# ASSIGNMENT 1
# SWIFT PRACTICING: PUNCHED CARDS[1]

A secret team of programmers is plotting to disrupt the programming language landscape and bring punched cards back by introducing a new language called Punched Card Python that lets people code in Python using punched cards! Like good disrupters, they are going to launch a viral campaign to promote their new language before even having the design for a prototype. For the campaign, they want to draw punched cards of different sizes in ASCII art.

The ASCII art of a punched card they want to draw is similar to an **R**×**C** matrix without the top-left cell. That means, it has (**R**·**C**)−1 cells in total. Each cell is drawn in ASCII art as a period (.) surrounded by dashes (-) above and below, pipes (|) to the left and right, and plus signs (+) for each corner. Adjacent cells share the common characters in the border. Periods (.) are used to align the cells in the top row.

For example, the following is a punched card with **R**=3 rows and **C**=4 columns:

```
..+-+-+-+
..|.|.|.|
+-+-+-+-+
|.|.|.|.|
+-+-+-+-+
|.|.|.|.|
+-+-+-+-+
```

There are more examples with other sizes in the samples below. Given the integers **R** and **C** describing the size of a punched card, print the ASCII art drawing of it as described above.

---

1 This question appeared in the Qualification Round of Google Code Jam 2022

## Input

The first line of the input gives the number of test cases, **T**. **T** lines follow, each describing a different test case with two integers **R** and **C**: the number of rows and columns of the punched card that must be drawn.

## Output

For each test case, output one line containing `Case #x:`, where *x* is the test case number (starting from 1). Then, output (2·**R**)+1 additional lines with the ASCII art drawing of a punched card with **R** rows and **C** columns.

## Limits

Time limit: 5 seconds.

Memory limit: 1 GB.

## Constraints

1≤**T**≤20.

2≤**R**≤10.

2≤**C**≤10.

## Sample Input

```
3
3 4
2 2
2 3
```

## Sample Output

```
Case #1:
..+-+-+-+
..|.|.|.|
+-+-+-+-+
|.|.|.|.|
+-+-+-+-+
|.|.|.|.|
+-+-+-+-+
Case #2:
..+-+
..|.|
+-+-+
|.|.|
```

```
+-+-+
Case #3:
..+-+-+
..|.|.|
+-+-+-+
|.|.|.|
+-+-+-+
```

## DELIVERABLE

Write two pieces of Code.

1. The first code should prepare the sample input file. Here, 20 sets of **R** and **C** pairs should be generated randomly and saved on file.
2. The second code should take the input file and generate the corresponding output sequence (as shown in example output).
3. **Both codes should be done in Swift. Upload the two codes + your output file (3 files needed….. Not 1, not 2, but 3 files).**
4. **Note: Do not copy code. Will use similarity check to deduct marks automatically.**