Lab session 6: Imperative Programming

Problem 1: Short circuit

Consider 4 locations, numbered 0, 1, 2, and 3. The following table shows the distances between the locations.

	0	1	2	3
0	0	5	6	2
1	5	0	1	3
2	6	1	0	1
3	2	3	1	0

There exist $4! = 24$ circuits that visit each location exactly once. However these circuits do not all have the same length. For example, the following two circuits differ in length:

- $0 \to 1 \to 2 \to 3$ and back to 0 with length $5 + 1 + 1 + 2 = 9$.
- $2 \to 3 \to 1 \to 0$ and back to 2 with length $1 + 3 + 5 + 6 = 15$.

For this simple example, it is not hard to see that the shortest circuit length is 9. However, if we increase the number of locations, it gets harder to determine the length of the optimal circuit.

The first line of the input of this problem contains a positive integer `n`: the number of locations. You may assume that this number is at most 10. The next `n` lines contain the distance matrix. Because the distances are symmetric, only half the matrix is given. The output of the program must be the length of the shortest circuit that visits all locations.

Example 1:

input:

```
4
0
5 0
6 1 0
2 3 1 0
```

output:

```
9
```

Example 2:

input:

```
3
0
9 0
3 8 0
```

output:

```
20
```

Example 3:

input:

```
6
0
1 0
5 9 0
8 6 7 0
2 9 4 6 0
8 8 1 5 7 0
```

output:

```
19
```

Problem 2: Statistics

In this problem you will compute some statistical properties of a series of numbers: the *mean*, the *median*, and the *mode*. Recall their definitions:

- The *mean* of a series of numbers is the average. It is calculated by summing up the series and dividing by the length of the series.
- The *median* of a series of numbers is the number that appears in the middle of the list when arranged from smallest to largest.
- The *mode* of a series of numbers is the value that appears most often. Note that the mode may not exist.

The input for this problem consists of an odd positive integer `n`: the length of the series. Next a series of `n` non-negative integers is given. Note that the values are chosen such that you can safely sum them up in an `int` without overflow. The program should print the mean, median, and mode of the series. The mean should be rounded to two decimal places after the decimal dot. [Important note: To compute the mean of a series, it is important to use the type `double`, and not `floats` (because of their limited accuracy).]

Example 1:

input:
```
5
1 2 3 4 1
```
output:
```
mean: 2.20
median: 2
mode: 1
```

Example 2:

input:
```
5
1 4 2 3 5
```
output:
```
mean: 3.00
median: 3
mode: NONE
```

Example 3:

input:
```
3
43 40 43
```
output:
```
mean: 42.00
median: 43
mode: 43
```

Problem 3: String Conversion

The input for this problem consists of two strings of characters: `str1` and `str2`. The task is to compute the minimum number of operations needed to convert `str1` into `str2`. There are three operations available which can only be performed on `str1`:

- `Insert`: insert a character at any position
- `Delete`: delete a character at any position
- `Replace`: replace a character by another character

For example, to convert `str1="saturday"` into `str2="sunday"` the optimal conversion is to `delete` the letters `a` and `t`, and to `replace` the letter `r` by the letter `n`. Therefore, the minimum number of operations for this example is 3.

The input consists of two strings, which are terminated by a newline (i.e. `'\n'`) character. You may assume that the sum of the lengths of the two strings does not exceed 25. The output

must be the minimal number of operations to convert the first string into the second.

Example 1:

input:

```
saturday
sunday
```

output:

```
3
```

Example 2:

input:

```
fish
chips
```

output:

```
4
```

Example 3:

input:

```
Recursion
Droste effect
```

output:

```
12
```

Problem 4: Maze

In this problem you need to solve a maze that is represented by ASCII input. The input consists of a line with two positive integers m and n (both are less than 20), followed by a $m \times n$ grid that represents the maze. The maze consists of m lines, each containing n characters followed by a newline (`'\n'`) character. The entrance of the maze is represented by the capital letter `S`, and the exit is represented by the capital letter `E`.

Your program should find the *shortest path* through the maze starting at `S` and ending at `E`. This path is only allowed to pass grid cells containing a dot (`'.'`) character. The path should be marked with the character `'#'` in the output. A path consists of steps in 4 directions: North, East, South, and West (so no diagonal moves). [Note that the test cases in Themis are chosen such that the shortest path is unique.]

Example 1:

input:

```
11 11
+-+-+-+-+-+
S.|...|...|
+.+.+.+-+.+
|.|.|.....|
+.+-+-+-+.+
|...|.|...|
+.+.+.+.+-+
|.|...|.|.|
+.+-+.+.+.+
|...|.....E
+-+-+-+-+-+
```

output:

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

Example 2:

input:

```
11 11
+-+-+-+-+-+
S.|.......|
+.+.+-+-+.+
|.|.|.....|
+.+.+.+-+-+
|.|.|.....|
+.+.+-+-+.+
|.|...|...|
+.+.+.+.+-+
|...|.|...E
+-+-+-+-+-+
```

output:

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

Example 3:

input:

```
11 11
+-+-+-+-+-+
S.|.....|.|
+.+.+.+.+.+
|...|.|...|
+-+-+.+-+-+
|.|...|...|
+.+.+-+.+.+
|.|.....|.|
+.+-+-+-+.+
|.........E
+-+-+-+-+-+
```

output:

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