

Laboratory Session 01

Recursion

1 Power of a Number

Write a recursive function `power(base, exponent)` that calculates the power of a given number, assuming non-negative integer for bases and exponents.

Examples

Input	Expected Output
<code>power(2, 3)</code>	8
<code>power(5, 0)</code>	1
<code>power(3, 4)</code>	81

2 Annual interest

Write a recursive function `annual_interest(principal, rate, years)` that computes the accumulated amount of money after a given number of years, given an initial principal amount and an annual interest rate (e.g., 0.05 for 5%).

The accumulated amount is defined by the following recurrence relation:

$$A(n) = \begin{cases} P & \text{if } n = 0 \\ A(n-1)(1+r) & \text{if } n > 0 \end{cases}$$

where P is the initial principal amount and r is the annual interest rate. Since the interest is compounded annually, the amount for each year is obtained by applying the interest rate r to the amount accumulated during the previous year.

Examples

Input	Expected Output
<code>annual_interest(1000.0, 0.05, 2)</code>	1102.5
<code>annual_interest(500.0, 0.10, 3)</code>	665.5

3 Fibonacci sequence

The Fibonacci sequence is a mathematical sequence of numbers where each term is the sum of the two preceding ones. Formally, it is defined by the following recurrence relation:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

Write a recursive function `fibonacci(n)` that takes a non-negative integer n and calculates the n -th Fibonacci number $F(n)$ based on the recurrence defined above.

Examples

Input	Expected Output
<code>fibonacci(6)</code>	8
<code>fibonacci(8)</code>	21

4 Subset Sum

The Subset Sum problem is a classic decision problem in computer science. Given a set (or multiset) of integers $S = \{s_1, s_2, \dots, s_n\}$ and a target integer T , the goal is to determine whether there exists a subset $S' \subseteq S$ such that the sum of its elements equals T :

$$\sum_{x \in S'} x = T$$

From a recursive perspective, for any given element $s_i \in S$, you face two distinct choices:

1. **Include s_i in the subset:** The remaining problem is to find a subset from the rest of the list that sums to $T - s_i$.
2. **Exclude s_i from the subset:** The remaining problem is to find a subset from the rest of the list that sums to T .

Write a recursive function `subset_sum(numbers, target)` that takes a list of integers `numbers` and an integer `target`, and returns `True` if a valid subset exists, or `False` otherwise.

Examples

Input	Expected Output
<code>subset_sum([3, 34, 4, 12, 5, 2], 9)</code>	<code>True</code>
<code>subset_sum([3, 34, 4, 12, 5, 2], 30)</code>	<code>False</code>

5 Instructions

How to Complete This Lab

In this lab, you are provided with this PDF containing the exercise instructions, the necessary libraries (if there are libraries to use) and expected Input/Output examples (e.g., **Input:** `PowerOf(2, 3)` $\rightarrow$ **Output:** `8`).

Along with this document, you have received two Python files:

- **exercises.py:** This file contains the function signatures with empty bodies for all exercises. For example:

```
def PowerOf(base: int, exponent: int) -> int: pass
```

Your task is to replace the `pass` keyword with your actual implementation. Do not change the function names or their signatures. You can add other functions to support your code.

- **main.py:** This file contains a suite of automated tests. Once you have written your code in **exercises.py**, run **main.py**. It will automatically import your functions, test them against various inputs, and report if your outputs are correct (True for correct result, False otherwise).