**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Artificial Intelligence (BITS F444/ CS F407)**

**I Semester 2017-18**

**Programming Assignment-2**

**Coding Details**

**(October 3, 2017)**

*Instruction: Type the details precisely and neatly*

1. ID : **2015A7PS0078P**

Name: **Naveen Venkat**

1. Mention the names of Submitted files :
   1. modules.py
   2. graphics.py
   3. driver.py
   4. config.py
2. Total number of submitted files: **4**
3. Name of the folder : **2015A7PS0078P**
4. Have you checked that all the files you are submitting have your name in the top? **yes**
5. Have you checked that all the files you are submitting are in the folder as specified in (and no subfolder exists)? **yes**
6. Problem formulation
   1. State representation:

**[ <boardcells>, <player> ]**

**Where, <boardcells> is a list (16 values) of the 4x4 board represented in row major form. Each of these 16 values can be either +1 (maxplayer’s coin), -1 (minplayer’s coin) or 0 (empty cell).**

**<player> is a variable denoting the player whose move is next (minplayer (-1) or maxplayer(+1) )**  
**eg. The initial state is: [ [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], 1]**

* 1. Pseudo code of your successor function

**successor\_function(state, action):**

**depending upon the action {1,2,3,4}:  
 row = get\_the\_empty\_row\_of\_ith\_column( board, action )**

**col = action – 1  
 board[row][col] = player**

* 1. Terminal states generation process

**Enumerate the possible triplets corresponding to the terminal states (a triplet is a list of 3 indices in the row major representation of the 4x4 board that if occupied by a player would result in his win)**

* 1. Data structure to store terminal states

**List of triplets of possible locations for terminal states. These triplets determine the adjacent cells that one needs to occupy simultaneously to win  
Eg. [ [0,1,2], [1,2,3], [4,5,6], [5,6,7] … and so on ]**

* 1. Method to access terminal states and corresponding utility values

**The function terminal\_test checks if any of the triplets mentioned in 7. (d) contain the same values (i.e. one of the two players has played his move in those cells). If so, then that player would win and the corresponding utility value (+1 for maxplayer, -1 for minplayer) would be returned.**

1. Minimax Technique details
   1. Node structure:

**Node has variables to store each of : state, action taken to reach that node, utility value of the node**

* 1. Method to ensure the correctness of terminal test (describe in maximum 4 lines)  
     **Only two possibilities are there to this game. Either one of the players would win, or a draw would occur. The terminal test considers all the possibilities of the winning situations by brute force. Also, if any such triplet combination is not found, neither parties would win - hence a draw would occur. This is also captured by the terminal test. Thus it is correct.**
  2. Total number of nodes generated to play one game: **7251628**
  3. Write the statistics here as asked

R1 = **7251628** R2 = **896 Bytes** R3 = **16**

R4 = **88.667 s** R5= **0.9883 nodes per µs**

* 1. Code status : **implemented fully**

1. Alpha Beta technique details:
   1. Explain the logic used for pruning (in maximum four lines)  
      **The pruning algorithm follows that in the textbook. Alpha (in alphabeta\_minvalue function) and Beta (in alphabeta\_maxvalue function) values are compared with the value of v obtained from the children of the node running these functions. In case the comparison results in true, further checking of the children is ceased and the value highest (in maxvalue) or the lowest (in minvalue) of v is returned.**
   2. Total number of nodes generated to play one game: **43910**
   3. Write the statistics here as asked

R6 = **43910** R7 = **0.9934** R8 = **6.42134 s**

1. Code status **implemented fully**

1. Comparative analysis

Fill in the following information based of 10 independent games

|  |  |  |
| --- | --- | --- |
|  | Minimax Algorithm | Alpha Beta Pruning |
| Average number of nodes created | **7251628** | **43910** |
| Average time taken | **84.4182** | **6.42134** |
| Number of times machine wins (player M) | **10** | **10** |

1. GUI details
   1. Created the GUI : **yes**
   2. Have created it according to the specifications? **yes**
   3. Which module of Python is used for creating graphics? **turtle**
   4. Is this under the standard Python library or not? **yes**
   5. If not, why? **N/A**
2. Graphics details:
   1. Is turtle graphics working fine for displaying the board and coins? **yes**
   2. How have you calibrated the board and human input to play the game?

**Each click on the screen will be sent to the humanEvent function which calls other functions to : (1) check whether the correct cell has been clicked, and, (2) if yes and if it is the human’s move then it will play the move. Wrong clicks will be ignored.**

* 1. How are you showing the base line?

**The base line is assumed to be above the 4x4 board (the moves will be played in a top – bottom fashion)**

* 1. How are you showing the move of the machine?

**Using a green circle at the corresponding cell**

* 1. How are you showing the move of the human player?

**Using blue circle at the corresponding cell**

1. Compilation Details:
   1. Code Compiles (Yes/ No): **Yes**
   2. Mention the .py files that do not compile: **N/A**
   3. Any specific function that does not compile: **N/A**
   4. Ensured the compatibility of your code with the specified Python version : **Yes**
   5. 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).   
      **Please execute the following from the terminal:  
      python driver.py**
2. Driver Details: Does it take care of the options specified earlier: **Yes**
3. Execution status

**The code executes properly as specified. The minimax algorithm will take a while (~70-80 s) for the first move. After that, it improves exponentially (owing to the reduced number of nodes in the search tree). I have set the appropriate settings in the config.py file (important) so that the expected result may come. Changing the values in this file may hamper the execution.**

1. Declaration: I, Naveen Venkat 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: 2015A7PS0078P Name: Naveen Venkat

Date: 3/10/17

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Should not exceed three pages