**Challenge- 1:** *Time Calculator* **(**5**)**

#include <iostream>

using namespace std;

enum GameStatus { DRAW, WIN, IN\_PROGRESS };

enum PlayerTurn { FIRST\_PLAYER = 1, SECOND\_PLAYER };

class GameBoard

{

private:

char data[3][3][3]; // Array used for board

GameStatus gameStatus = IN\_PROGRESS;

int validMovesCount = 0;

public:

GameBoard();

void displayBoard();

bool isValidPosition(char pos);

bool isAlreadyMarked(char pos);

void markBoard(char pos, char playerSymbol);

int getValidMovesCount();

GameStatus getGameStatus();

};

GameBoard::GameBoard()

{

char ch = '1';

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

for (int k = 0; k < 3; k++, ch++)

{

data[i][j][k] = ch;

}

}

ch = '1';

}

}

void GameBoard::displayBoard()

{

cout << endl;

for (int j = 0; j <= 2; j++)

{

for (int k = 0; k <= 2; k++)

{

cout << " ";

for (int i = 0; i <= 2; i++)

{

cout << data[k][j][i];

cout << " ";

}

cout << " ";

}

cout << endl;

}

cout << "----------- ----------- -----------\n 1\t 2\t 3";

cout << "\n\n";

}

bool GameBoard::isValidPosition(char pos)

{

return ((pos % 10 >= 1) && (pos % 10 <= 9));

}

bool GameBoard::isAlreadyMarked(char pos)

{

int i = (pos / 10) - 1, j = (pos % 10) - 1;

return ((data[i][j / 3][j % 3] >= '1' && data[i][j / 3][j % 3] <= '9'));

}

void GameBoard::markBoard(char pos, char playerSymbol)

{

int condition = pos % 10 – 1;

data[pos / 10 - 1][condition /3][ condition %3] = playerSymbol;

validMovesCount++;

}

int GameBoard::getValidMovesCount()

{

return validMovesCount;

}

GameStatus GameBoard::getGameStatus()

{

return gameStatus;

}

int main()

{

//TicTacToe tic;

//tic.playGame();

return 0;

}