**Lab 03**

**Object Oriented Programming Lab**

**Marks: 33**

**Challenge-1:** *TIC TAC TOE*

**GameBoard.h**

#ifndef GAMEBOARD\_H

#define GAMEBOARD\_H

#include <iostream>

using namespace std;

enum PlayerTurn { FIRST\_PLAYER = 1, SECOND\_PLAYER = 2 };

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

class GameBoard

{

private:

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

int validMovesCount = 0;

GameStatus gameStatus = IN\_PROGRESS;

void updateGameStatus();

bool checkBoardWinStatus(int );

public:

GameBoard();

void displayBoard();

void markBoard(int pos, char playerSymbol);

bool isValidPosition(int pos);

bool isAlreadyMarked(int pos);

int getValidMovesCount();

GameStatus getGameStatus();

};

#endif // !GAMEBOARD\_H

**GameBoard.cpp**

#include "GameBoard.h"

//Private Functions:

void GameBoard::updateGameStatus() ----------- (6)

{

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

{

for (int j = 0; j < 3; j++) //checking 3D WIN status

{

if ((data[0][i][j] < '1') || (data[0][i][j] > '9'))

{

if (data[0][i][j] == data[1][i][j] && data[1][i][j] == data[2][i][j])

{

gameStatus = WIN;

return;

}

}

}

}

int board = 0;

while (board < 3) //checking every board WIN status

{

if (checkBoardWinStatus(board))

{

gameStatus = WIN;

return;

}

board++;

}

if (validMovesCount == 27)

gameStatus = DRAW;

}

Checking 3D win status : 3

Unique board checking : 2

Atomicity (checkBoardWinStatus) -> -0.25

Correct Draw status check : 1

bool GameBoard::checkBoardWinStatus(int board) //to check every board WIN status ------------- (4)

{

char lastSymbol;

for (int j = 0; j < 3; j++) //checking down every col, changing rows

{

lastSymbol = data[b][0][j];

int i = 1;

while ((i < 3) && (lastSymbol == data[b][i][j]))

{

i++;

}

if (i == 3)

return true;

}

for (int i = 0; i < 3; i++) //checking across every row, changing cols

{

lastSymbol = data[b][i][0];

int j = 1;

while ((j < 3) && (lastSymbol == data[b][i][j]))

{

j++;

}

if (j == 3)

return true;

}

if ((data[b][0][0] == data[b][1][1]) && (data[b][1][1] == data[b][2][2])) //checking left diagonal

return true;

if ((data[b][0][2] == data[b][1][1]) && (data[b][1][1] == data[b][2][0])) //checking right diagonal

return true;

return false;

}

Vertical line check : 1.5

Horizontal line check : 1.5

Diagonals check : 1

//Public Functions:

GameBoard::GameBoard() ------------- (2)

{

char ch = '1';

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

{

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

{

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

{

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

}

}

ch = '1';

}

}

void GameBoard::displayBoard() ------------- (4)

{

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";

}

if displaying in different format -> 2

void GameBoard::markBoard(int pos, char playerSymbol) ------ (6)

{

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

data[i][j / 3][j % 3] = playerSymbol;

validMovesCount++;

updateGameStatus();

//updating gameStatus if there is no function of updateStatus the marks of the updateStatus shift here and there will be deduction of 1 marks for breaking automicity.

}

Finding correct board no : 1

Finding correct cell no. (row,col) : 2

Incrementing validMovesCount : 1

Updating game status : 2

(Atomicity -> -1)

bool GameBoard::isValidPosition(int pos) -----------(2)

{

if ((pos / 10) < 1 || (pos / 10) > 3)

return false;

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

}

Valid board no. check : 1

Valid cell no. check : 1

bool GameBoard::isAlreadyMarked(int pos) ----------- (3)

{

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

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

}

Finding correct board no. : 1

Finding correct cell no. (row,col) : 2

int GameBoard::getValidMovesCount() ------------- (0.5)

{

return validMovesCount;

}

GameStatus GameBoard::getGameStatus() ------------ (0.5)

{

return gameStatus;

}

**TicTacToe.h**

#ifndef TIC\_TAC\_TOE\_H

#define TIC\_TAC\_TOE\_H

#include"GameBoard.h"

class TicTacToe

{

bool isValidPlayerSymbol(char );

void inputPlayers(char& player1Symbol, char& player2Symbol);

public:

void playGame();

};

#endif // !TIC\_TAC\_TOE\_H

**TicTacToe.cpp**

#include "TicTacToe.h"

//Private Functions:

bool TicTacToe::isValidPlayerSymbol(char symbol)

{

return !(symbol >= '1' && symbol <= '9');

}

void TicTacToe::inputPlayers(char& player1Symbol, char& player2Symbol)

{

bool validSymbol;

do

{

cout << "Enter Player 1 Symbol : ";

cin >> player1Symbol;

validSymbol = isValidPlayerSymbol(player1Symbol);

if (!validSymbol)

cout << "Not a Valid Symbol!\n";

}

while (!validSymbol);

do

{

cout << "Enter Player 2 Symbol : ";

cin >> player2Symbol;

validSymbol = isValidPlayerSymbol(player2Symbol);

if (!validSymbol || player2Symbol == player1Symbol)

cout << "Not a Valid Symbol!\n";

}

while (!validSymbol || player2Symbol == player1Symbol);

}

//Public Functions:

void TicTacToe::playGame() ------------ (5)

{

char player1Symbol;

char player2Symbol;

inputPlayers(player1Symbol,player2Symbol);

GameBoard board;

int pos;

int random = 1 + rand() % 2;

PlayerTurn currentPlayer = (random == 1 ? FIRST\_PLAYER : SECOND\_PLAYER);

char currentSymbol = (random == 1 ? player1Symbol : player2Symbol);

while (board.getGameStatus() == IN\_PROGRESS)

{

bool validMove = false;

board.displayBoard();

do

{

cout << "\nPlayer " << currentPlayer << " turn: Enter Cell #: ";

cin >> pos;

if (board.isValidPosition(pos) && !board.isAlreadyMarked(pos))

{

board.markBoard(pos, currentSymbol);

validMove = true;

}

}

while (!validMove);

if (board.getGameStatus() == WIN)

{

board.displayBoard();

cout << "\nGame Won by Player : " << currentPlayer << "\n";

}

else if (board.getGameStatus() == DRAW)

{

cout << "\nGame DRAW";

}

else

{

currentSymbol = (currentPlayer == FIRST\_PLAYER ? player2Symbol : player1Symbol);

currentPlayer = (currentPlayer == FIRST\_PLAYER ? SECOND\_PLAYER : FIRST\_PLAYER);

}

}

}

Random player selection for 1st turn : 1.5

Win condition : 0.5

Draw condition : 0.5

Changing players if game continues : 0.5

For rest of game flow if running correctly : 2

No/wrong input validation for players symbol -> -0.75

No Atomicity -> -0.5

No/wrong input validation for move -> -0.75

**QUICK REVISION:**

GameBoard() (2)

void displayBoard() (4)

if displaying in different format -> 2

void markBoard(int pos, char playerSymbol) (6)

Finding correct board no : 1

Finding correct cell no. (row,col) : 2

Incrementing validMovesCount : 1

Updating game status : 2

(Atomicity -> -1)

bool isValidPosition(int pos) (2)

Valid board no. check : 1

Valid cell no. check : 1

bool isAlreadyMarked(int pos) (3)

Finding correct board no. : 1

Finding correct cell no. (row,col) : 2

int getValidMovesCount() (0.5)

GameStatus getGameStatus (0.5)

updateGameStatus (total 6)

Checking 3D win status : 3

Unique board checking : 2

Atomicity (checkBoardWinStatus) -> -0.25

Correct Draw status check : 1

checkBoardWinStatus (total 4)

Vertical line check : 1.5

Horizontal line check : 1.5

Diagonals check : 1

void TicTacToe::playGame() (5)

Random player selection for 1st turn : 1.5

Win condition : 0.5

Draw condition : 0.5

Changing players if game continues : 0.5

For rest of game flow if running correctly : 2

No/wrong input validation for players symbol -> -0.75

No Atomicity -> -0.5

No/wrong input validation for move -> -0.75

**Penalty Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Penalty List | Labs | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Indentation putting { Infront of loop header, in do while, putting while with closing } | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Meaningful Variable Names |  | -2 | -2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Camel Case Notation | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Atomicity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Syntax error | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Linker error | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong function prototypes | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Class interface or additional members |  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use of library function/class without permission | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Continue statement | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| cin/cout where it isn’t needed | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Multi-filing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong #ifndef or name of header file |  |  | -2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Global functions |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Multiple classes in one header file |  |  | -3 |  |  |  |  |  |  |  |  |  |  |  |  |  |