ASSIGNMENT-1

Members:

Siri Akanksha Grandhi – 1910110393

Sree Sahithi Kantamneni – 1910110402

Contribution:

Both of us have done it equally by dividing all the functions. (50% and 50%)

CLASSES:

- MagicCube
- Game

FUNCTIONS IN MagicCube:

- **correctIndex(int) : int**

  This function checks and returns the correct index after each row and column increment.
- **MagicCube() : int [][][]**

  Generates a Magic Cube by keeping the first element in ((n/2),(n/2),(n-1)) position in the 3D matrix. Then it keeps inserting values into places by incrementing rows and decrementing columns. If in any case, the position is filled, then from that position, insert the next element in the position, which we can obtain from incrementing columns and decrementing layers. If in any case, that position is also non empty, then we can go to the new position from the older position by incrementing rows and layers.

FUNCTIONS IN Game:

*(**place**: a number in the magic cube; **position**: indexes of a place in the magic cube; Computer:’O’; Human:’X’)*

- **isFull(char[][][]) : Boolean**

  This function returns true if the entire game board is filled. In any other cases, it returns false.
- **combination(char[][][], ArrayList, int) : int**

  This function adds all the winning chance positions into an array. This function has an input argument ArrayList which contains all the places in the magic cube where the human or computer already placed their ‘X’ or ‘O’ respectively, depending on the situation. Function takes 2 values from the input ArrayList and checks the magic cube condition with the other empty places and checks for collinearity and places all the resultant empty places that can make a collinear line, into another array. That particular array is passed into another function called as bestInsertCondition() which returns an array. Combination function returns a value present in the magic cube, if all the functions return meaningful values. In the other case, combination returns -1.
- **collinear(int, int) : Boolean**

  It checks whether places a, b and (42-a-b) in the magic cube form a collinear line or not.
- **bestInsertCondition(char[][][], ArrayList, ArrayList) : int[]**

  It takes Array from the combination function and checks for collinearity. It checks for the empty places in the magic cube which can have maximum number of collinear lines. It then stores the value of the place and its count for the maximum number of collinear lines possible in an array. The function returns the array.
- **winChance(char[][][], ArrayList, int) : int**

  It calls combination function and returns the value returned by it.
- **countLines(char[][][], ArrayList, int) : int**

This function returns the number of collinear lines made by human and computer.

- **printCount(char[][][], ArrayList, int, char) : void**

  This function calls countLines() and prints the collinear lines of human and computer.

- **insertBest(char[][][], ArrayList) : int[]**

  If computer cannot get any winning condition and computer has nothing to block human's play, then instead of making computer place 'O' in a random position, this function check for places where most number of collinear lines can be formed. It returns an array which has the place along with the number of collinear lines that can be formed from it.

- **insertCorner(char[][][]) : int**

  This functions returns the places which are the corners of the magic cube.

- **getIndex(int, int[][][]) : int[]**

  This function returns the position of place given as argument in the magic board.

- **isEmpty(int, char[][][]) : Boolean**

  This function calls the getIndex function and checks if that particular position is empty or not.

- **printBoard(char[][][]) : void**

  This function prints the board of the game.