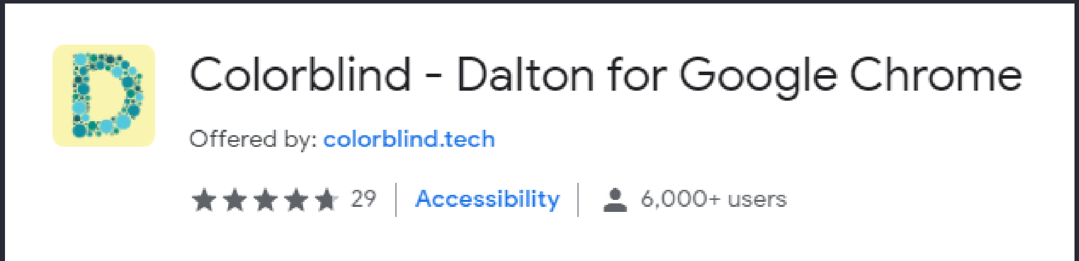
ALG⌽RITHMS AS A T⌽⌽L ⌽F TH⌽UGHT

Conor Hoekstra

code_report

NVIDIA RAPIDS

#include



Colorblind - Dalton for Google Chrome

Offered by: colorblind.tech

★★★★★ 29 | Accessibility | 6,000+ users

https://github.com/codereport/Talks

Algorithms +
Data
Structures =
Programs

NVIDIA

RAPIDS

ode_report

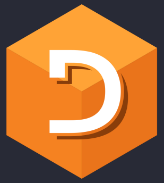
YouTube

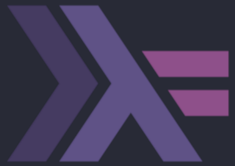
NVIDIA®

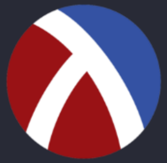
RAPIDS

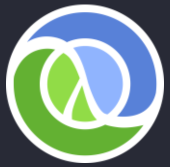
C++

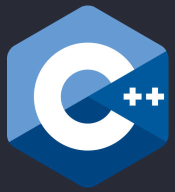
cpp.chat

1. 

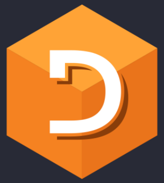
2. 

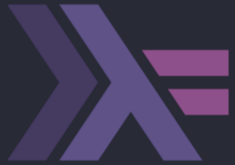
3. 

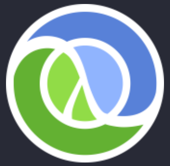
4. 

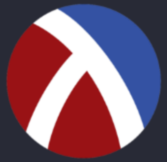
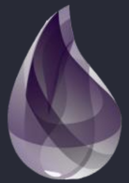
5. 

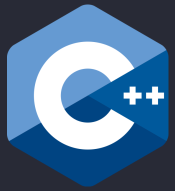
♡

1. 

2. 

3. 

4.  / 

5. 

1979 ACM Turing Award Lecture

Delivered at ACM '79, Detroit, Oct. 29, 1979

The 1979 ACM Turing Award was presented to Kenneth E. Iverson by Walter Carlson, Chairman of the Awards Committee, at the ACM Annual Conference in Detroit, Michigan, October 29, 1979.

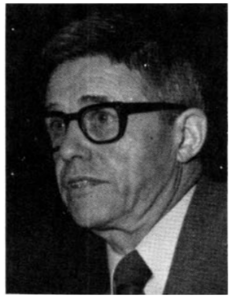
In making its selection, the General Technical Achievement Award Committee cited Iverson for his pioneering effort in programming languages and mathematical notation resulting in what the computing field now knows as APL. Iverson's contributions to the implementation of interactive systems, to the educational uses of APL, and to programming language theory and practice were also noted.

Born and raised in Canada, Iverson received his doctorate in 1954 from Harvard University. There he served as Assistant Professor of Applied Mathematics from 1955–1960. He then joined International Business Machines, Corp. and in 1970 was named an IBM Fellow in honor of his contribution to the development of APL.

Dr. Iverson is presently with I.P. Sharp Associates in Toronto. He has published numerous articles on programming languages and has written four books about programming and mathematics: *A Programming Language* (1962), *Elementary Functions* (1966), *Algebra: An Algorithmic Treatment* (1972), and *Elementary Analysis* (1976).

Notation as a Tool of Thought

Kenneth E. Iverson
IBM Thomas J. Watson Research Center



Key Words and Phrases: APL, mathematical notation
CR Category: 4.2

The importance of nomenclature, notation, and language as tools of thought has long been recognized. In chemistry and in botany, for example, the establishment of systems of nomenclature by Lavoisier and Linnaeus did much to stimulate and to channel later investigation. Concerning language, George Boole in his *Laws of Thought* [1, p.24] asserted "That language is an instrument of human reason, and not merely a medium for the expression of thought, is a truth generally admitted."

Mathematical notation provides perhaps the best-known and best-developed example of language used consciously as a tool of thought. Recognition of the important role of notation in mathematics is clear from the quotations from mathematicians given in Cajori's *A History of Mathematical Notations* [2, pp.332,331]. They are well worth reading in full, but the following excerpts suggest the tone:

> By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.
>
> A.N. Whitehead

Author's present address: K.E. Iverson, I.P Sharp Associates, 145 King Street West, Toronto, Ontario, Canada M5H1J8.

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

¨ / \ ⍨ ⌸ ⍣ . ⍥

ALGORITHMS

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

¨ / \ ⍨ ⌸ ⍣ . ⍤

OPERATORS

FUNCTIONS

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

¨ / \ ⍨ ⌸ ⍣ . ⍤

OPERATORS

VERBS

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

¨ / \ ⍨ ⌸ ⍣ . ⍤

ADVERBS & CONJUNCTIONS

ALGORITHMS

⌽ ⍉ ∪ ⍋ ⊆ ↑ ⍳ ∊

¨ / \ ⍨ ⌸ ⍣ . ⍤

OPERATORS

ALGORITHMS

⌽	reverse
∪	unique
⊆	partition
⍳	iota

ALGORITHMS

⌽	reverse	reverse
∪	unique	unique
⊆	partition	partition
⍳	iota	iota

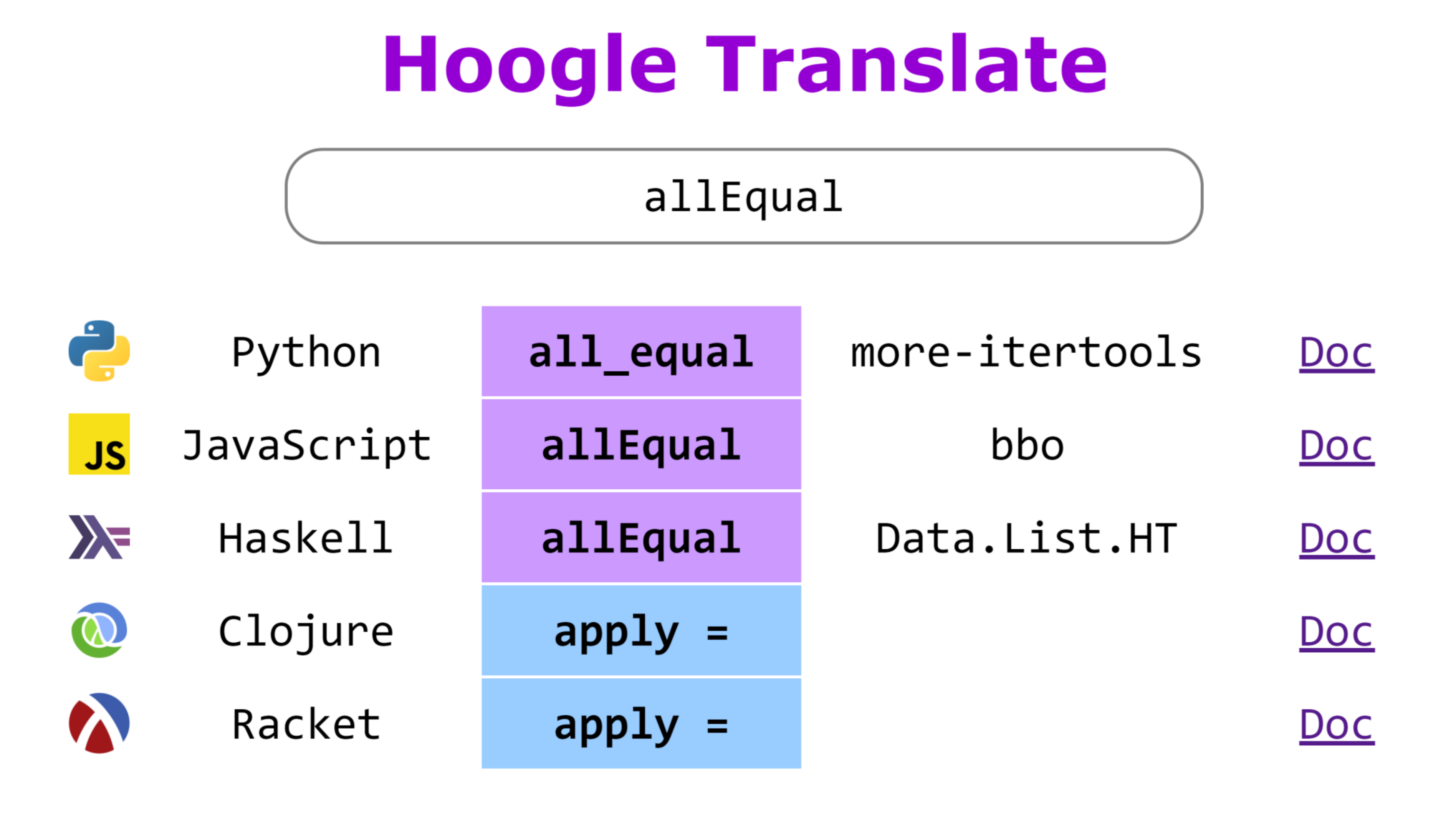
allEqual

```haskell
import Data.List.HT (allEqual)

allEqual [1,2,3,4] -- False
allEqual [1,1,1,1] -- True
```

```python
from more_itertools import all_equal

all_equal([1,2,3,4]) # False
all_equal([1,1,1,1]) # True
```



Hoogle Translate

allEqual

	Python	**all_equal**	more-itertools	Doc
JS	JavaScript	**allEqual**	bbo	Doc
	Haskell	**allEqual**	Data.List.HT	Doc
	Clojure	**apply =**		Doc
	Racket	**apply =**		Doc

SOLUTION #1

FREE

⊃

first

```
⊃ 1 2 3 4
```

1

```
{(⊃⍵)=⍵} 1 2 3 4
```

```
{(⊃1 2 3 4)=1 2 3 4}
```

1=1 2 3 4

1 0 0 0

```
^/1 0 0 0
```

```
  ^/1 0 0 0

1 ^ 0 ^ 0 ^ 0
```

APL	/ (reduce)	-	Doc
CUDA	reduce	Thrust	Doc
D	reduce	algorithm.iteration	Doc
Ruby	reduce	Enumerable	Doc
Python	reduce	itertools	Doc
Elixir	reduce	Enum	Doc
Kotlin	reduce	collections	Doc
Clojure	reduce	core	Doc
C++	reduce	<numeric>	Doc
Haskell	foldl	Data.List	Doc
Racket	foldl	base	Doc
Rust	fold	trait.Iterator	Doc
q	over	-	Doc
C#	Aggregate	Enumerable	Doc
J	/ (insert)	-	Doc
C++	accumulate	<numeric>	Doc

```
  ^/1 0 0 0

1 ^ 0 ^ 0 ^ 0
```

0

```
∧/{(⊃⍵)=⍵}
```

```
∧/{(⊃⍵)=(⊢⍵)}
```

FREE

FREE

```
∧/⊃=⊢
```

⊃	first
=	equal
∧	and
⊢	same
/	reduce

SOLUTION #2

FREE

```
(1=≠)∪
```

U

unique

u 1 2 2 4

1 2 4

```
≢ 1 2 4
```

tally

3

```
1 = 3
```

0

```
1=≠∪1  2  2  4
```

FREE

(1 = ≠) u

FREE

```
(1=≢)∪
```

∪	unique
=	equal
≢	tally

```
(1≥≢)∪
```

∪	unique
≥	gte
≢	tally

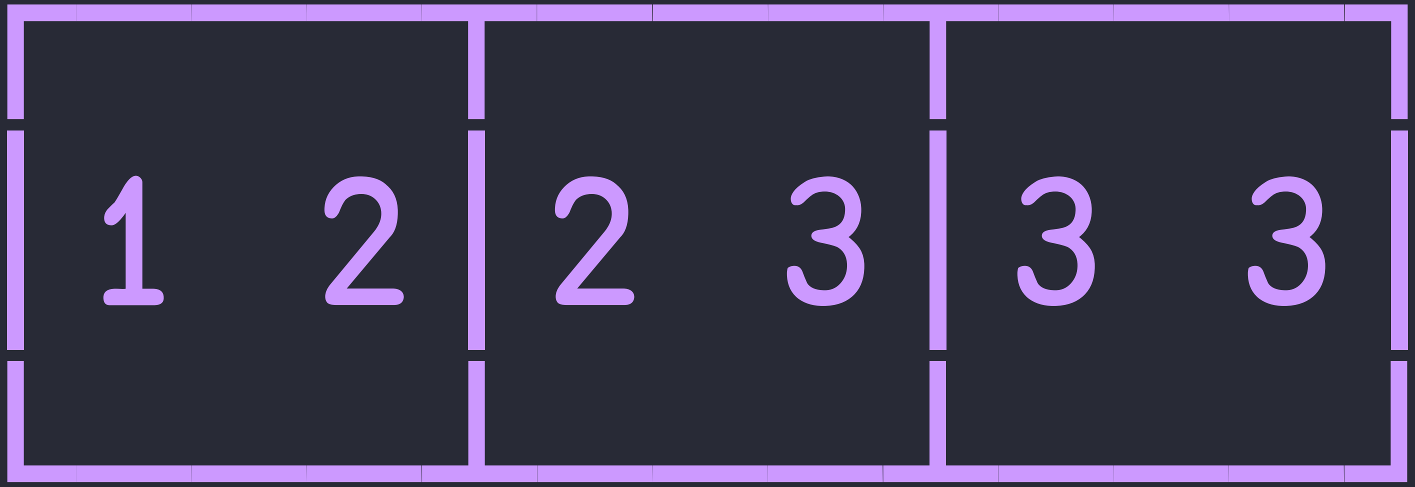
SOLUTION #3

FREE

```
∧/2=/⊢
```

```
2=/1 2 3 3
```

```
2,/1 2 3 3
```



1 2	2 3	3 3

```
2 ,/1 2 2 3 3
```

```
┌───┬───┬───┐
│1 2│2 3│3 3│
└───┴───┴───┘
```

```
2=/1 2 3 3
```

1 2	2 3	3 3

2≠1 2 3 3

1=2	2=3	3=3

2=/1 2 3 3

0	0	1

2 ≠ 1 2 3 3

1=2	2=3	3=3

2+/1 2 3 3

1+2	2+3	3+3

2+/1 2 3 3

3	5	6

`pandas.DataFrame.rolling()` pandas

`cudf.Series.rolling()` RAPIDS

`cudf::rolling_window()` RAPIDS

```
2=/1 2 3 3
```

0 0 1

```
^/0 0 1
```

0

FREE

```
∧/2=/⊢
```

FREE

```
∧/2=/⊢
```

∧	and
=	equal
⊢	same
/	reduce

SOLUTION #4

FREE

```
⌊/=⌈/
```

```
L/1 2 3 4
```

1

```
⌊/1 2 3 4
```

```
⌈/1 2 3 4
```

4

```
{(⌊/⍵)=⌈/⍵}
```

```
{(⌊/⍵)=(⌈/⍵)}
```

FREE

```
⌊/=⌈/
```

FREE

```
⌊/=⌈/
```

```
⌊ min
⌈ max
= equal
/ reduce
```

/	reduce	3
=	equal	3
∧	and	2
⊢	same	2
⊃	first	1
∪	unique	1
≥	gte	1
≢	tally	1
⌊	min	1
⌈	max	1

SOLUTION #5

FREE

```
(1≥≢)⊣⌸
```

```
{αω}⌸7 8 8 9 9 9
```

```
{⍺⍵}⌸7 8 8 9 9 9
```

```
┌─┬─────┐
│7│1    │
├─┼─────┤
│8│2 3  │
├─┼─────┤
│9│4 5 6│
└─┴─────┘
```

```
{⍺}⌸7 8 8 9 9 9

        7 8 9
```

7 8 8 9 9 9

7 8 9

```
(1≥≠)⊣⌸7 8 8 9 9 9

0
```

FREE

```
(1≥≠)⊣⌸
```

SOLUTION #6

FREE

```
(⊃∧.=⊢)1 2 3
```

```
(⊃1 2 3)∧.=(⊢1 2 3)
```

```
(⊃1 2 3)∧.=1 2 3
```

```
1∧.=1 2 3
```

```
∧.=

1 1 1
1 2 3
```

^ . =

100

```
^.=
```

```
0
```

SOLUTIONS

```
#1      ∧/⊃=⊢
#2      (1=≢)∪
#3      ∧/2=/⊢
#4      ⌊/=⌈/
#5      (1≥≢)⊣⌸
#6      ⊃∧.=⊢      ⍝ Aaron Hsu
#7      ⍋≡⍒        ⍝ Adám Brudzewsky
#8      1∘⌽≡⊢      ⍝ Bob Therriault
```

```
     SOLUTIONS              PROFILING (cmpx)
#1     ∧/⊃=⊢          0% ⎕⎕⎕⎕⎕⎕
#2    (1=≢)∪        -32% ⎕⎕⎕⎕
#3    ∧/2=/⊢         -6% ⎕⎕⎕⎕⎕⎕
#4     ⌊/=⌈/         -4% ⎕⎕⎕⎕⎕⎕
#5   (1≥≢)⊣⌸        -24% ⎕⎕⎕⎕⎕
#6     ⊃∧.=⊢        -49% ⎕⎕⎕
#7      ⍋≡⍒        +558% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
#8     1∘⌽≡⊢        -24% ⎕⎕⎕⎕⎕
```

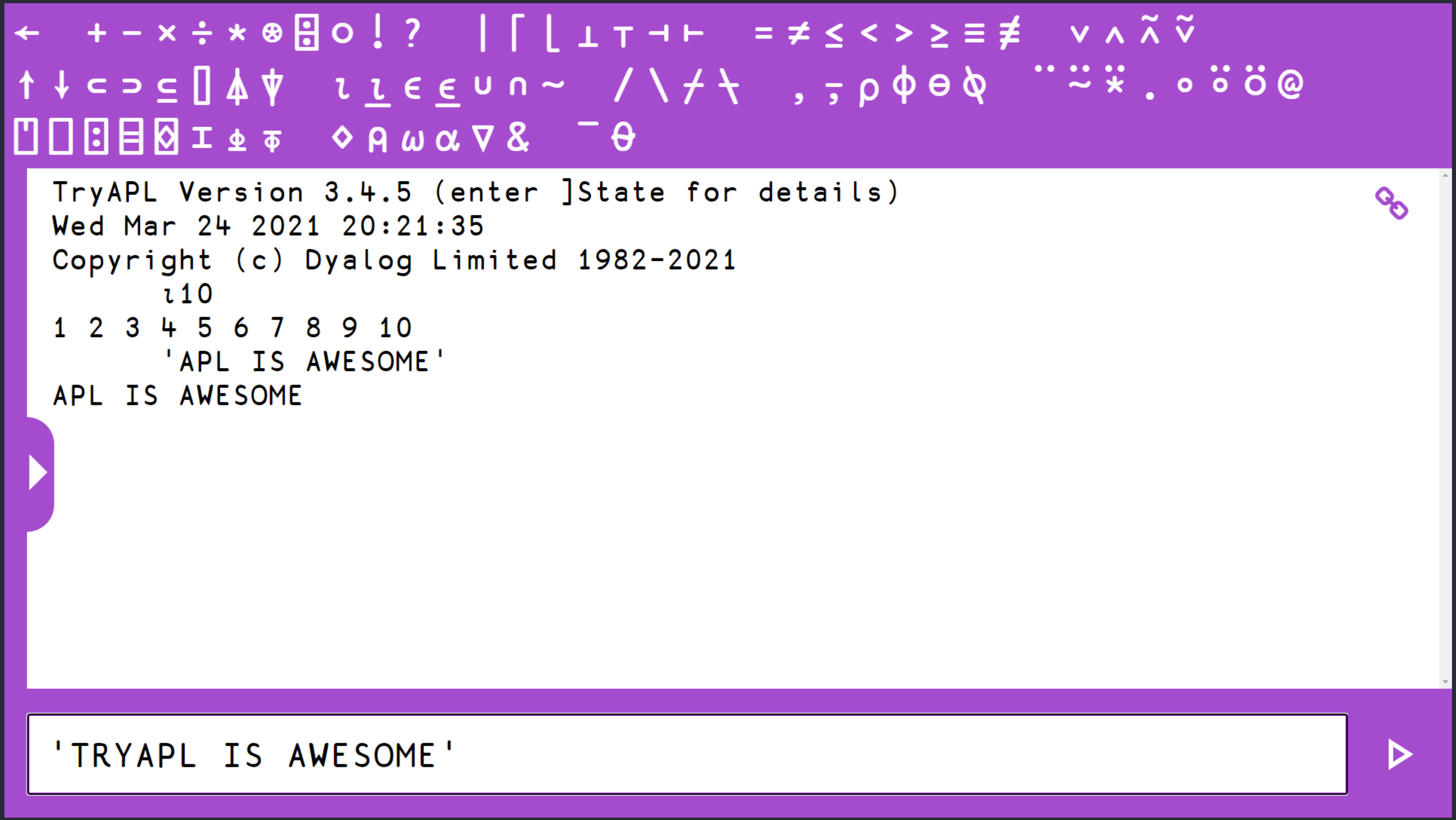
```
   SOLUTIONS                  PROFILING (cmpx)
#1      ∧/⊃=⊢          0% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
#2     (1=≢)∪        -29% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□
#3     ∧/2=/⊢         -6% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
#4      ⌊/=⌈/         -9% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
#5     (1≥≢)⊣⌸       -23% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
#6      ⊃∧.=⊢        -50% □□□□□□□□□□□□□□□□□□□□
#7      ⍋≡⍒
#8      1∘⌽≡⊢        -25% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

/	reduce	3
=	equal	3
∧	and	2
⊢	same	2
⊃	first	1
∪	unique	1
≥	gte	1
≢	tally	1
⌊	min	1
⌈	max	1

=	equal	4
⊢	same	4
/	reduce	3
∧	and	3
⊃	first	2
≥	gte	2
≢	tally	2
∪	unique	1
⌊	min	1
⌈	max	1
⊣	left	1
⌸	key	1
.	inner	1

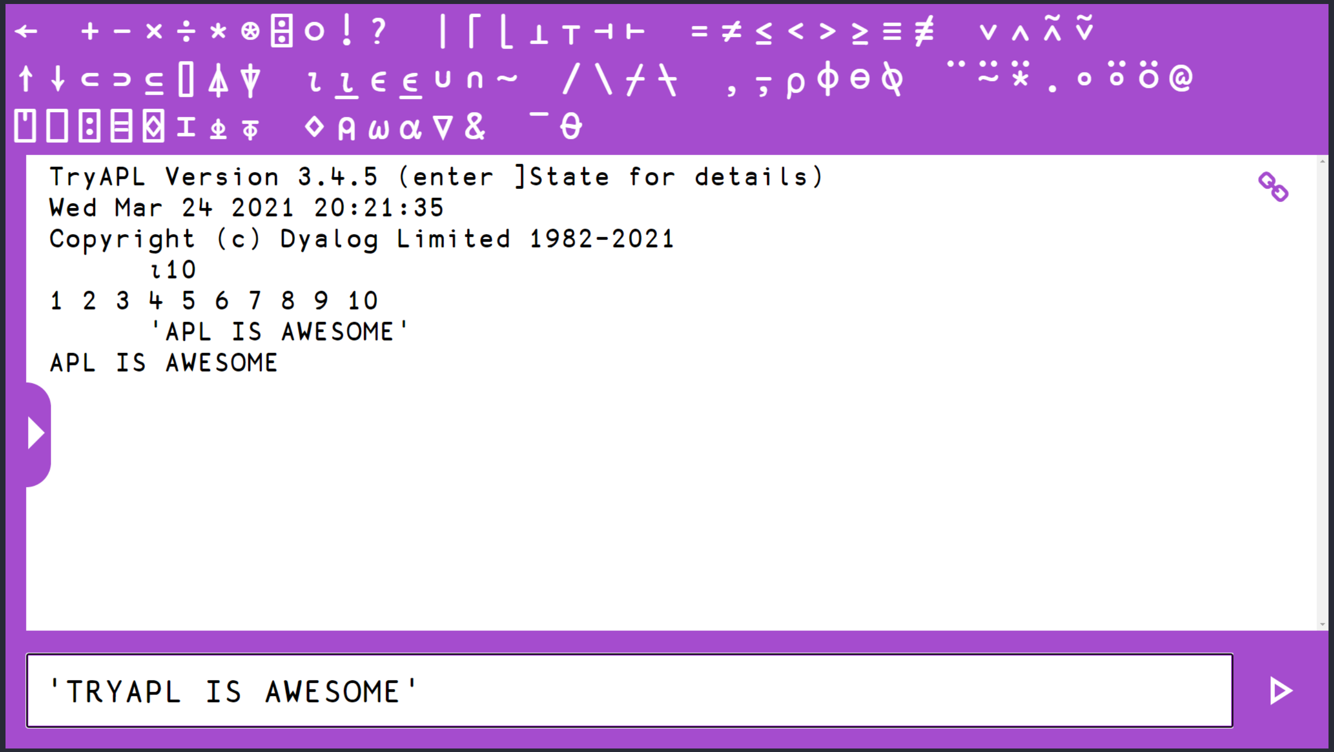
ALG⌽RITHMS AS A T⌽⌽L ⌽F TH⌽UGHT

=	equal	4
⊢	same	4
/	reduce	3
∧	and	3
⊃	first	2
≥	gte	2
≢	tally	2
∪	unique	1
⌊	min	1
⌈	max	1
⊣	left	1
⌸	key	1
.	inner	1



```
TryAPL Version 3.4.5 (enter ]State for details)
Wed Mar 24 2021 20:21:35
Copyright (c) Dyalog Limited 1982-2021
      ⍳10
1 2 3 4 5 6 7 8 9 10
      'APL IS AWESOME'
APL IS AWESOME
```

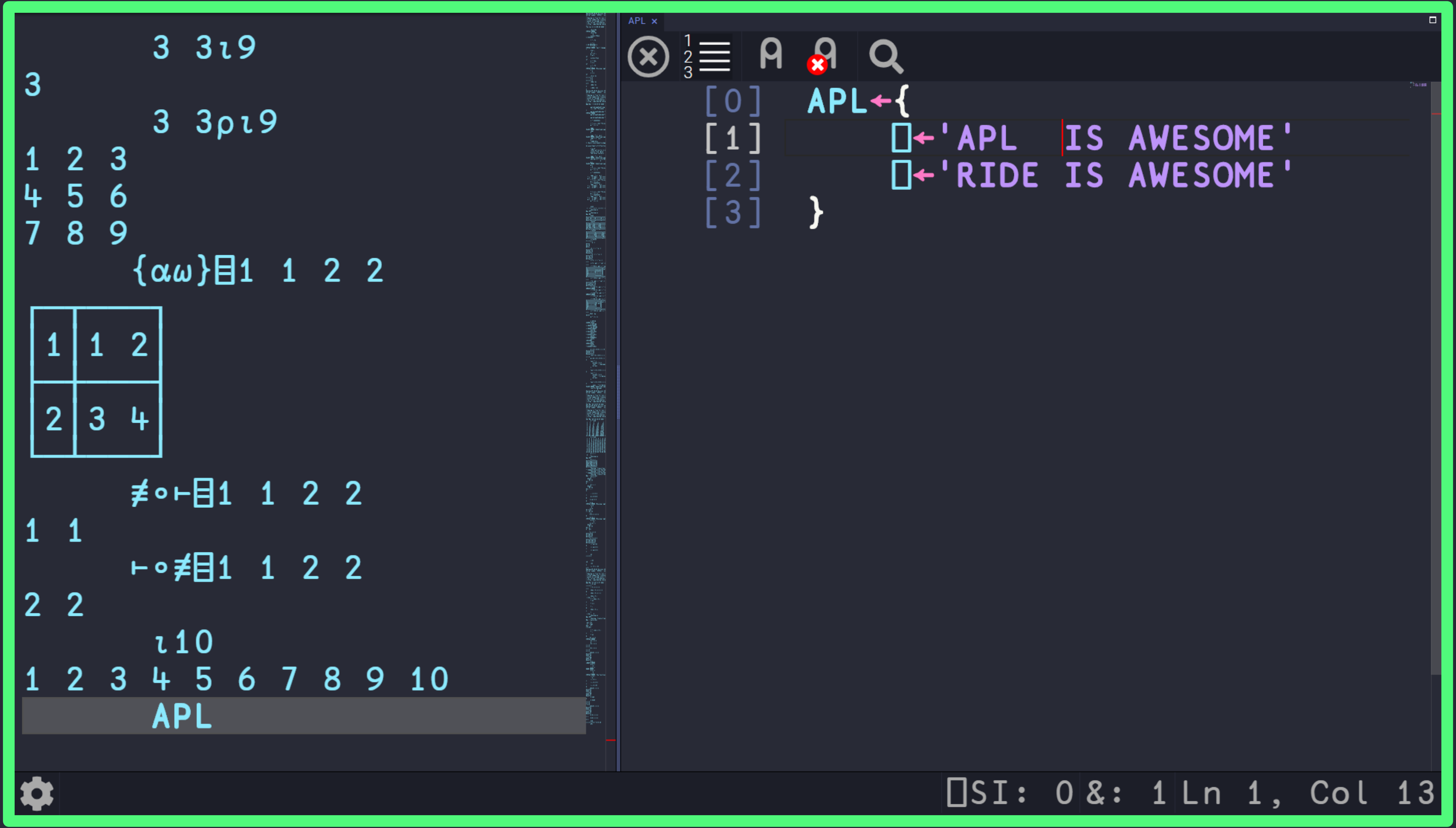
```
'TRYAPL IS AWESOME'
```



```
TryAPL Version 3.4.5 (enter ]State for details)
Wed Mar 24 2021 20:21:35
Copyright (c) Dyalog Limited 1982-2021
      ⍳10
1 2 3 4 5 6 7 8 9 10
      'APL IS AWESOME'
APL IS AWESOME
```

```
'TRYAPL IS AWESOME'
```

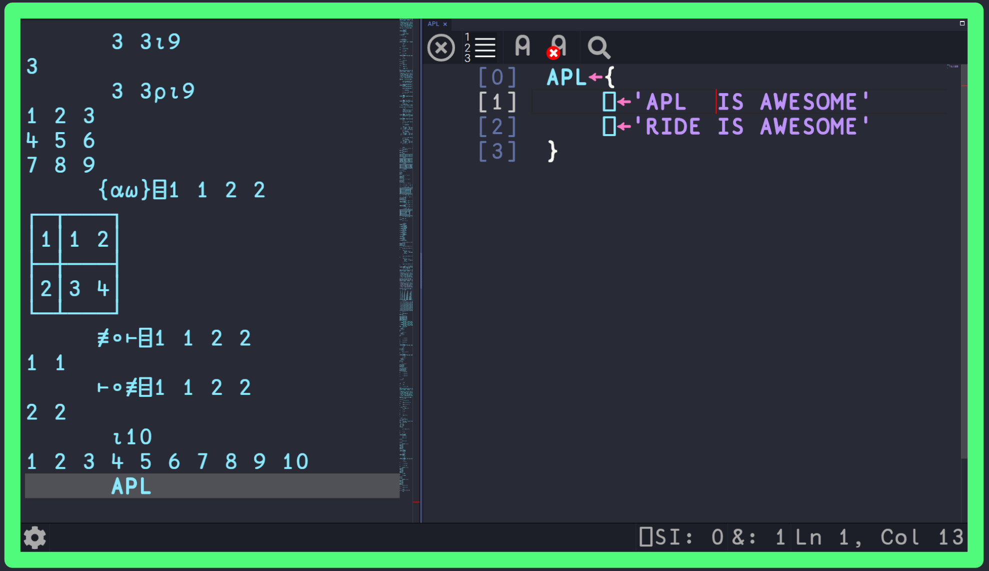
https://tryapl.org/



```
      3 3⍳9
3
      3 3⍴⍳9
1 2 3
4 5 6
7 8 9
      {⍺⍵}⌸1 1 2 2
┌─┬───┐
│1│1 2│
├─┼───┤
│2│3 4│
└─┴───┘
      ≢∘⊢⌸1 1 2 2
1 1
      ⊢∘≢⌸1 1 2 2
2 2
      ⍳10
1 2 3 4 5 6 7 8 9 10
      APL
```

```
APL←{
    ⎕←'APL  IS AWESOME'
    ⎕←'RIDE IS AWESOME'
}
```

⎕SI: 0 &: 1 Ln 1, Col 13

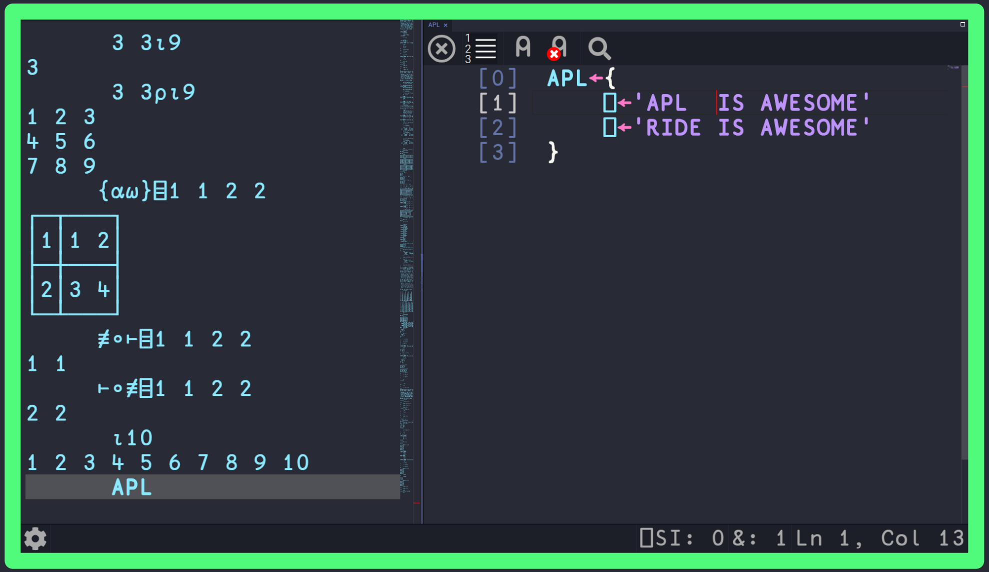


```
      3 3⍳9
3
      3 3⍴⍳9
1 2 3
4 5 6
7 8 9
     {⍺⍵}⌸1 1 2 2
┌─┬───┐
│1│1 2│
├─┼───┤
│2│3 4│
└─┴───┘
     ≢∘⊢⌸1 1 2 2
1 1
     ⊢∘≢⌸1 1 2 2
2 2
      ⍳10
1 2 3 4 5 6 7 8 9 10
      APL
```

```
APL←{
    ⎕←'APL  IS AWESOME'
    ⎕←'RIDE IS AWESOME'
}
```

```
⎕SI: 0 &: 1 Ln 1, Col 13
```

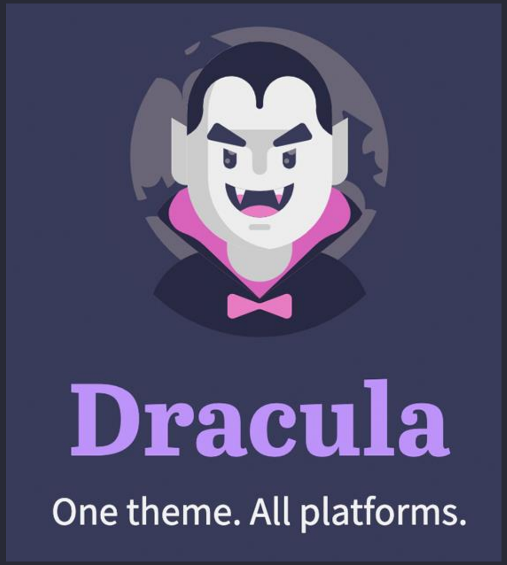
https://www.dyalog.com/
https://github.com/Dyalog/ride



```
      3 3⍳9
3
      3 3⍴⍳9
1 2 3
4 5 6
7 8 9
      {⍺⍵}⌸1 1 2 2
┌─┬───┐
│1│1 2│
├─┼───┤
│2│3 4│
└─┴───┘
      ≢∘⊢⌸1 1 2 2
1 1
      ⊢∘≢⌸1 1 2 2
2 2
      ⍳10
1 2 3 4 5 6 7 8 9 10
      APL
```

```
[0]   APL←{
[1]       ⎕←'APL  IS AWESOME'
[2]       ⎕←'RIDE IS AWESOME'
[3]   }
```

```
⎕SI: 0 &: 1 Ln 1, Col 13
```



https://www.dyalog.com/
https://github.com/Dyalog/ride

Thank You!

Conor Hoekstra

code_report

NVIDIA RAPIDS

#include

Questions?

Conor Hoekstra

code_report

NVIDIA RAPIDS

#include