**Tic Tac Toe Board Analyzer Design Instruction**

1. **The cases of winner**

**Win-Group** refers to the group of the same symbols, such as X, which are lined in the horizontal or vertical or diagonal direction. The number of the symbols equal to the Number-to-Win. (Without specification, the default of Number-to-Win is 3 in the following samples).

For winner “X”, it win the game when:

* There is only one Win-Group with symbol “X” Or all Win-Groups (2 or above) are intersected in the same grid.
* The total count of “X” is 1 more than that of “O”.

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| ∙ | X | ∙ | O |
| O | O | X | ∙ |
| . | X | . | O |

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| ∙ | X | ∙ | O |
| X | O | X | ∙ |
| O | ∙ | ∙ | O |

For winner “O”, it wins the game when:

* There is only one Win-Group with symbol “O” Or all Win-Groups (2 or above) are intersected in the same grid.
* The total count of “O” equals to that of “X”.

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| ∙ | X | ∙ | O |
| O | O | O | ∙ |
| X | X | . | O |

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| O | ∙ | O | X |
| X | O | X | ∙ |
| O | ∙ | O | ∙ |

|  |  |  |  |
| --- | --- | --- | --- |
| X | O | O | X |
| O | X | X | O |
| X | O | O | X |
| O | X | X | O |

1. **The case of DRAW**

It is DRAW when:

* There is no any empty grid in the board.
* No winner.
* No any ERRORs.

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| ∙ | ∙ | O | ∙ |
| ∙ | ∙ | ∙ | X |
| O | O | ∙ | X |

1. **The case of IN\_PROGRESS**

It is IN\_PROGRESS when:

* There is at least one empty grid in the board.
* No winner.
* No any ERRORs.

1. **The cases of ERRORs**

* ERROR\_CASE1: Invalid Board Error Winning Sequence: More than twice of Number-to-Win jointed in a line (Here, Number-to-Win=2).

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | ∙ | O |
| ∙ | X | ∙ | ∙ |
| ∙ | O | X | ∙ |
| O | ∙ | ∙ | X |

* ERROR\_CASE2: Invalid Board Error Winning Sequence: 2 and above Win-Groups but not all the wins intersected in the same gird.

|  |  |  |  |
| --- | --- | --- | --- |
| X | X | X | O |
| ∙ | O | ∙ | X |
| ∙ | O | O | X |
| O | ∙ | ∙ | X |

* ERROR\_CASE3:

Invalid Board Too Many Os: if sum(O)>sum(X)

Invalid Board Too Many Xs: if sum(O)> sum(X)-1

* ERROR\_CASE4: Invalid Board Winner did not make last move: There is a Win-Group or intersected Win-Groups of “X” when sum(O)==sum(X).

|  |  |  |  |
| --- | --- | --- | --- |
| X | X | X | O |
| ∙ | O | ∙ | ∙ |
| ∙ | O | O | X |
| O | ∙ | ∙ | X |

* ERROR\_CASE5: Invalid Board Winner did not make last move: There is a Win-Group or intersected Win-Groups of “O” when sum(O)==sum(X)-1.

|  |  |  |  |
| --- | --- | --- | --- |
| X | ∙ | X | ∙ |
| ∙ | ∙ | O | ∙ |
| ∙ | O | ∙ | X |
| O | ∙ | ∙ | X |

1. **The algorithm of searching for Win-Group**

* Define the coordinators of every grid in the board.

x

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | ∙ | ∙ | ∙ | ∙ |
| 1 | ∙ | ∙ | ∙ | ∙ |
| 2  y | ∙ | ∙ | ∙ | ∙ |
| 3 | ∙ | ∙ | ∙ | ∙ |

* Traverse every grid in the board in the sequence of from Row0 to Row N and from Column0 to Column N to looking for all of Win-Groups.

For every grid, the possible searching directions include left, top-left, top, top-right, right, bottom-right, bottom, bottom-left. Considering program will traverse the whole board, so only four of the total eight directions should be selected. They are left, top-left, top, top-right separately, which are highlighted with red color. Other 4 directions can be covered by the selected ones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | ∙ | ∙ | ∙ | O |
| 1 | ∙ | X | ∙ | ∙ |
| 2 | ∙ | ∙ | X | ∙ |
| 3 | O | ∙ | ∙ | ∙ |

For example, the source code of searching along top-left direction of grid (x, y) is as following:

*for(i=0, count=0; ((x-i)>=0)&&((y-i)>=0)&&(mElements[x-i][y-i]==mElements[x][y]); i++){*

*if(++count==* Number-to-Win*){*

*…*

*}*

*}*

Here, “x” and “y” are coordinators. The “count” is the number of the same symbols joined along the searching direction

* Check the following error cases during searching (refer to 4.**The cases of ERRORs)**. If any error, return ERROR with error type.
  + ERROR\_CASE1
  + ERROR\_CASE2

1. **The judgment of each searching result**

After all searching, other error cases will be checked:

* ERROR\_CASE0
* ERROR\_CASE3
* ERROR\_CASE4

If there are no errors and no winner, IN\_PROGRESS and DRAW will be checked.

1. **The flow chart of method** BoardAnalyzer()



1. **The flow chart of method CheckIntersectOfWins() for checking ERROR\_CASE1 or ERROR\_CASE2**

