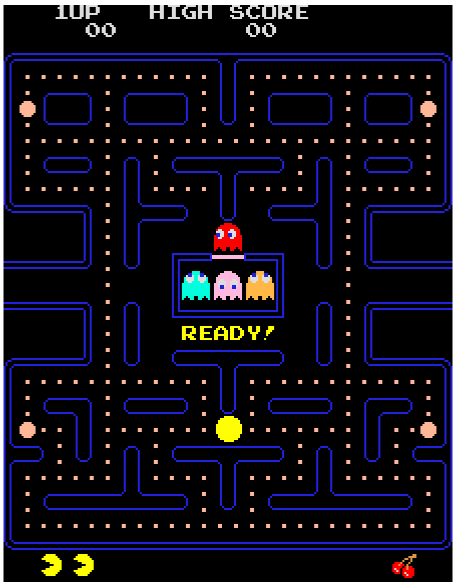
Pac-Man

Pac-Man is an arcade maze chase game, originated from Japan, in 1980. It is considered one of the classics of the medium and an icon of 1980s popular culture.

The player navigates Pac-Man through a maze containing dots, known as Pac-Dots, and four multi-colored ghosts: Blinky, Pinky, Inky, and Clyde. The goal of the game is to accumulate as many points as possible by collecting the dots and eating ghosts. When all of the dots in a stage is eaten, that stage is completed and the player will advance to the next one.



Waypoint 1: Load Pac-Man Map

We have recreated the original map of Pac Man and stored it in the directory `./map` of this project. This is a text file, encoded in UTF-8, representing the Pac-Man's maze with the Unicode characters "═", "║", "╔", "╗", "╚", and "╝". The Unicode character "·" represents a Pac-Dot. Space characters represent areas where Pac-Man cannot move on.

Write a function `load_map` that takes the file path name of a Pac-Man map and returns an array of lines.

For example:

```
>>> pacman_map = load_map('./map/level1.amap')
>>> for line in pacman_map:
...     print(line)
...
╔════════════╗╔════════════╗
║············║║············║
║·╔══╗·╔═══╗·║║·╔═══╗·╔══╗·║
║·║  ║·║   ║·║║·║   ║·║  ║·║
║·╚══╝·╚═══╝·╚╝·╚═══╝·╚══╝·║
║··························║
║·╔══╗·╔╗·╔══════╗·╔╗·╔══╗·║
║·╚══╝·║║·╚══╗╔══╝·║║·╚══╝·║
```

```
║······║║····║║····║║······║
╚════╗·║╚══╗·║║·╔══╝║·╔════╝
     ║·║╔══╝·╚╝·╚══╗║·║
     ║·║║··········║║·║
     ║·║║·╔══  ══╗·║║·║
═════╝·╚╝·║      ║·╚╝·╚═════
··········║      ║··········
═════╗·╔╗·╚══════╝·╔╗·╔═════
     ║·║║··········║║·║
     ║·║║·╔══════╗·║║·║
╔════╝·╚╝·╚══╗╔══╝·╚╝·╚════╗
║············║║············║
║·╔══╗·╔═══╗·║║·╔═══╗·╔══╗·║
║·╚═╗║·╚═══╝·╚╝·╚═══╝·║╔═╝·║
║···║║················║║···║
╚═╗·║║·╔╗·╔══════╗·╔╗·║║·╔═╝
╔═╝·╚╝·║║·╚══╗╔══╝·║║·╚╝·╚═╗
║······║║····║║····║║······║
║·╔════╝╚══╗·║║·╔══╝╚════╗·║
║·╚════════╝·╚╝·╚════════╝·║
║··························║
╚══════════════════════════╝
```

Waypoint 2: Simplify Pac-Man Map

We need to simplify the representation of this map with less characters. We want to simplify that data structure of the map so that we can compress the file map; it will consume less space on disk.

Write a function `simplify_map` that takes a Pac-Man map and returns a simplified version where:

- the Unicode characters "`═`", "`║`", "`╔`", "`╗`", "`╚`", and "`╝`" are replaced with the ASCII character "`*`";
- the Unicode character "`·`" is replaced with the ASCII character "`.`" (dot).

For example:

```
>>> pacman_map = load_map('./map/level1.amap')
>>> simplified_map = simplify_map(pacman_map)
>>> for line in simplified_map:
...     print(line)
...
****************************
*............**............*
*.****.*****.**.*****.****.*
*.*  *.*   *.**.*   *.*  *.*
*.****.*****.**.*****.****.*
*..........................*
*.****.**.********.**.****.*
*.****.**.********.**.****.*
*......**....**....**......*
******.*****.**.*****.******
     *.*****.**.*****.*
     *.**..........**.*
     *.**.***  ***.**.*
******.**.*      *.**.******
..........*      *..........
******.**.********.**.******
     *.**..........**.*
     *.**.********.**.*
******.**.********.**.******
*............**............*
*.****.*****.**.*****.****.*
*.****.*****.**.*****.****.*
*...**................**...*
***.**.**.********.**.**.***
***.**.**.********.**.**.***
*......**....**....**......*
*.**********.**.**********.*
*.**********.**.**********.*
*..........................*
****************************
```

Waypoint 3: Convert Simplified Pac-Man Map to Original

We will only store on the disk, with the file extension `.map`, simplified versions of the Pac-Man maps. These files will be later compressed in a binary format.

We need however to convert a simplified Pac-Man map to its original version, which is more human-readable, i.e., that can be naturally visualized by humans.

Write a function `prettify_map` that takes a Pac-Man simplified map and returns a human-readable version.

For example:

```
>>> pacman_map = load_map('./map/level1.map')
>>> prettified_map = prettify_map(pacman_map)
>>> for line in prettified_map:
...     print(line)
...
╔════════════╗╔════════════╗
║............║║............║
║.╔══╗.╔═══╗.║║.╔═══╗.╔══╗.║
║.║  ║.║   ║.║║.║   ║.║  ║.║
║.╚══╝.╚═══╝.╚╝.╚═══╝.╚══╝.║
║..........................║
║.╔══╗.╔╗.╔══════╗.╔╗.╔══╗.║
║.╚══╝.║║.╚══╗╔══╝.║║.╚══╝.║
║......║║....║║....║║......║
╚════╗.║╚══╗.║║.╔══╝║.╔════╝
     ║.║╔══╝.╚╝.╚══╗║.║
     ║.║║..........║║.║
     ║.║║.╔══  ══╗.║║.║
═════╝.╚╝.║      ║.╚╝.╚═════
```

```
..........║      ║..........
═════╗.╔╗.╚══════╝.╔╗.╔═════
     ║.║║..........║║.║     
     ║.║║.╔══════╗.║║.║     
╔════╝.╚╝.╚══╗╔══╝.╚╝.╚════╗
║............║║............║
║.╔══╗.╔═══╗.║║.╔═══╗.╔══╗.║
║.╚═╗║.╚═══╝.╚╝.╚═══╝.║╔═╝.║
║...║║................║║...║
╚═╗.║║.╔╗.╔══════╗.╔╗.║║.╔═╝
╔═╝.╚╝.║║.╚══╗╔══╝.║║.╚╝.╚═╗
║......║║....║║....║║......║
║.╔════╝╚══╗.║║.╔══╝╚════╗.║
║.╚════════╝.╚╝.╚════════╝.║
║..........................║
╚══════════════════════════╝
```

Waypoint 4: RLE Compression

Run-Length Encoding (RLE) is a simple form of data compression that reduces a sequence of a same character with a number representing the length of this sequence, followed with this character.

For example, the following line composed of 28 characters:

```
******.**.*      *.**.******
```

could be translated, using RLE, to the following line of 22 characters:

```
6*1.2*1.1*6 1*1.2*1.6*
```

We reduce the number of characters representing this line. The line is 6 characters shorter. We save 21% of disk space for this line.

Note: RLE is not useful with files that don't have long sequences as it could greatly increase the file size.

Compress Initial Map

Write a function `compress_map_with_rle` that takes a Pac-Man map and returns the compressed version of this map using RLE.

```
>>> pacman_map = load_map('./map/level1.map')
>>> compressed_map = compress_map_with_rle(pacman_map)
>>> for line in compressed_map:
...     print(line)
...
28*
1*12.2*12.1*
```

```
1*1.4*1.5*1.2*1.5*1.4*1.1*
1*1.1*2 1*1.1*3 1*1.2*1.1*3 1*1.1*2 1*1.1*
1*1.4*1.5*1.2*1.5*1.4*1.1*
1*26.1*
1*1.4*1.2*1.8*1.2*1.4*1.1*
1*1.4*1.2*1.8*1.2*1.4*1.1*
1*6.2*4.2*4.2*6.1*
6*1.5*1.2*1.5*1.6*
5 1*1.5*1.2*1.5*1.1*
5 1*1.2*10.2*1.1*
5 1*1.2*1.3*2 3*1.2*1.1*
6*1.2*1.1*6 1*1.2*1.6*
10.1*6 1*10.
6*1.2*1.8*1.2*1.6*
5 1*1.2*10.2*1.1*
5 1*1.2*1.8*1.2*1.1*
6*1.2*1.8*1.2*1.6*
1*12.2*12.1*
1*1.4*1.5*1.2*1.5*1.4*1.1*
1*1.4*1.5*1.2*1.5*1.4*1.1*
1*3.2*16.2*3.1*
3*1.2*1.2*1.8*1.2*1.2*1.3*
3*1.2*1.2*1.8*1.2*1.2*1.3*
1*6.2*4.2*4.2*6.1*
1*1.10*1.2*1.10*1.1*
1*1.10*1.2*1.10*1.1*
1*26.1*
28*
```

We can determine the data compression ratio of the RLE algorithm, that is the reduction in data-representation size produced by this algorithm:

```
>>> pacman_map_size = sum([len(line) for line in pacman_map])
>>> pacman_map_size
815
>>> compressed_map_size = sum([len(line) for line in compressed_map])
>>> compressed_map_size
571
```

The compression ratio is `815 / 571 = 1.4`, notated as an explicit ratio, `7:5`. The space saving is `1 - (571 / 815)`, nearly 30%.

Uncompress Packed Map

Write a function `uncompress_map_with_rle` that takes a *simplified Pac-Man map that was encoded with the Run-Length Encoding (RLE) algorithm, and returns the decoded *simplified Pac-Man map.

```
>>> compressed_map = load_map('./map/level1.rle')
>>> prettified_map = prettify_map(uncompress_map_with_rle(compressed_map))
>>> for line in prettified_map:
...     print(line)
...
╔════════════╗╔════════════╗
║............║║............║
║.╔══╗.╔═══╗.║║.╔═══╗.╔══╗.║
║.║  ║.║   ║.║║.║   ║.║  ║.║
║.╚══╝.╚═══╝.╚╝.╚═══╝.╚══╝.║
║..........................║
║.╔══╗.╔╗.╔══════╗.╔╗.╔══╗.║
║.╚══╝.║║.╚══╗╔══╝.║║.╚══╝.║
║......║║....║║....║║......║
╚════╗.║╚══╗.║║.╔══╝║.╔════╝
     ║.║╔══╝.╚╝.╚══╗║.║
     ║.║║..........║║.║
     ║.║║.╔══  ══╗.║║.║
═════╝.╚╝.║      ║.╚╝.╚═════
..........║      ║..........
═════╗.╔╗.╚══════╝.╔╗.╔═════
     ║.║║..........║║.║
     ║.║║.╔══════╗.║║.║
╔════╝.╚╝.╚══╗╔══╝.╚╝.╚════╗
║............║║............║
║.╔══╗.╔═══╗.║║.╔═══╗.╔══╗.║
║.╚═╗║.╚═══╝.╚╝.╚═══╝.║╔═╝.║
║...║║................║║...║
╚═╗.║║.╔╗.╔══════╗.╔╗.║║.╔═╝
╔═╝.╚╝.║║.╚══╗╔══╝.║║.╚╝.╚═╗
║......║║....║║....║║......║
║.╔════╝╚══╗.║║.╔══╝╚════╗.║
║.╚════════╝.╚╝.╚════════╝.║
║..........................║
╚══════════════════════════╝
```