Exercise 1a: Creation Easy

Create a namespace `req` where `status` is `200` and `Method` is `{4+2×⍵}`.

```
      your ◇ expressions
      req.status
200
      req.Method 200
404
```


Exercise 1b: Updating Easy

Within `req`, apply `Method` to `status`, and update `status` with the result.

```
      req.status
200
      your_expression
      req.status
404
```


Exercise 1c: Importing a workspace Medium

Write a function `Into` that copies a workspace into a namespace (using `⎕CY`).

```
      dfns←⎕NS⍬
      'dfns.dws' Into dfns
      dfns.disp dfns.morse'SOS'
```

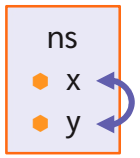
...	---	...

This corner intentionally left blank

Exercise 2a: Swap names Easy

Write an *expression* that swaps the values of variables `x` and `y` in a namespace `ns` .

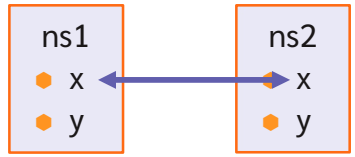
```
      ns←⎕NS⍬
      ns.x←10 ⋄ ns.y←20
      your_expression
      ns.x ns.y
20 10
```



Exercise 2b: Swap namespaces Easy

Write an *expression* that swaps the values of the variables named `x` in the namespaces `ns1` and `ns2` .

```
      ns1←⎕NS⍬ ⋄ ns2←⎕NS⍬
      ns1.x←10 ⋄ ns2.x←20
      ns1.y←30 ⋄ ns2.y←40
      your_expression
      ns1.x ns1.y ns2.x ns2.y
20 30 10 40
```



Exercise 2c: Testing if reference Medium

Write a function `ScalarRef` that returns a scalar Boolean value indicating whether its argument is a scalar namespace.

```
      ns←⎕NS⍬ ⋄ ns.a←10
      vec←ns.a 'abc' (ns ns) ns (⎕NS⍬) 42 ⍬ #
      ScalarRef¨ vec
0 0 0 1 1 0 0 1
```

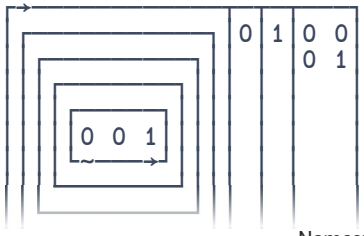
Use one or more of these scalar namespace properties:

- `40⎕ATX` is 9
- `⎕DR` is 326
- Depth (`≡`) is 0 (simple scalar)
- Allows dot syntax (`ns.name`)

Exercise 2d: Indicate references Hard

Write a function `RefMask` that returns an array of the same structure as its argument, but with bits indicating any ns refs.

```
      ]disp RefMask (⊂⊂⊂1 2 ns) 3 ns (2 2⍴'abc',⎕NS⍬)
```



Exercise 3a: Get multiple values Medium

Write a function `Get` that takes a namespace as left argument and a vector of names as right argument and returns the corresponding values from the namespace.

```
      ns←⎕NS⍬
      ns.(foo bar baz)←(1 2 3) 'Hello' 42

      names←'foo' 'bar' 'foo' 'baz'
      ns Get names
 1 2 3  Hello  1 2 3  42
```


Exercise 3b: Set a value Medium

Write a function `Set` that takes a two-element `(name value)` vector as right argument, then does the corresponding assignment.

```
      Set 'my' (⍳5)
      10×my
10 20 30 40 50
```


Exercise 3c: Set in a namespace Medium

Improve `Set` so that it takes a namespace as left argument and does the corresponding assignment in that namespace.

```
      ns←⎕NS⍬
      ns Set 'my' (⍳5)
      10×ns.my
10 20 30 40 50
```


Exercise 3d: Set multiple values Hard

Improve `Set` so that it handles multiple two-element `(name value)` vectors as right argument.

```
      ns←⎕NS⍬
      ns Set ('your' 'Hello')('my' 'World')
      ns.(your my)
 Hello  World
```

Exercise 4a: Is argument a root? Easy

Write a function `IsRoot` that takes a namespace as argument that tells you whether that namespace is a root namespace.

```
      IsRoot ⎕SE.Dyalog.Utils
0
      IsRoot #
1
      IsRoot ⎕SE
1
```


Exercise 4b: What is my root? Medium

Write a function `FindRoot` that takes a namespace as argument and returns its root.

```
      FindRoot ⎕SE.Dyalog.Utils
⎕SE
      FindRoot #
#
      FindRoot ⎕NS⍬
#
```


Exercise 4c: Our roots? Easy — based on `FindRoot`

Write a function `FindRoots` that takes an arbitrary array of namespaces and finds the root for each namespace.

```
      FindRoots ⍪⎕SE.Dyalog.Utils(#,⎕NS⍬)⎕SE
```

⎕SE
# #
⎕SE

Exercise 4d: Namespace lineage Hard

Write a function `Line` that takes a single namespace and returns its lineage (as a vector of refs) from root to leaf.

```
      Line ⎕SE.Dyalog.Utils
 ⎕SE  ⎕SE.Dyalog  ⎕SE.Dyalog.Utils

      Line ⎕SE.cbbot.bandsb2.sb.io
 ⎕SE  ⎕SE.cbbot  ⎕SE.cbbot.bandsb2
⎕SE.cbbot.bandsb2.sb  ⎕SE.cbbot.bandsb2.sb.io
```

Exercise 5: Where are my children? Impossible

Write a function that lists all the children of a given namespace.

Tips:

- Note: ⎕NL is Name List, not Children List
- Plan: You'll have to crawl through the entire workspace
- Think: How could namespaces still be out of reach?

Exercise 5: Where are my children? Impossible

```
∇ children←{arg}ListChildren target;args;next;parents;visited
  :If 0=⎕NC'arg' ⋄ arg←(# ⎕SE target)(0ρ#)(0ρ#) ⋄ :EndIf
  (parents children visited)←arg
  next←∪∈parents.(⍎¨'##' '⎕THIS',⎕NL ¯9)  ⍝ visit all reachable ns
  next~←visited                           ⍝ excepted visited ones
  :If 0∈ρnext ⋄ :Return ⋄ :EndIf          ⍝ (0ρ#).## is NONCE ERROR
  children∪←(target=next.##)/next         ⍝ append children of target
  visited,←next                           ⍝ no next has been visited
  :If 0∈ρnext ⋄ :Return ⋄ :EndIf          ⍝ F¨0ρ# is NONCE ERROR
  args←next children visited
  children∪←args ListChildren target      ⍝ recur on unvisited ns
  ⍝ TO-DO: refs in arrays, dervs, locals in threads, fields in OO, …
∇
```

This corner intentionally left blank

This corner intentionally left blank