```
def company = [
   "Engineering" : ["Alice": 5000, "Bob": 4500],
   "Sales"       : ["Carol": 3000, "David": 3500],
   "HR"          : ["Eve": 4000, "Frank": 3700]
]
```

Expected Output:

```
def totalSalaries = [
   "Engineering" : 9500,
   "Sales"       : 6500,
   "HR"          : 7700
]
```

```
def products = [
   "Electronics" : ["Phone": [500, 550, 600], "Laptop": [1000, 1100, 1200]],
   "Furniture"   : ["Table": [200, 250], "Chair": [150, 170]]
]
```

Expected Output:

```
def avgPrices = [
   "Phone"  : 550,
   "Laptop" : 1100,
   "Table"  : 225,
   "Chair"  : 160
]
```

```
def studentGrades = [
   "John"  : ["Math": 85, "English": 78, "Science": 92],
   "Sarah" : ["Math": 88, "English": 90, "Science": 85],
   "Mike"  : ["Math": 60, "English": 65, "Science": 70]
]
```

Expected Output:

```
def avgGrades = [
    "John"  : 85.0,
    "Sarah" : 87.67,
    "Mike"  : 65.0
]
```

```
def warehouseStock = [
    "Warehouse1" : ["Laptop": 10, "Phone": 20, "Tablet": 15],
    "Warehouse2" : ["Laptop": 5, "Phone": 10, "Tablet": 20],
    "Warehouse3" : ["Laptop": 8, "Phone": 12, "Tablet": 7]
]
```

Expected Output:

```
def totalStock = [
    "Laptop" : 23,
    "Phone"  : 42,
    "Tablet" : 42
]
```

```
def cityPopulation = [
    "CityA" : ["District1": 50000, "District2": 75000, "District3": 100000],
    "CityB" : ["District1": 60000, "District2": 85000],
    "CityC" : ["District1": 70000, "District2": 90000, "District3": 110000, "District4": 50000]
]
```

Expected Output:

```
def totalCityPopulation = [
    "CityA" : 225000,
    "CityB" : 145000,
    "CityC" : 320000
]
```

```
def employeeHierarchy = [
   "Engineering" : ["Alice": [25, 6000], "Bob": [30, 4500]],
   "Sales"       : ["Carol": [28, 5000], "David": [23, 3500]],
   "HR"          : ["Eve": [35, 7000], "Frank": [40, 6500]]
]
```

Expected Output:

```
def employeeDepartment = [
   "Alice" : "Engineering",
   "Bob"   : "Engineering",
   "Carol" : "Sales",
   "David" : "Sales",
   "Eve"   : "HR",
   "Frank" : "HR"
]
```

Next:

```
def companyProjects =
   "CompanyA" : [
      "ProjectX": ["Alice", "Bob", "Carol"],
      "ProjectY": ["Alice", "David"]
   ],
   "CompanyB" : [
      "ProjectZ": ["Bob", "Carol", "Eve"],
      "ProjectW": ["David", "Frank"]
   ]
]
```

Expected Output:

```
def employeeProjectCount = [
   "Alice" : 2,
   "Bob"   : 2,
   "Carol" : 2,
   "David" : 2,
```

```
    "Eve"   : 1,
    "Frank" : 1
]
```

```
def universities = [
    "UniversityA": [
        "Math101" : ["Prof. Smith", "Prof. Johnson"],
        "CS102"   : ["Prof. Johnson", "Prof. Williams"]
    ],
    "UniversityB": [
        "Physics101" : ["Prof. Smith", "Prof. Lee"],
        "CS102"      : ["Prof. Johnson"]
    ]
]
```

Expected Output:

```
def professorUniversities = [
    "Prof. Smith"   : ["UniversityA", "UniversityB"],
    "Prof. Johnson" : ["UniversityA", "UniversityB"],
    "Prof. Williams": ["UniversityA"],
    "Prof. Lee"     : ["UniversityB"]
]
```

```
def departmentBudgets = [
    "Engineering": [
        "ProjectX": 100000,
        "ProjectY": 150000
    ],
    "Marketing"  : [
        "ProjectA": 50000,
        "ProjectB": 30000
    ],
    "Sales"      : [
```

```
        "ProjectZ": 80000,
        "ProjectW": 60000
    ]
]
```

Expected Output:

```
def totalDepartmentBudgets = [
    "Engineering" : 250000,
    "Marketing"   : 80000,
    "Sales"       : 140000
]
```

```
def studentGrades = [
    "John": [
        "Math": 85,
        "English": 78,
        "Science": 92
    ],
    "Sarah": [
        "Math": 88,
        "English": 90,
        "Science": 85
    ],
    "Mike": [
        "Math": 60,
        "English": 65,
        "Science": 70
    ]
]
```

Expected Output:

```
def avgCourseGrades = [
    "Math"    : 77.67,
```

```
    "English" : 77.67,
    "Science" : 82.33
]
```

```
def cityPopulation = [
    "CityA" : [
        "District1": [50000, 52000, 54000],
        "District2": [75000, 78000, 81000]
    ],
    "CityB" : [
        "District1": [60000, 62000, 65000],
        "District3": [70000, 73000, 76000]
    ]
]
```

Expected Output:

```
def districtGrowth = [
    "District1" : 8.0,   // Percentage growth from 50000 to 54000
    "District2" : 8.0,   // Percentage growth from 75000 to 81000
    "District3" : 8.57   // Percentage growth from 70000 to 76000
]
```

```
def projectSkills = [
    "ProjectX": [
        "Alice": ["Java", "Groovy", "SQL"],
        "Bob": ["Groovy", "Kotlin", "Python"]
    ],
    "ProjectY": [
        "Carol": ["Python", "SQL", "JavaScript"],
        "David": ["JavaScript", "HTML", "CSS"]
    ]
]
```

```
def projectUniqueSkills = [
    "ProjectX": ["Java", "Groovy", "SQL", "Kotlin", "Python"],
    "ProjectY": ["Python", "SQL", "JavaScript", "HTML", "CSS"]
]
```

Next

Problem 1: Group Products by Price Range and Category

Question: Create a new nested map that groups products into "Low", "Medium", and "High" price ranges based on their price.

```
def products = [
    "Electronics": [
        "Laptop": 1000,
        "Phone" : 300,
        "Headphones": 100
    ],
    "Furniture": [
        "Chair" : 150,
        "Table" : 450,
        "Lamp"  : 50
    ],
    "Clothing": [
        "Shirt": 40,
        "Jeans": 60,
        "Jacket": 200
    ]
]
```

Expected Output:

```
[
    "Low": [
        "Headphones": 100,
```

```
    "Chair" : 150,
    "Lamp"  : 50,
    "Shirt": 40,
    "Jeans": 60
  ],
  "Medium": [
    "Phone" : 300,
    "Table" : 450,
    "Jacket": 200
  ],
  "High": [
    "Laptop": 1000
  ]
]
```

Problem 2: Deep Merge Two Maps with Nested Structures

Question: Write a function that merges two nested maps deeply. If the same key exists in both maps, keep the latest value from the second map.

```
def userProfile1 = [
  "John": [
    "age" : 30,
    "address": [
      "city": "New York",
      "zip": "10001"
    ],
    "phone": "123-456"
  ]
]

def userProfile2 = [
  "John": [
```

```
        "age" : 31,
        "address": [
           "city": "Los Angeles",
           "zip": "90001"
        ],
        "email": "john@example.com"
    ]
]
```

Expected Output:

```
[
    "John": [
        "age": 31,
        "address": [
           "city": "Los Angeles",
           "zip": "90001"
        ],
        "phone": "123-456",
        "email": "john@example.com"
    ]
]
```

Problem 3: Aggregate Employee Performance by Department

Question: Calculate the average performance score for each department.

```
def company = [
    "Engineering": [
        "Alice" : 85,
        "Bob"   : 92,
        "Carol" : 75
    ],
    "Marketing": [
        "David" : 88,
```

```
        "Eve"   : 91
    ],
    "Sales": [
        "Frank" : 65,
        "Grace" : 78
    ]
]
```

Expected Output:

```
[
    "Engineering": 84,
    "Marketing"  : 89.5,
    "Sales"      : 71.5
]
```

Problem 4: Create a Map of Top Performers in Each Department

Question: Create a new map showing the top performer (employee with the highest score) in each department.

```
def departments = [
    "Engineering": [
        "Alice" : 85,
        "Bob"   : 92,
        "Carol" : 75
    ],
    "HR": [
        "David" : 88,
        "Eve"   : 81,
        "Frank" : 90
    ],
    "Finance": [
        "Grace" : 95,
        "Hank"  : 80
```

```
    ]
]
```

Expected Output:

```
[
    "Engineering": ["Bob": 92],
    "HR"        : ["Frank": 90],
    "Finance"   : ["Grace": 95]
]
```

Problem 5: Normalize Stock Quantities across Multiple Warehouses

Question: Normalize the stock quantities so that each item's quantity is expressed as a percentage of the total stock available across all warehouses.

```
def warehouses = [
    "Warehouse1": [
        "Laptop": 50,
        "Phone" : 100,
        "Tablet": 75
    ],
    "Warehouse2": [
        "Laptop": 30,
        "Phone" : 150,
        "Tablet": 50
    ],
    "Warehouse3": [
        "Laptop": 20,
        "Phone" : 50,
        "Tablet": 100
    ]
]
```

Expected Output:

```
[
```

```
    "Laptop": [
        "Warehouse1": 50/100 * 100,
        "Warehouse2": 30/100 * 100,
        "Warehouse3": 20/100 * 100
    ],
    "Phone": [
        "Warehouse1": 100/300 * 100,
        "Warehouse2": 150/300 * 100,
        "Warehouse3": 50/300 * 100
    ],
    "Tablet": [
        "Warehouse1": 75/225 * 100,
        "Warehouse2": 50/225 * 100,
        "Warehouse3": 100/225 * 100
    ]
]
```

Problem 6: Generate Hierarchical Employee-Manager Map

Question: Generate a hierarchical map that lists employees under their respective managers.

```
def employees = [
    "CEO": [
        "VP1": [
            "Manager1": ["Employee1", "Employee2"],
            "Manager2": ["Employee3"]
        ],
        "VP2": [
            "Manager3": ["Employee4", "Employee5"],
            "Manager4": ["Employee6"]
        ]
    ]
]
```

Expected Output:

```
[
   "CEO": [
      "VP1": [
         "Manager1": ["Employee1", "Employee2"],
         "Manager2": ["Employee3"]
      ],
      "VP2": [
         "Manager3": ["Employee4", "Employee5"],
         "Manager4": ["Employee6"]
      ]
   ]
]
```