nước đi có điểm số cao nhất).
*Nếu **lượt đi của người chơi (HUMAN)**, ta **chọn giá trị nhỏ nhất** (nước đi có điểm số thấp nhất).
*Thuật toán Minimax:
+ Máy chơi tối ưu (tối đa điểm số).
+ Người chơi giả định cũng chơi tối ưu (tối thiểu điểm số của máy).
*Trả về nước đi tốt nhất

In ra màn hình và hàm main():

```python
def clean():
    os_name = platform.system().lower()
    if 'windows' in os_name:
        system('cls')
    else:
        system('clear')

def render(state, c_choice, h_choice):
    chars = {
        -1: h_choice,
        +1: c_choice,
        0: ' '
    }
    str_line = '---------------'

    print('\n' + str_line)
    for row in state:
        for cell in row:
            symbol = chars[cell]
            print(f'| {symbol} |', end='')
        print('\n' + str_line)
```

Xóa console

In bảng trò chơi

```python
def ai_turn(c_choice, h_choice):

    depth = len(empty_cells(board))
    if depth == 0 or game_over(board):
        return

    clean()
    print(f'Computer turn [{c_choice}]')
    render(board, c_choice, h_choice)

    if depth == 9:
        x = choice([0, 1, 2])
        y = choice([0, 1, 2])
    else:
        move = minimax(board, depth, COMP)
        x, y = move[0], move[1]

    set_move(x, y, COMP)
    time.sleep(1)
```

*Tính độ sâu và kiểm tra kết thúc trò chơi
*Xóa màn hình và hiển thị thông tin
*Máy tính chọn nước đi
-Nếu depth == 9, nghĩa là đây **là nước đi đầu tiên của ván chơi** (bàn cờ hoàn toàn trống).
-Máy tính chọn một ô ngẫu nhiên (x, y) trong các vị trí [0, 1, 2] để đi trước. Choice là hàm random.
*Nếu **không phải nước đi đầu tiên**, máy tính **dùng thuật toán Minimax** để tìm nước đi tối ưu nhất.

```python
def human_turn(c_choice, h_choice):
    depth = len(empty_cells(board))
    if depth == 0 or game_over(board):
        return

    # Dictionary of valid moves
    move = -1
    moves = {
        1: [0, 0], 2: [0, 1], 3: [0, 2],
        4: [1, 0], 5: [1, 1], 6: [1, 2],
        7: [2, 0], 8: [2, 1], 9: [2, 2],
    }

    clean()
    print(f'Human turn [{h_choice}]')
    render(board, c_choice, h_choice)

    while move < 1 or move > 9:
        try:
            move = int(input('Use numpad (1..9): '))
            coord = moves[move]
            can_move = set_move(coord[0], coord[1], HUMAN)

            if not can_move:
                print('Bad move')
                move = -1
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')
```

*Tính độ sâu và kiểm tra kết thúc trò chơi
*Tạo danh sách nước đi hợp lệ
*Xóa màn hình và hiển thị thông tin
*Vòng lặp nhập nước đi.
*Kiểm tra và thực hiện nước đi
***Xử lý lỗi khi nhập**

```python
def main():
    clean()
    h_choice = ''  # X or O
    c_choice = ''  # X or O
    first = ''  # if human is the first

    # Human chooses X or O to play
    while h_choice != 'O' and h_choice != 'X':
        try:
            print('')
            h_choice = input('Choose X or O\nChosen: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Setting computer's choice
    if h_choice == 'X':
        c_choice = 'O'
    else:
        c_choice = 'X'
```

*Xóa màn hình và khởi tạo biến
*Người chơi chọn X hoặc O
*Gán ký hiệu cho máy tính

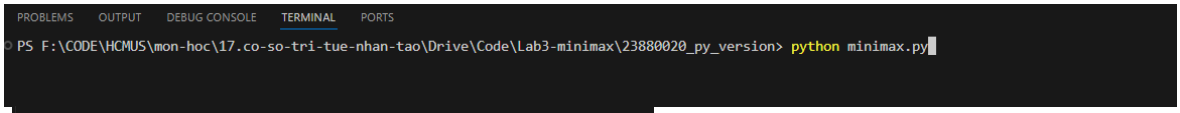
```python
    # Human may starts first
    clean()
    while first != 'Y' and first != 'N':
        try:
            first = input('First to start?[y/n]: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Main loop of this game
    while len(empty_cells(board)) > 0 and not game_over(board):
        if first == 'N':
            ai_turn(c_choice, h_choice)
            first = ''

        human_turn(c_choice, h_choice)
        ai_turn(c_choice, h_choice)
```

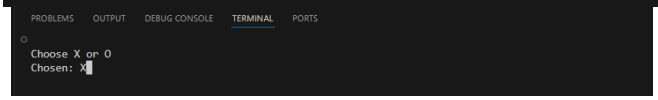
*Người chơi quyết định ai đi trước
*Chạy vòng lặp chính của trò chơi

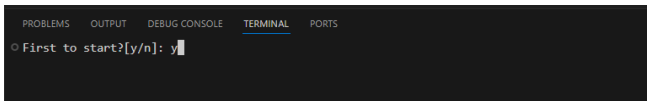
```python
    # Game over message
    if wins(board, HUMAN):
        clean()
        print(f'Human turn [{h_choice}]')
        render(board, c_choice, h_choice)
        print('YOU WIN!')
    elif wins(board, COMP):
        clean()
        print(f'Computer turn [{c_choice}]')
        render(board, c_choice, h_choice)
        print('YOU LOSE!')
    else:
        clean()
        render(board, c_choice, h_choice)
        print('DRAW!')

    exit()


if __name__ == '__main__':
    main()
```

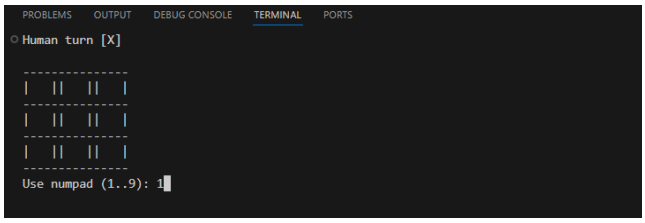
*Kiểm tra kết quả trò chơi
*Thoát chương trình
*Chạy chương trình nếu file được thực thi trực tiếp

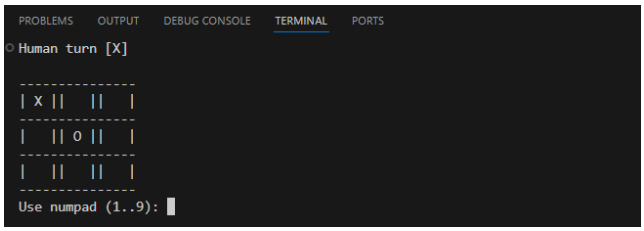
Ý 2: Dựa trên code có sẵn để cài đặt chạy trên máy:

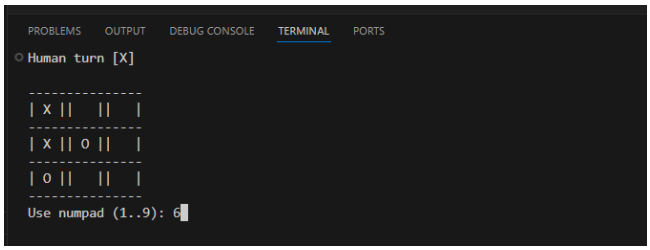


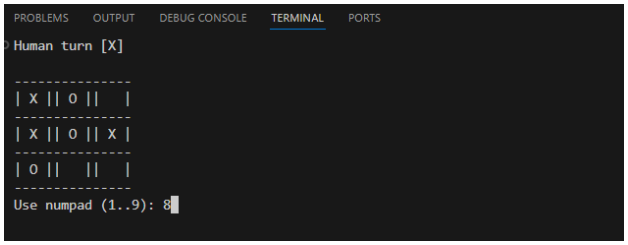


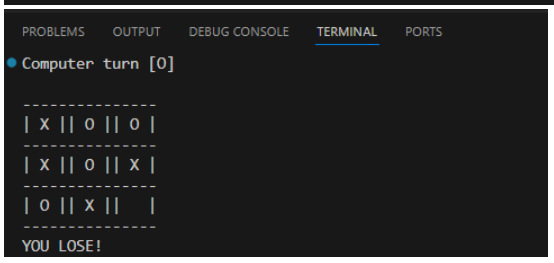












Ý 3: Điều chỉnh code để chấp nhận input bất kỳ (trạng thái ban đầu bất kỳ). Ví dụ, đã có sẵn một số bước đã thực hiện:

-Thêm hàm khai báo trạng thái ban đầu:

```python
def get_initial_board():
    """
    Nhập trạng thái ban đầu của bảng từ người dùng.
    Người dùng nhập 9 số (0: trống, 1: máy, -1: người) theo từng hàng.
    """
    print("Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):")
    print("0: Ô trống, 1: Máy (O/X), -1: Người (X/O)")

    while True:
        try:
            # Nhập 9 số nguyên
            values = list(map(int, input().split()))

            # Kiểm tra số lượng đúng 9 số
            if len(values) != 9 or not all(v in [-1, 0, 1] for v in values):
                raise ValueError

            # Chuyển danh sách thành ma trận 3x3
            return [values[:3], values[3:6], values[6:]]

        except ValueError:
            print("Dữ liệu không hợp lệ. Vui lòng nhập lại!")
```

Chỉnh lại hàm main:

```python
def main():
    clean()

    # Nhập trạng thái ban đầu của bảng từ người dùng
    global board
    board = get_initial_board()

    # Xác định số lượt đi còn lại
    depth = len(empty_cells(board))

    # Kiểm tra nếu trò chơi đã kết thúc
    if game_over(board) or depth == 0:
        render(board, "O", "X")  # Hiển thị trạng thái bảng
        if wins(board, HUMAN):
            print("Trò chơi đã kết thúc: Người chơi THẮNG!")
        elif wins(board, COMP):
            print("Trò chơi đã kết thúc: Máy TÍNH THẮNG!")
        else:
            print("Trò chơi đã kết thúc: HÒA!")
        exit()
```

```python
    h_choice = ''  # X or O
    c_choice = ''  # X or O
    first = ''  # if human is the first

    # Human chooses X or O to play
    while h_choice != 'O' and h_choice != 'X':
        try:
            print('')
            h_choice = input('Choose X or O\nChosen: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Setting computer's choice
    if h_choice == 'X':
        c_choice = 'O'
    else:
        c_choice = 'X'
```

```
    # Human may starts first
    clean()
    while first != 'Y' and first != 'N':
        try:
            first = input('First to start?[y/n]: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Main loop of this game
    while len(empty_cells(board)) > 0 and not game_over(board):
        if first == 'N':
            ai_turn(c_choice, h_choice)
            first = ''

        human_turn(c_choice, h_choice)
        ai_turn(c_choice, h_choice)
```

```
    # Game over message
    if wins(board, HUMAN):
        clean()
        print(f'Human turn [{h_choice}]')
        render(board, c_choice, h_choice)
        print('YOU WIN!')
    elif wins(board, COMP):
        clean()
        print(f'Computer turn [{c_choice}]')
        render(board, c_choice, h_choice)
        print('YOU LOSE!')
    else:
        clean()
        render(board, c_choice, h_choice)
        print('DRAW!')

    exit()


if __name__ == '__main__':
    main()
```

-Kết quả

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version> python minimax-initial-state.py
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):
0: Ô trống, 1: Máy (O/X), -1: Người (X/O)
0 0 1 -1 1 0 0 -1 0

Choose X or O
Chosen: X
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

First to start?[y/n]: y
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Human turn [X]

---------------
|   ||   || O |
---------------
| X || O ||   |
---------------
|   || X ||   |
---------------
Use numpad (1..9): 1
```

```
Computer turn [O]

---------------
| X ||   || O |
---------------
| X || O ||   |
---------------
| O || X ||   |
---------------
YOU LOSE!
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version>
```

Ý 4: Thực hiện trên input có giới hạn lớn hơn như 10x10 hay 20 x 20 với luật chơi có thể thay đổi như 5 ô liên tiếp.

```python
from math import inf as infinity
from random import choice
import platform
import time
from os import system

def get_game_settings():
    """ Nhập kích thước bảng và số lượng ô liên tiếp cần để thắng """
    while True:
        try:
            # Nhập kích thước bảng và điều kiện thắng
            size = int(input("Nhập kích thước bảng (ví dụ: 10 cho 10x10): "))
            win_condition = int(input(f"Nhập số ô liên tiếp cần để thắng (tối đa {size}): "))

            # Kiểm tra thông tin nhập vào
            if size < 3 or win_condition > size or win_condition < 3:
                print("Thông tin không hợp lệ! Hãy nhập lại.")
                continue

            return size, win_condition

        except ValueError:
            print("Vui lòng nhập số nguyên hợp lệ!")

# Lấy thông tin từ người chơi
BOARD_SIZE, WIN_CONDITION = get_game_settings()
HUMAN = -1
COMP = +1
```

```python
# Cập nhật trạng thái ban đầu theo bảng dynamic
def get_initial_board(size):
    """
    Nhập trạng thái ban đầu của bảng từ người dùng.
    Người dùng nhập `size * size` số (0: trống, 1: máy, -1: người) theo từng hàng.
    """
    print(f"Nhập trạng thái ban đầu của bảng ({size * size} số, cách nhau bởi dấu cách):")
    print("0: Ô trống, 1: Máy (O/X), -1: Người (X/O)")

    while True:
        try:
            # Nhập toàn bộ trạng thái bàn cờ
            values = list(map(int, input().split()))

            # Kiểm tra số lượng ô nhập vào có khớp với kích thước bảng không
            if len(values) != size * size or not all(v in [-1, 0, 1] for v in values):
                raise ValueError

            # Chuyển danh sách thành ma trận `size x size`
            return [values[i * size:(i + 1) * size] for i in range(size)]

        except ValueError:
            print(f"Dữ liệu không hợp lệ! Vui lòng nhập đúng {size * size} số (0, 1, -1) cách nhau bởi dấu cách.")

# Khai báo board game
board = get_initial_board(BOARD_SIZE)
```

```python
def evaluate(state):
    """ Đánh giá trạng thái trò chơi """
    if wins(state, COMP):
        score = +1  # Máy thắng
    elif wins(state, HUMAN):
        score = -1  # Người thắng
    else:
        score = 0  # Hòa

    return score

# Cập nhật luật thắng cho bảng lớn
def wins(state, player):
    """ Kiểm tra người chơi có thắng không với điều kiện WIN_CONDITION """
    size = len(state)

    # Kiểm tra theo hàng ngang
    for row in state:
        for i in range(size - WIN_CONDITION + 1):
            if all(cell == player for cell in row[i:i + WIN_CONDITION]):
                return True

    # Kiểm tra theo cột dọc
    for col in range(size):
        for i in range(size - WIN_CONDITION + 1):
            if all(state[i + j][col] == player for j in range(WIN_CONDITION)):
                return True

    # Kiểm tra đường chéo chính \
    for i in range(size - WIN_CONDITION + 1):
        for j in range(size - WIN_CONDITION + 1):
            if all(state[i + k][j + k] == player for k in range(WIN_CONDITION)):
                return True

    # Kiểm tra đường chéo phụ /
    for i in range(size - WIN_CONDITION + 1):
        for j in range(WIN_CONDITION - 1, size):
            if all(state[i + k][j - k] == player for k in range(WIN_CONDITION)):
                return True

    return False
```

```python
def game_over(state):
    """ Kiểm tra trò chơi đã kết thúc chưa """
    return wins(state, HUMAN) or wins(state, COMP)


def empty_cells(state):
    """ Tìm các ô trống trên bàn cờ """
    cells = []
    for x, row in enumerate(state):
        for y, cell in enumerate(row):
            if cell == 0:
                cells.append([x, y])

    return cells


def valid_move(x, y):
    """ Kiểm tra nước đi hợp lệ """
    if [x, y] in empty_cells(board):
        return True
    else:
        return False


def set_move(x, y, player):
    """ Thực hiện nước đi của người chơi hoặc máy """
    if valid_move(x, y):
        board[x][y] = player
        return True
    else:
        return False
```

```python
# Tối ưu Minimax với tỉa nhánh Alpha-Beta
def minimax(state, depth, player, alpha, beta):
    """ Thuật toán Minimax tối ưu với tỉa nhánh Alpha-Beta """
    if depth == 0 or game_over(state):
        return [-1, -1, evaluate(state)]  # Trả về điểm đánh giá nếu đã đến độ sâu tối thiểu hoặc kết thúc trò chơi

    best = [-1, -1, -infinity] if player == COMP else [-1, -1, +infinity]

    for x, y in empty_cells(state):
        state[x][y] = player
        score = minimax(state, depth - 1, -player, alpha, beta)
        state[x][y] = 0
        score[0], score[1] = x, y  # Lưu lại vị trí nước đi

        if player == COMP:
            if score[2] > best[2]:
                best = score
            alpha = max(alpha, score[2])  # Cập nhật alpha
        else:
            if score[2] < best[2]:
                best = score
            beta = min(beta, score[2])  # Cập nhật beta

        if beta <= alpha:
            break  # Cắt nhánh nếu alpha >= beta

    return best


def clean():
    """ Clear console """
    os_name = platform.system().lower()
    if 'windows' in os_name:
        system('cls')
    else:
        system('clear')
```

```python
# Cập nhật console để hiển thị bảng lớn hơn
def render(state):
    """ Hiển thị bàn cờ với kích thước linh hoạt """
    size = len(state)
    print("\n" + "   " + " ".join([str(i) for i in range(size)]))

    for i, row in enumerate(state):
        row_str = f"{i}  " + " | ".join(["X" if cell == HUMAN else "O" if cell == COMP else " " for cell in row])
        print(row_str)
        print("  " + "-" * (size * 4 - 1))  # In dấu "-" để phân cách các hàng


def ai_turn():
    """ Lượt đi của máy """
    depth = len(empty_cells(board))
    if depth == 0 or game_over(board):
        return

    clean()
    print("Lượt của Máy!")
    render(board)

    # Nếu bàn cờ toàn trống, máy chọn ngẫu nhiên
    if depth == BOARD_SIZE * BOARD_SIZE:
        x, y = choice(range(BOARD_SIZE)), choice(range(BOARD_SIZE))
    else:
        move = minimax(board, depth, COMP, -infinity, infinity)
        x, y = move[0], move[1]

    set_move(x, y, COMP)
    time.sleep(1)
```

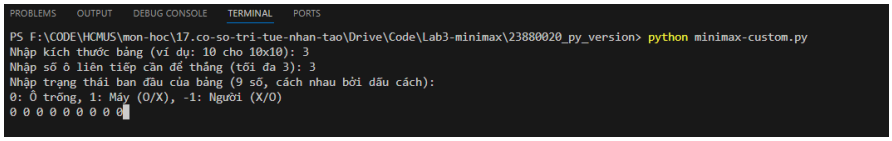
```python
def human_turn():
    """ Lượt đi của người chơi """
    depth = len(empty_cells(board))
    if depth == 0 or game_over(board):
        return

    clean()
    print("Lượt của Bạn!")
    render(board)

    # Người chơi nhập tọa độ để đánh
    while True:
        try:
            x, y = map(int, input("Nhập tọa độ (dạng: dòng cột): ").split())
            if set_move(x, y, HUMAN):
                break
            else:
                print("Nước đi không hợp lệ, thử lại.")
        except ValueError:
            print("Vui lòng nhập số hợp lệ!")
```

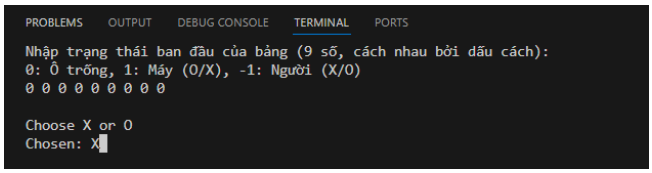
```python
def main():
    """ Chạy trò chơi """
    clean()

    # Nhập trạng thái ban đầu của bảng từ người dùng
    global board
    board = get_initial_board(BOARD_SIZE)

    # Xác định số lượt đi còn lại
    depth = len(empty_cells(board))

    # Kiểm tra nếu trò chơi đã kết thúc
    if game_over(board) or depth == 0:
        render(board)  # Hiển thị trạng thái bảng
        if wins(board, HUMAN):
            print("Trò chơi đã kết thúc: Người chơi THẮNG!")
        elif wins(board, COMP):
            print("Trò chơi đã kết thúc: Máy TÍNH THẮNG!")
        else:
            print("Trò chơi đã kết thúc: HÒA!")
        exit()

    h_choice = ''  # X or O
    c_choice = ''  # X or O
    first = ''  # if human is the first

    # Người chơi chọn X hoặc O
    while h_choice != 'O' and h_choice != 'X':
        try:
            print('')
            h_choice = input('Choose X or O\nChosen: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

```

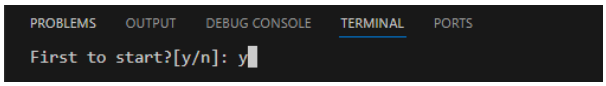
```
    # Chọn lựa của máy
    if h_choice == 'X':
        c_choice = 'O'
    else:
        c_choice = 'X'

    # Người chơi có thể bắt đầu trước
    clean()
    while first != 'Y' and first != 'N':
        try:
            first = input('First to start?[y/n]: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Main loop của trò chơi
    while len(empty_cells(board)) > 0 and not game_over(board):
        if first == 'N':
            ai_turn()
            first = ''

        human_turn()
        ai_turn()
```

```
    # Thông báo kết quả khi trò chơi kết thúc
    if wins(board, HUMAN):
        clean()
        print(f'Human turn [{h_choice}]')
        render(board)
        print('YOU WIN!')
    elif wins(board, COMP):
        clean()
        print(f'Computer turn [{c_choice}]')
        render(board)
        print('YOU LOSE!')
    else:
        clean()
        render(board)
        print('DRAW!')

    exit()


if __name__ == '__main__':
    main()
```

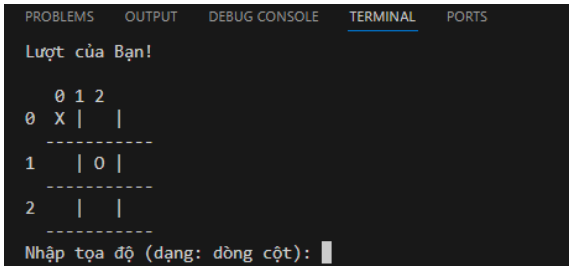
-Kết quả


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version> python minimax-custom.py
Nhập kích thước bảng (ví dụ: 10 cho 10x10): 3
Nhập số ô liên tiếp cần để thắng (tối đa 3): 3
Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):
0: Ô trống, 1: Máy (O/X), -1: Người (X/O)
0 0 0 0 0 0 0 0 0


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):
0: Ô trống, 1: Máy (O/X), -1: Người (X/O)
0 0 0 0 0 0 0 0 0
Choose X or O
Chosen: X

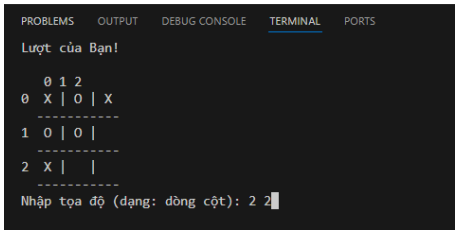

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
First to start?[y/n]: y


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!
0 1 2
0 | |
1 | |
2 | |
Nhập tọa độ (dạng: dòng cột): 0 0


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!
0 1 2
0 X | |
1 | O |
2 | |
Nhập tọa độ (dạng: dòng cột):


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!
0 1 2
0 X | |
1 | O |
2 | |
Nhập tọa độ (dạng: dòng cột): 0 2


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!
0 1 2
0 X | O | X
1 | O |
2 | |
Nhập tọa độ (dạng: dòng cột): 2 0


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!
0 1 2
0 X | O | X
1 O | O |
2 X | |
Nhập tọa độ (dạng: dòng cột): 2 2


Computer turn [0]
0 1 2
0 X | O | X
1 O | O |
2 X | | X
YOU LOSE!
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version>
On Battery