Problem 1: Recursive Digital Root

Compute the digital root of an integer recursively until a single-digit value remains.

Function Prototype: digital_root

Input:
A single integer N ($0 \leq N \leq 10^9$)

Output:
A single integer — the digital root of N.

Sample Input:

9875

Sample Output:

2

Explanation: 9+8+7+5 -> 29; 2+9 -> 11; 1+1 -> 2

———————————————————————————————————

Problem 2: Maximum and Minimum in an Array

Find both maximum and minimum values in an array using a single function call.

Function Prototype: find_min_max

Input:
First line: N ($1 \leq N \leq 100$)
Second line: N space-separated integers.

Output:
Print the maximum and minimum separated by a space.

Sample Input:

5
3 9 1 4 7

Sample Output:

9 1

———————————————————————————————————

Problem 3: Matrix Multiplication with Transpose

Compute C = A × B^T

Input:
First line: r1 c1 r2
Next r1×c1 integers for A, followed by r2×c1 integers for B.

Output:
Matrix C in row-major order.

Sample Input:

- 2 3 2
  1 2 3
  4 5 6
  7 8 9
  10 11 12

Sample Output:

- 50 122
  68 167

———————————————————————————————————

Problem 4: Factorial with Overflow Detection

Compute factorial recursively and detect overflow (> 32-bit).

Function Prototype: factorial_safe

Input:
A single integer N ($0 \leq N \leq 20$)

Output:
Print N! or 'OVERFLOW' if it exceeds 32-bit unsigned integer range.

Sample Input:

- 13

Sample Output:

- OVERFLOW

———————————————————————————————————

Problem 5: Palindrome Check

Check whether a given string is a palindrome using two-pointer traversal.

Function Prototype: is_palindrome

Input:
A single string (no spaces, up to 100 chars).

Output:
YES if palindrome, NO otherwise.

Sample Input:

- MADAM

Sample Output:

- YES

———————————————————————————————————————

Problem 6: Centered Pyramid Pattern

Print a centered pyramid of stars using nested loops.

Function Prototype: print_pyramid

Input:
A single integer N ($1 \leq N \leq 10$)

Output:
A pyramid with N rows of '*' characters.

Sample Input:

- 3

Sample Output:

-
```
  *
 ***
*****
```

———————————————————————————————————————