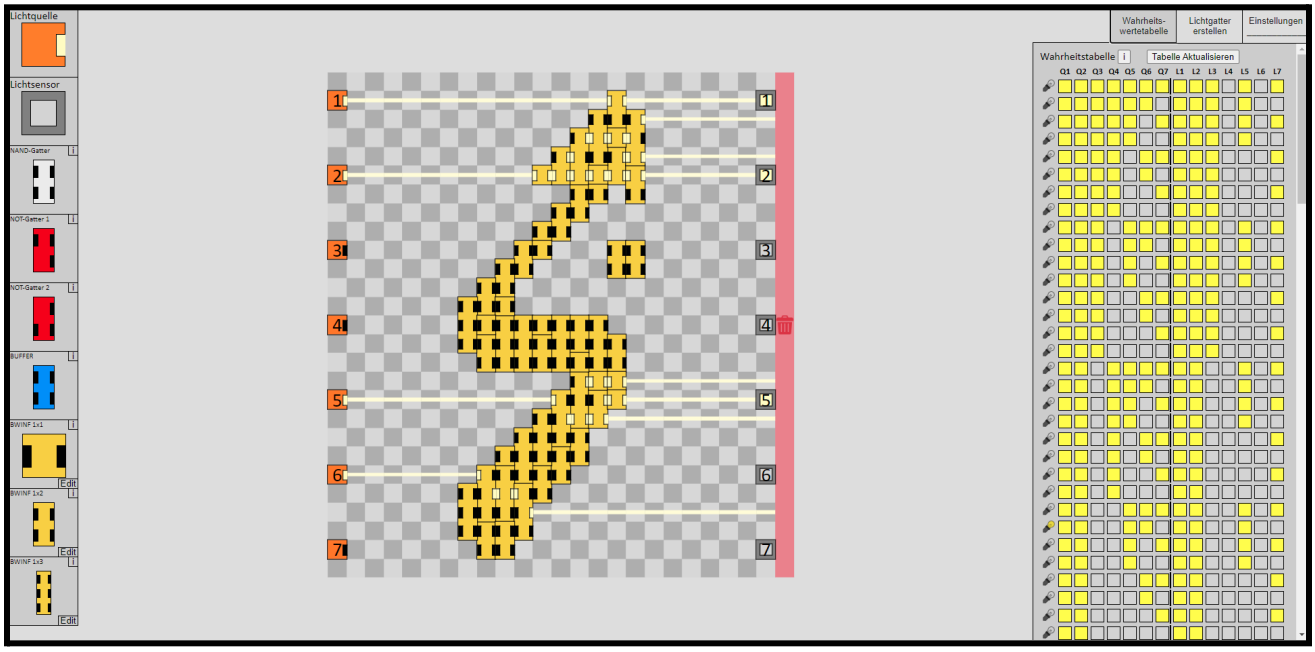
Team-ID: 00918, Teammitglieder: Bjarne Leichsenring, Nikolai Melnik, Oskar Schories

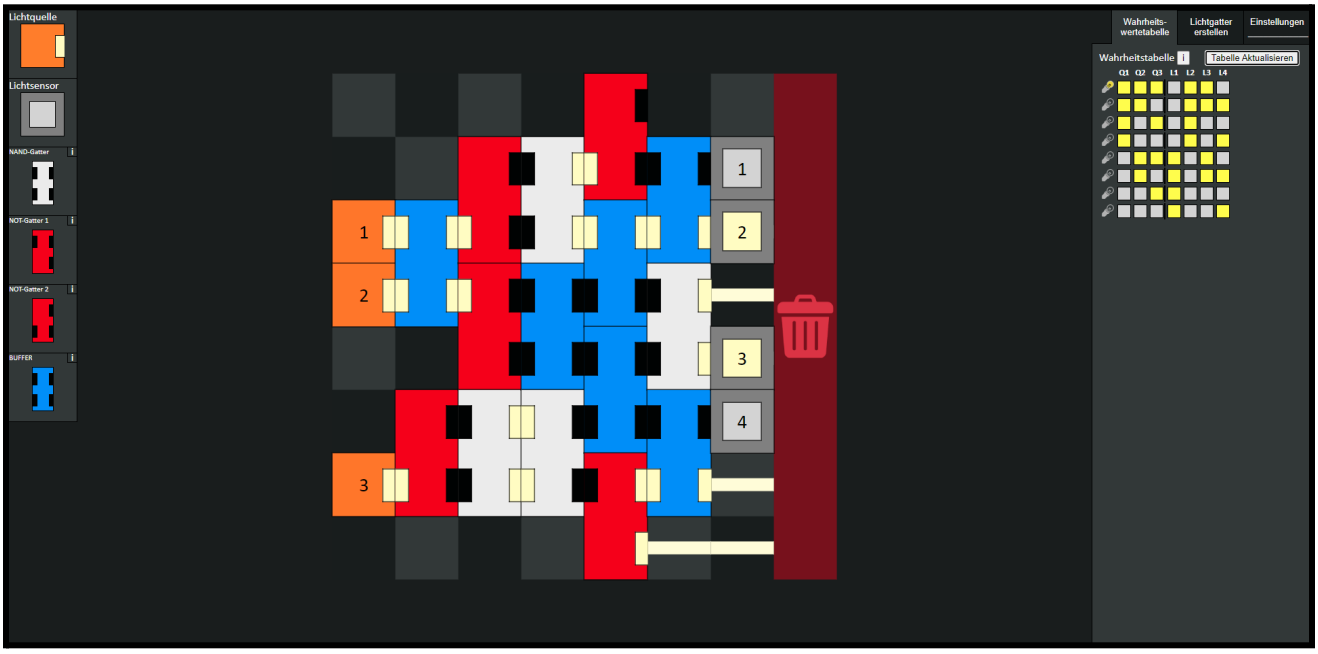
Aufgabe 4: Nandu

Anmerkungen zur Website

Die Website kann mit der Datei `NanduWebsite/nandu.html` in einem Browser abgerufen werden. Diese hat alle Basisfunktionalitäten der Aufgabenstellung, aber auch eine Wahrheitstabelle für die aufgebauten Nandufelder, eine Import- und Exportfunktion, und eine Funktion um *eigene Lichtgatter* zu erstellen. Diese können natürlich von der Python-Implementation abgelesen werden, was die Implementation komplexer gestaltet.

`Dateien: Aufgabe4/files/bwinf-nandu.txt (oben) Aufgabe4/files/nandu3.txt (unten)`





Lösungsidee

Das Programm iteriert durch alle Reihen des Nandufelds und prüft, welche Lichter `An` bzw. `Aus` sind.

In jeder Reihe betrachten wir das erste Element der Liste und vergleichen den vorderen Teil mit allen möglichen Gattern. Falls wir ein passendes gefunden haben, evaluieren wir die Licht-Inputs, welche die Outputs der vorherigen Reihe sind, um die Outputs des Gatters zu bestimmen. Daraufhin schneiden wir das Gatter aus der Nandu-Reihe aus und suchen von vorne beginnend das nächstpassende Gatter, bis wir die Reihe durchgegangen sind. Wir wiederholen diesen Prozess für alle Reihen.

Implementierung

Die Lösungsidee wurde in Python umgesetzt. Wegen der Kompatibilität zur Website, auf der man eigene Lichtgatter erstellen kann, müssen wir diese auch importieren können. Deswegen importiert die `LightGate`-Klasse ein JSON, da die Website auch so die selbst-erstellen Gatter exportiert. Input-files ohne eigene Gatter (wie die Beispiele auf der BWINF-Seite) können ebenfalls evaluiert werden.

Folgende Klassen wurden implementiert:

```
LightGate(id: Union[list[str], int], json_data: dict)
```

Klasse zur Representation eines Lichtgatters. `json_data` enthält alle Daten für dieses Gatter, von dessen Länge bis hin zu der Wahrheitstabelle, welche alle auf der Website angepasst werden können. Im Programm werden alle Gatter definiert, welche in der Aufgabenstellung vorgegeben sind.

Die `id` ist entweder eine Liste, welche das Gatter identifizieren lässt, oder ein Integer, welcher in der Initialisation die echte `id` auch als eine Liste von Integers speichert. Mit dieser identifiziert das Programm die Gatter.

Folgende Methoden wurden implementiert:

```
evaluateNanduRow(row: NanduRow, input_values: LightValueRow, gates:
list[LightGate], source_values: list[bool], sensor_values: list[bool]) ->
LightValueRow
```

Evaluiert eine Nandureihe nach der vorherig beschriebenen Logik.

```
evaluateNanduGrid(grid: list[NanduRow], gates: list[LightGate],
source_values: list[bool], sensor_values: list[bool]) -> list[bool]
```

Evaluiert ein Nandufeld mit den vorgegebenen Lichtquellen-angaben, indem es mit *`evaluateNanduRow(..)`* durch jede Reihe iteriert.

`importNanduGrid(filename: str) -> tuple[list[NanduRow], list[LightGate]])`
Importiert ein Nandufeld und dessen Lichtgatter, welche dort definiert wurden.

`solveNandu(filename: str, output: str)`
Löst ein Nandufeld aus einer Datei und gibt dessen Wahrheitstabelle als CSV-Datei aus.

Beispiele:

Ausgabe von nandu1.txt

```
Q1  ; Q2  ; L1  ; L2
An  ; An  ; Aus ; Aus
An  ; Aus ; An  ; An
Aus ; An  ; An  ; An
Aus ; Aus ; An  ; An
```

Ausgabe von nandu2.txt

```
Q1  ; Q2  ; L1  ; L2
An  ; An  ; An  ; Aus
An  ; Aus ; Aus ; An
Aus ; An  ; Aus ; An
Aus ; Aus ; Aus ; An
```

Ausgabe von nandu3.txt

```
Q1  ; Q2  ; Q3  ; L1  ; L2  ; L3  ; L4
An  ; An  ; An  ; Aus ; An  ; An  ; Aus
An  ; An  ; Aus ; Aus ; An  ; An  ; An
An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; An  ; Aus ; An
Aus ; An  ; An  ; An  ; Aus ; An  ; Aus
Aus ; An  ; Aus ; An  ; Aus ; An  ; An
Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An
```

Ausgabe von nandu4.txt

```
Q1  ; Q2  ; Q3  ; Q4  ; L1  ; L2
An  ; An  ; An  ; An  ; Aus ; Aus
An  ; An  ; An  ; Aus ; Aus ; An
An  ; An  ; Aus ; An  ; Aus ; Aus
An  ; An  ; Aus ; Aus ; Aus ; Aus
An  ; Aus ; An  ; An  ; Aus ; Aus
An  ; Aus ; An  ; Aus ; Aus ; An
An  ; Aus ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; An  ; An  ; An  ; Aus
Aus ; An  ; An  ; Aus ; An  ; An
Aus ; An  ; Aus ; An  ; An  ; Aus
Aus ; An  ; Aus ; Aus ; An  ; Aus
Aus ; Aus ; An  ; An  ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; Aus ; An
```

```
Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; Aus ; Aus
```

Ausgabe von nandu5.txt

```
Q1  ; Q2  ; Q3  ; Q4  ; Q5  ; Q6  ; L1  ; L2  ; L3  ; L4  ; L5
An  ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
An  ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus
An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus
An  ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
An  ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus
An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus
An  ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
An  ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus
An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus
An  ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An
An  ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus
An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus
Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
```

```
Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An
Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus
Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus
```

Eingabe von bwinf-nandu.txt

```
27 24
X  Q1 X  X  X  Q2 X  X  X  Q3 X  X  X  Q4 X  X  X  Q5 X  X  X  Q6 X  X  X  Q7 X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  M3 M3 M3 X  X  X  X  X  X  X  M3 M3 M3 X  X
X  X  X  X  X  X  X  X  X  X  X  M2 M2 M1 M2 M2 X  X  X  X  X  M3 M3 M3 M2 M2 X
X  X  X  X  X  X  X  X  X  X  M2 M2 M1 M3 M3 M3 X  X  X  X  M1 M3 M3 M3 M2 M2 X
X  X  X  X  X  X  X  X  X  M2 M2 X  X  M3 M3 M3 X  X  X  M3 M3 M3 M3 M3 M3 X  X
X  X  X  X  X  M1 X  X  M1 M1 X  X  X  M1 M2 M2 X  X  M3 M3 M3 M2 M2 X  X  X  X
X  X  X  X  M2 M2 X  M2 M2 X  X  X  X  M3 M3 M3 X  M3 M3 M3 M2 M2 X  X  X  X  X
X  X  X  M3 M3 M3 M2 M2 X  X  X  X  X  M2 M2 M1 M3 M3 M3 M2 M2 X  X  X  X  X  X
X  X  M3 M3 M3 M1 M1 X  X  X  X  X  X  M3 M3 M3 M1 M2 M2 X  X  X  X  X  X  X  X
X  M2 M2 M1 M2 M2 X  X  X  M2 M2 X  X  X  M2 M2 M1 M1 X  X  X  X  X  X  X  X  X
X  X  M3 M3 M3 M2 M2 X  X  M2 M2 X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
X  L1 X  X  X  L2 X  X  X  L3 X  X  X  L4 X  X  X  L5 X  X  X  L6 X  X  X  L7 X
```

{"name":"BWINF 1x1","color":"#F3D214","realHeight":1,"inOut":[["I1"],["O1"]],"rules":[[true],[false]]}
{"name":"BWINF
1x2","color":"#F3D214","realHeight":2,"inOut":[["I1","I2"],["O1","O2"]],"rules":[[false,true],[true,false],[
true,true],[false,false]]}
{"name":"BWINF
1x3","color":"#F3D214","realHeight":3,"inOut":[["I1","I2","I3"],["O1","O2","O3"]],"rules":[[true,false,true]

```
,[false,true,false],[true,false,true],[false,true,false],[true,false,true],[false,true,false],[true,false,tr
ue],[false,false,false]]}
```

Ausgabe von bwinf-nandu.txt

```
 Q1  ; Q2  ; Q3  ; Q4  ; Q5  ; Q6  ; Q7  ; L1  ; L2  ; L3  ; L4  ; L5  ; L6  ; L7
 An  ; An  ; An  ; An  ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An
 An  ; An  ; An  ; An  ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus
 An  ; An  ; An  ; An  ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An
 An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus
 An  ; An  ; An  ; An  ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An
 An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An
 An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; An  ; Aus ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An
 An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus
 An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An
 An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus
 An  ; An  ; An  ; Aus ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An
 An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An
 An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; Aus ; An  ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An
 An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus
 An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An
 An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus
 An  ; An  ; Aus ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An
 An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An
 An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; Aus ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An
 An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus
 An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An
 An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus
 An  ; An  ; Aus ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An
 An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
 An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An
 An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
 An  ; Aus ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An
 An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus
 An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An
 An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus
 An  ; Aus ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An
 An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An
 An  ; Aus ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An
 An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus
 An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An
 An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus
 An  ; Aus ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An
 An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An
 An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
 An  ; Aus ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An
```

```
An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An
An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An
An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An
An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An
An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An
An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An
An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An
An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; An  ; An  ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An
Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An
Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An
Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An
Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; An  ; Aus ; An  ; An  ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An
Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An
Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus
Aus ; An  ; An  ; Aus ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An
Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An
Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; Aus ; An  ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An
Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An
Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; Aus ; An  ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An
Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An
Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An
Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An
Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; An  ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An
Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An
Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; An  ; An  ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An
Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; An  ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An
Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; An  ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An
Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An
Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; An  ; An  ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An
```

```
Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; An
Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An
Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; An
Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An
Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An
Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An
Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; Aus ; An  ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An
Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; An  ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An
Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; An
Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; Aus ; An  ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An
Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An  ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; An
Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus ; Aus
```

Quelltext

Die Kommentare wurden aus Gewohnheit in Englisch geschrieben.

```python
from itertools import product
import json
from typing import Union # cannot use list[str] | int due to the feature being to recent (i think it was implemented 3.9ish?)

# this code is fully compactable with the website. that means, it can use any table that has been generated from the website,
# and is able to read and process the custom gates that the user has created.

# you should totally check out the website my teammate spent the whole autumn holidays making it :D


# the types below are purely for readability

# a list of booleans describing what outputs should be on or off
TruthtableValue = list[bool]

# a list of booleans describing what lights are shining through the specific indexes
LightValueRow = list[bool]

# a list of strings defining what gates the row is made of
NanduRow = list[str]

# a list describing the connections, and whether they exist at a certain index
```

```python
# a connection list for inputs could look like ["I1",None,"I2",None,None,"I3","I4"]
# a connection list for outputs would have O's instead of I's
# we have to work with this due to the implementation of the website having a feature to turn off inputs/outputs at
certain places
ConnectionList = list[str]

class LightGate:

    def __init__(self, id: Union[list[str], int] , json_data: dict):


        # custom markers are always saved as a sequence of M<int>'s, so we can automate that
        if isinstance(id, int):
            self.id: list[str] = [f"M{id}"] * json_data["realHeight"]
        else:
            self.id: list[str] = id

        # yes, there is a "realWidth", however, it is unused in the current version of the website
        # json_data.inOut also contains a connection list for all 4 sides of a gate, it is however also unused
        self.size: int = json_data["realHeight"]
        self.inputs: ConnectionList = json_data["inOut"][0]
        self.outputs: ConnectionList = json_data["inOut"][1]

        self.truth_table: list[TruthtableValue] = json_data["rules"][::-1]


    def process(self, input_values: LightValueRow):

        # Initial output values; blocked by the gate
        output_values: LightValueRow = [False] * len(input_values)

        # the thing has no outpots or no inputs, no light can be let through
        if not any(self.outputs) or not any(self.inputs):
            return output_values

        rule_index = 0
        input_count = sum(1 if k else 0 for k in self.inputs)
        for input, value in zip(self.inputs, input_values):
            if input != None and value == True:
                # inputs are named in the form of "I<int>", we can just extract the integer and gets it's
identification
                input_id: int = int(input[1:])
                rule_index += 2 ** (input_count - input_id)

        output_rule: TruthtableValue = self.truth_table[rule_index]

        for index, output in enumerate(self.outputs):
            if output != None:
                output_id: int = int(output[1:])
                output_values[index] = output_rule[output_id - 1]


        return output_values
```

```
# default gates that are defined by the task. we also define an empty one for X so we do not need to add an edge case
for it later

BlueMarker = LightGate(
    id=["B", "B"],
    json_data={
        "realHeight" : 2,
        "inOut" : [
            ["I1", "I2"],
            ["O1", "O2"]
        ],
        "rules" : [
            # O1    O2         I1   I2
            [True, True],  #  true, true
            [True, False], #  true, false
            [False, True], #  false, true
            [False, False] #  false, false
        ]
    }
)

WhiteMarker = LightGate(
    id=["W","W"],
    json_data={
        "realHeight":2,
        "inOut": [
            ["I1","I2"],
            ["O1","O2"]
        ],
        "rules" : [
            # O1     O2       I1  I2
            [False, False], # true, true
            [True, True],   # true, false
            [True, True],   # false, true
            [True, True]    # false ,false
        ]
    }
)

RedMarker = LightGate(
    id=["r","R"],
    json_data={
        "realHeight":2,
        "inOut" : [
            [None,"I1"],
            ["O1","O2"]
        ],
        "rules" : [
            #  O1     O2        I1
            [False, False], # true
            [True, True]    # false
        ]
```

```
    }
)

ReflectedRedMarker = LightGate(
    id=["R","r"],
    json_data={
        "realHeight":2,
        "inOut" : [
            ["I1",None],
            ["O1","O2"]
        ],
        "rules" : [
            #  O1     O2        I1
            [False, False],  # true
            [True, True]     # false
        ]
    }
)

empty = LightGate(
    id=["X"],
    json_data={
        "realHeight":1,
        "inOut" : [
            ["I1"],
            ["O1"]
        ],
        "rules" : [
            # O1       I1
            [True], # true
            [False] # false
        ]
    }
)

DEFAULT_GATE_LIST = [
    BlueMarker,
    WhiteMarker,
    RedMarker,
    ReflectedRedMarker,
    empty
]

def evaluateNanduRow(row: NanduRow, input_values: LightValueRow, gates: list[LightGate], source_values: list[bool],
sensor_values: list[bool]) -> LightValueRow:

    # the inputs that are processed are going to land here
    output_values: LightValueRow = []

    while row:

        # we do not need a specific edge case for when an "X" is used; A gate for that should be contained in the gate
list
```

```
        # however, we need one for sources and sensors (quelle/lichtsensor)

        # the light sensor lets light through, which it is not confirmed to do in the task description;
        # however, it does not conflict with the example files and is replicating the websites functionality

        if row[0][0] == "Q" or row[0][0] == "L":
            id: int = int(row[0][1:])
            id -= 1 # the id's start counting from one, list indexes start counting from 0
            if row[0][0] == "Q":
                output_values.append(source_values[id])
            else:
                sensor_values[id] = input_values[0]
                output_values.append(input_values[0])

            # remove the first element from the row and input_values
            row = row[1:]
            input_values = input_values[1:]

        else:

            # we iterate through all the possible gates, and compare the slice that has the size of the gate to the
gate itself
            for gate in gates:
                compare_gate = row[:gate.size]
                if gate.id == compare_gate:

                    process_input = input_values[:gate.size]

                    processed_input = gate.process(process_input)
                    output_values.extend(processed_input)

                    # we cut out the rest to process it in the next iteration
                    row = row[gate.size:]
                    input_values = input_values[gate.size:]

                    break
            else:
                raise Exception(f"Couldn't process further from the following row expression: {row}")

    return output_values

def evaluateNanduGrid(grid: list[NanduRow], gates: list[LightGate], source_values: list[bool], sensor_values:
list[bool]) -> list[bool]:
    # simply calls evaluateNanduRow on every row through a grid.

    # initially everything is off
    output_values = [False] * len(grid[0])

    for row in grid:

        # source_values and sensor_values are edited internally, so we do not need to reassign them
        output_values = evaluateNanduRow(
            row = row,
```

```
            input_values = output_values, # input values to the next row are the output values from the previous rows
            gates = gates,
            source_values = source_values,
            sensor_values = sensor_values
        )
        #print(" ".join(k.ljust(2) for k in row), "  ", " ".join("##" if k else ".." for k in output_values))
        #print()

    return sensor_values

def importNanduGrid(filename: str) -> tuple[list[NanduRow], list[LightGate]]:
    with open(filename, "r") as f:
        data: str = f.read()

    lines = data.split("\n")

    width, height = map(int, lines[0].split())

    grid: list[NanduRow] = [row.split() for row in lines[1:height+1]]

    gates: list[LightGate] = DEFAULT_GATE_LIST

    # if height+2 is out of range, we get an empty iterator, so it's all fine
    for index, line in enumerate(lines[height+2:]):
        try:
            gate = LightGate(
                id = index + 1, # id's should start from 1, indexes start from 0
                json_data = json.loads(line)
            )
            gates.append(gate)
        except json.decoder.JSONDecodeError:
            raise Exception(f"Couldn't extract gate {index} from file: {line}")

    return grid, gates

def solveNandu(filename: str, output: str):

    grid, gates = importNanduGrid(filename)

    # we assume that the sources/sensors are assigned to the numbers 1..n in the file, so we can just count them
    count_sources: int = 0
    count_sensors: int = 0

    for row in grid:
        for block in row:
            if block[0] == "Q":
                count_sources += 1
            if block[0] == "L":
                count_sensors += 1

    truth_table = []

    # itertools.product generates all the variants for the truth table we need
```

```
    for light_variant in product((True, False),repeat=count_sources):
        truth_table.append((light_variant, evaluateNanduGrid(
            grid = grid,
            gates = gates,
            source_values = list(light_variant),
            sensor_values = [False] * count_sensors # all sensors should initially be off
        )))

    # exported als CSV. we use semicolons as separators, shouldn't be a problem as google sheets and excel can work
with those
    with open(output, "w", encoding="utf-8") as f:
        # creates the header that describes what column belong to what input/output
        header = ";".join(f" Q{n+1}  " for n in range(count_sources)) + ";"
        header += ";".join(f" L{n+1}  " for n in range(count_sensors))
        f.write(header + "\n")
        for source_values, sensor_values in truth_table:
            f.write(";".join(" An  " if val else " Aus " for val in (list(source_values) + sensor_values)) + "\n")

if __name__ == "__main__":
    print("WARNING: The CSV Files in nandu/output/ use semicolons as separators.")
    for i in range(5):
        solveNandu(f"files/nandu{i+1}.txt",f"output/nandu{i+1}.csv")
    solveNandu("files/bwinf-nandu.txt", "output/bwinf-nandu.csv")
```