GROOVY BASIC PROBLEMS

.collect : **Transforms each element of a collection and returns a new collection with the results.**

def numbers = [1, 2, 3, 4, 5]

Expected Output:

[1, 4, 9, 16, 25]

def names = ["Alice", "Bob", "Carol"]

Expected Output:

["ALICE", "BOB", "CAROL"]

def words = ["apple", "banana", "pear"]

Expected Output:

[5, 6, 4]

def items = ["Book", "Pen", "Laptop"]

Expected Output:

["Item: Book", "Item: Pen", "Item: Laptop"]

def pricesInDollars = [10, 20, 30, 40] (assuming 1 USD = 0.85 EUR).

Expected Output:

[8.5, 17.0, 25.5, 34.0]

.find : **Returns the first element in a collection that matches a given condition.**

def numbers = [1, 3, 5, 8, 9]

Expected Output:

8

def names = ["Bob", "Alice", "Charlie"]

Expected Output:

"Alice"

GROOVY BASIC PROBLEMS

def numbers = [10, 25, 60, 80]

Expected Output:

60

def words = ["dog", "catalog", "rat", "scatter"]

Expected Output:

"catalog"

```
def products = [
   [name: "Laptop", price: 1200],
   [name: "Phone", price: 800],
   [name: "Tablet", price: 600]
]
```

Expected Output:

[name: "Laptop", price: 1200]

.findAll : Returns all elements in a collection that match a given condition.

def numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Expected Output:

[2, 4, 6, 8, 10]

def names = ["Alice", "Bob", "Sarah", "Steve", "Carol"] (find all names that start with the letter "S")

Expected Output:

["Sarah", "Steve"]

def numbers = [25, 60, 75, 30, 10, 80] (find all numbers greater than 50.)

Expected Output:

[60, 75, 80]

GROOVY BASIC PROBLEMS

```
def words = ["apple", "banana", "pear", "plum", "grape"]
```

Expected Output:

```
["apple", "banana", "pear", "grape"]
```

```
def products = [
   [name: "Laptop", price: 1000],
   [name: "Phone", price: 500],
   [name: "Tablet", price: 300]
]
```

Expected Output:

```
[[name: "Laptop", price: 1000]]
```

.each : Iterates over a collection (like a list or map) and performs an action on each element.

```
def products = [Laptop: 1000, Phone: 500, Tablet: 300]
```

Output:

```
Laptop costs 1000

Phone costs 500

Tablet costs 300
```

```
def numbers = [5, 10, 15, 20]
```

Output:

```
Sum: 50
```

```
def stock = [Apples: 50, Bananas: 30, Oranges: 20] (update quantity)
```

Output:

```
[Apples: 60, Bananas: 40, Oranges: 30]
```

.collectEntries : Transforms a map or list into a new map by applying a transformation to each key-value pair.

GROOVY BASIC PROBLEMS

def products = [Laptop: 1000, Phone: 500, Tablet: 300] (price is reduced by 20%.)

Expected Output:

[Laptop: 800, Phone: 400, Tablet: 240]

def names = ["Alice", "Bob", "Carol"]

Expected Output:

[Alice: 5, Bob: 3, Carol: 5]

def stock = [Laptop: 5, Phone: 0, Tablet: 3]

Expected Output:

[Laptop: "In Stock", Phone: "Out of Stock", Tablet: "In Stock"]

def numbers = [one: 1, two: 2, three: 3, four: 4]

Expected Output:

[one: 2, two: 4, three: 6, four: 8]

def countries = ["Canada", "Australia", "India", "Germany"]

Expected Output:

[Canada: "CAN", Australia: "AUS", India: "IND", Germany: "GER"]

.eachWithIndex : Iterates over a collection and performs an action on each element while keeping track of the element's index.

def numbers = [10, 20, 30, 40]

Expected Output:

0: 10

1: 20

2: 30

3: 40

GROOVY BASIC PROBLEMS

def numbers = [1, 2, 3, 4] (to calculate the sum of each number multiplied by its index.)

Expected Output:

Sum: 20

def items = ["Apple", "Banana", "Mango"]

Expected Output:

[0: "Apple", 1: "Banana", 2: "Mango"]

def products = ["Laptop", "Phone", "Tablet"] (o create a map where each product is paired with a unique code based on its index (e.g., "P1", "P2", etc.))

Expected Output:

[Laptop: "P1", Phone: "P2", Tablet: "P3"]

def numbers = [10, 20, 30, 40, 50] (to replace all elements at odd indices with the value -1.)

Expected Output:

[10, -1, 30, -1, 50]

GROOVY BASIC PROBLEMS