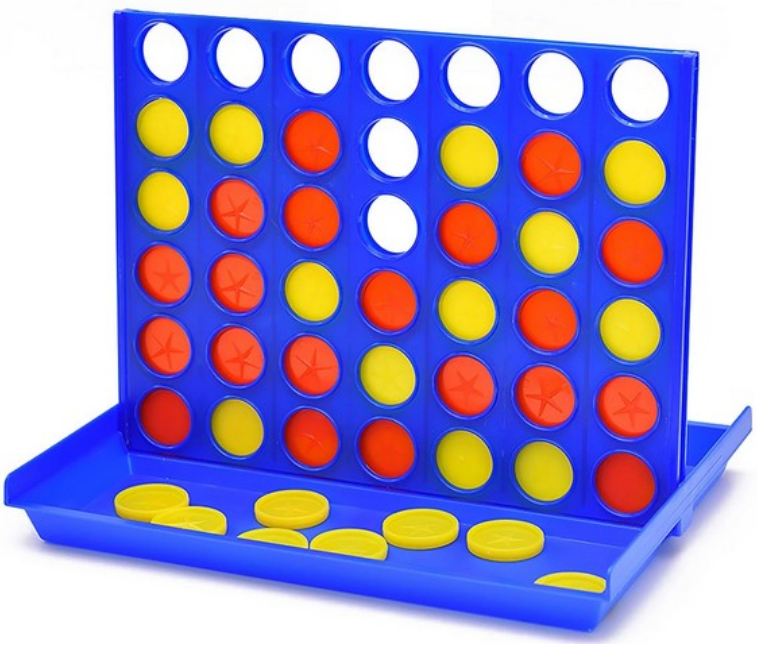
E01	Übungsprojekt Vier gewinnt Konsole	erhalten am:

„Algorithmik“ am Beispiel einen kleinen Spielprojekts: *Vier gewinnt*



Da in Prüfungen immer wieder Aufgaben auf Basis zweidimensionaler Felder / Arrays auftauchen ist es wichtig, sich mit dieser Datenstruktur intensiver zu befassen.
Dazu soll in mehreren Einheiten ein kleines *Vier gewinnt* Spiel entwickelt werden. Wir beginnen zunächst mit einer Konsolen Version und übernehmen die wesentlichen Komponenten abschließend in eine einfache GUI Version.
Natürlich ist das Internet voll mit fertigem Code und ChatGPT schüttelt jeden Algorithmus aus dem Ärmel.
Die Verwendung von fertigen Lösungen bringt Ihnen aber nichts im Hinblick auf Ihre Abschlussprüfung, deshalb appelliere ich an Ihre Programmierer:innen Ehre, die Beispiele selbst zu erstellen :-) Damit die Projekte vergleichbar bleiben sollten Sie sich grob an die hier vorgegebene Struktur halten

Einheit 01: 4x4 gewinnt mit einem Hauch MVC

Erstellen Sie ein **Konsolen-App (.Net Framework) Projekt** mit Namen **E01_Startprojekt**
Wir versuchen ein klein wenig vom Geist des **MVC Patterns** in dieses Spiel einzubauen und trennen das Modell (die interne Respresentation des momentanes Spielstandes) von der Ansicht, dem View, also der grafischen Darstellung des Modells. Als Modell soll ein zweidimensionales Array für ganze Zahlen dienen. 0 bedeutet: Feld ist leer, 1 oder 2 bedeuten Stein vom jeweiligen Spieler. Mit so einem einfachen “Zahlenmodell“ lassen sich übrigens viele ähnliche Speile erstellen, wie z.B. Schach, Dame, TicTacToe o.ä.

Wichtig:
Im Folgenden (*...und bei Anwendungsentwicklern immer!!*) sollen die Größen der beiden Dimensionen des Arrays immer über die Array Methode `GetLength(...)` ermittelt werden und **nie durch feste Zahlenwerte** (*magic values*) angegeben werden! So sind auch keine Konstanten notwendig, wie bei den primitiven C++-Feldern. So bleibt das Programm flexibel und bei Änderung der Array „Abmessungen“ läuft das restliche Programm ohne weitere Änderungen.

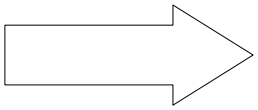
- Legen Sie in der `Main`-Methode ein Array für ganze Zahlen mit zunächst 4 Zeilen und 4 Spalten an. Das Array soll `spielbrett` heißen.

- Zur Initialisierung des Arrays schreiben Sie mit geeigneten Schleifen in jedes Element des Arrays eine 0, damit haben wir ein leeres Spielfeld.

- Für die grafische Darstellung des Spielstandes (View-Teil) geben Sie nun den Inhalt des Arrays grafisch aus. Legen Sie „oben“ direkt hinter dem Array dafür zur Ausgabe drei `Char`-Konstanten an (`ZEICHEN_LEER`, `ZEICHEN_SPIELER1`, `ZEICHEN_SPIELER2`) und weisen Sie diesen einen Punkt für ein leeres Feld, ein X für den Spielstein von Spieler1 und ein

O (Großes Oh) für den Spielstein von Spieler2 zu.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

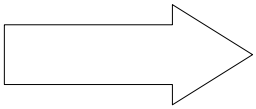


```
.   .   .   .

.   .   .   .

.   .   .   .

.   .   .   .


Drücken Sie eine beliebige
Taste . . .
```

interne Darstellung im Modell

Vom “View“ erzeugte Ansicht auf dem Bildschirm.

1	0	0	0
0	2	0	0
0	0	2	0
0	0	0	1



```
X   .   .   .

.   o   .   .

.   .   o   .

.   .   .   X


Drücken Sie eine beliebige
Taste . . .
```

- Geben Sie nun den “Spielstand“ des Arrays mit geeigneten Schleifen aus, wobei für die möglichen Zahlenwerte 0,1,2 die entsprechenden Konstanten ausgegeben werden.Hinter jedem Zeichen folgt ein Tabulatorschritt und zwischen den Zeilen erfolgen zwei Zeilenumbrüche.

- Ändern Sie die Array Dimensionen aus 5 x 5, das Programm sollte ohne weitere Änderungen laufen.
- Schreiben Sie testweise hinter den Initialisierungsschleifen in einige Elemente 1 oder 2 und kontrollieren Sie die korrekte Darstellung.
- Löschen Sie abschließend die Einträge von 1 und 2 und setzen Sie das Array wieder auf 4 x 4 zurück.