

Programming Assignment 1

1. Letter-Frequency Histogram

Count the frequency of each alphabetic character in the supplied text, ignoring case. Characters that are not letters are skipped.

Print one line per letter in the form `<letter> <count>` using lowercase letters. Order the output primarily by descending count; letters that tie are printed in alphabetical order.

Sample 1 Input

Hello, World!

Sample 1 Output

```
l 3
o 2
d 1
e 1
h 1
r 1
```

Sample 2 Input

banana BANANA

Sample 2 Output

```
a 6
b 2
n 2
```

2. Run-Length Encoder / Decoder

Implement two functions using run-length encoding (RLE).

- `compress(s)` converts a string like `aaabbc` into `a3b2c1`.
- `decompress(rle)` reverses the process.

Every run is guaranteed to have length at most 9, so each count fits in a single digit.

Sample 1 Input

aaabbc

Sample 1 Output

a3b2c1

Sample 2 Input

abcd

Sample 2 Output

a1b1c1d1

3. Print Matrix in Spiral Order

Fill an $n \times n$ grid with the integers 1 through n^2 in a clock-wise spiral that starts in the upper-left corner, proceeds right, then down, left, and up, tightening inward until the grid is full.

For $n = 4$ the grid should look like

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

The function `spiral_print(n)` must *return* the grid as a list of lists.

Sample 1 Input

3

Sample 1 Output

[[1, 2, 3], [8, 9, 4], [7, 6, 5]]

Sample 2 Input

4

Sample 2 Output

[[1, 2, 3, 4], [12, 13, 14, 5], [11, 16, 15, 6], [10, 9, 8, 7]]

4. Tic-Tac-Toe Winner

The board is a 3×3 list of lists whose entries are 'X', 'O', or the empty string ''. Return

- 'X' if X has three in a row,
- 'O' if O has three in a row,
- None otherwise.

You don't need to check if the board is valid or if the game is over.

Sample 1 Input

```
[[ 'X', 'O', 'X'], ['O', 'X', 'O'], [' ', 'O', 'X']]
```

Sample 1 Output

X

Sample 2 Input

```
[[ 'O', 'O', 'O'], ['X', ' ', 'X'], [' ', 'X', ' ']]
```

Sample 2 Output

O

Sample 3 Input

```
[[ 'X', 'O', 'X'], ['O', 'X', 'O'], ['O', 'X', 'O']]
```

Sample 3 Output

None

5. Merge Overlapping Intervals

Given a list of half-open intervals of the form (start, end) with $\text{start} < \text{end}$, write `merge(intervals)` that combines intervals which overlap or touch (that is, the next interval begins at or before the current one ends). The result must be sorted by start time.

Example: [(1,3), (2,4), (5,7), (7,8)] becomes [(1,4), (5,8)].

Sample 1 Input

```
[(1,3),(2,4),(5,7),(7,8)]
```

Sample 1 Output

```
[(1, 4), (5, 8)]
```

Sample 2 Input

```
[(0,1),(2,3)]
```

Sample 2 Output

```
[(0, 1), (2, 3)]
```

6. Balanced Brackets

Implement `is_balanced(expr)` that checks if a string of parentheses (`expr`) is balanced, meaning that every opening parenthesis (has a matching closing parenthesis) that appears later in the string.

Return `True` if every (has a matching) that appears later in the string and the pairs are properly nested; otherwise return `False`.

Sample 1 Input

`(( ))`

Sample 1 Output

True

Sample 2 Input

`(( ) ( ))`

Sample 2 Output

True

Sample 3 Input

`(( )) (`

Sample 3 Output

False