BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

Artificial Intelligence (BITS F444/ CS F407)
I Semester 2019-20
Programming Assignment-3
Coding Details
(October 17, 2019)

Instruction: Type the details precisely and neatly

1. ID : **2019H1030023P**
   Name: **Subhashis Dhar**

2. Mention the names of Submitted files :
   a. **ana.png**
   b. **logo.png**
   c. **main.py**
   d. **start.png**

3. Total number of submitted files: **4**
4. Name of the folder : **2019H1030023P_CSF407_PA_3**
5. Have you checked that all the files you are submitting have your name in the top? **Yes**
6. Have you checked that all the files you are submitting are in the folder as specified in 4 (and no subfolder exists)? **Yes**

7. Problem formulation
   a. State representation:
      State is represented by a 4x4 matrix with 0 denoting free space, 1 denoting Human piece and 2 denoting AI piece

   b. Pseudo code of your successor function

```
successor(board,c,p){
    c <- column where to place piece
    p <- whose piece to place
    r <- find next open row in column c
    if r is a valid location to place piece:
        board[r][c] <- p
    return board;
}
```

   c. Terminal states generation process
      Terminal state is detected staring from visiting window size of 3 from where the last piece is placed. 4 directions, horizontal, vertical, and 2 diagonals are searched for presence of 3 pieces of same type. From this window size I am checking the total number of pieces of a player. If piece count == 3 means this is a terminal state. Also check if the board is full and no moves are possible. This is also a terminating condition.

   d. Data structure to store terminal states
      Terminal states are not stored. I am checking whether a state is terminal or not using terminalState function after successor function gives me next state.

e. Method to access terminal states and corresponding utility values

I am checking whether a state is terminal or not using terminalState function after successor function gives me next state. Terminal nodes will have utility values depending on win or loss. If not terminal and search has reached depth limit, then I need to score the board using a certain heuristic. For that I have 3 conditions:

1. **If player is making 3 in a row, then it is winning state. Give +100**
2. **If player has 2 in a row, it is a good state. Give +5**
3. **If opponent has 2 in a row, then 1st prefer if you are winning or not. If not then block the opponent. Give -80.**

This scoring mechanism gives a utility value for each state.

8. Minimax Technique details
   a. Node structure:
   Board – 4X4 Matrix
   Column – which move resulted in this state
   Score – utility value associated with this node

   b. Method to ensure the correctness of terminal test (describe in maximum 4 lines)
   Terminal conditions are not stored. Whenever a new state is returned by successor function, it is tested for terminal state. Window size of 3 is calculated in 4 directions, horizontal, vertical, and 2 diagonals. From this window size I am checking the total number of pieces of a player. If piece count == 3 means this is a terminal state. Also check if the board is full and no moves are possible. This is also a terminating condition.

   c. Total number of nodes generated to play one game: **Varies from game to game: Typical value: 12552**
   d. Write the statistics here as asked
   Values vary from run to run: Typical values obtained are:
   R1 = **12552** R2 = **96bytes** R3 = **Limited to 6**
   R4 = **3.895 s** R5= **5203.728**

   e. Code status (implemented fully/ partially/ not done) **Implemented fully**

9. Alpha Beta technique details:
   a. Explain the logic used for pruning (in maximum four lines)

If the maximum score that the minimizing player is assured of becomes less than the minimum score that the maximizing player is assured of (i.e. beta ≤ alpha), then the maximizing player need not consider further descendants of this node, as they will never be reached in the actual play. Alpha and Beta denote the lower and upper bounds of the value that I am interested in. This windows size is initially -infinty to + infinity. As we progress through the game, this window reduces. If we get any value that is outside this limit, we discard it or prune it.

   b. Total number of nodes generated to play one game: **Varies from game to game: Typical value: 24**
   c. Write the statistics here as asked
   Values vary from run to run: Typical values obtained are:
   R6 = **24** R7 = **624** R8 = **2.954s**

   d. Code status (implemented fully/ partially/ not done) **Implemented fully**

10. Comparative analysis
    R9 = **2161536/3456 Bytes** R10 = **39.48ms/28.86ms** R11= **10/10**
    R12= **10/10**

Fill in the following information based of 10 independent games

	Minimax Algorithm	Alpha Beta Pruning
Average number of nodes created	**13425**	**26**
Average time taken	**3.86s**	**2.85s**
Number of times machine wins (player M)	**10**	**10**

11. GUI details
    a. Created the GUI (yes/ No): **Yes**
    b. Have created it according to the specifications?(yes/No) **Yes**
    c. Which module of Python is used for creating graphics? **PyQT5**
    d. Is this under the standard Python library or not? **Yes**
    e. If not, why? **N/A**

12. Graphics details:
    a. Is graphics working fine for displaying the board and coins? **Yes. different colored discs are used to represent human and ai pieces. Legend present in the program**

    b. How have you calibrated the board and accepted human input to play the game? **Yes. Human input can be given by pressing any of 4 buttons which correspond to 4 columns of the board**

    c. How are you showing the base line? **Yes**
    d. How are you showing the move of the machine? **Shown by disc of blue colour on the board**
    e. How are you showing the move of the human player? **Shown by disc of red colour on the board**

13. Compilation Details:
    a. Code Compiles (Yes/ No): **Yes**
    b. Mention the .py files that do not compile: **N/A**
    c. Any specific function that does not compile: **N/A**
    d. Ensured the compatibility of your code with the specified Python version(yes/no) **Yes**
    e. Instructions for compilation of your files mentioning the multi file compilation process used by you (We may use the replica of these for compiling your files while evaluating your code)
    python main.py

14. Driver Details: Does it take care of the options specified earlier(yes/no): **Yes**
15. Execution status (describe in maximum 2 lines)
    We have to select start button to start the game. AI will play first. We can then use the column buttons to specify which column we want to place our disc. Then AI places its move. This cycle continues unless anyone wins or there are no empty spaces on the board. Analysis can be seen using analysis module.
    Note: Analysis figures are cumulative. They add up as you play. Thus, comparing minimax stats with 2 games played and alpha beta prune with 1 game does not result in valid comparison. This information of how many of each game is played is shown in GUI

16. Declaration: I, **Subhashis Dhar** declare that I have put my genuine efforts in creating the python code for the given programming assignment and have submitted only the code developed by me. I have not copied any piece of code from any source. If the code is found plagiarized in any form or degree, I understand that a disciplinary action as per the institute rules will be taken against me and I will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.
    ID : **2019H1030023P** Name: **Subhashis Dhar**
    Date: **17th October 2019**

****