Pensieve header: A permutations package for March 30, 2016.

From http://www.math.toronto.edu/drorbn/classes/16-1750-ShamelessMathematica/About.html: **Possible Topics** (in no particular order). Whatever you may suggest, and the ~~Fibonacci numbers~~; ~~the Jones polynomial~~; ~~a more efficient Jones-algorithm~~; ~~a riddle on spheres~~; ~~Khovanov homology~~; Γ-calculus; the Hopf fibration; Hilbert's 13th problem; ~~non-commutative Gaussian elimination~~; free Lie algebras; the Baker-Campbell-Hausdorff formula; wacky numbers; ~~an order 4 torus~~; ~~the Schwarz Lantern~~; knot colourings; the Temperley-Lieb pairing; the dodecahedral link; sound experiments; barycentric subdivisions; ~~a Peano curve;~~ braid closures and Vogel's algorithm; ~~the insolubility of the quintic~~; phase portraits; ~~the Mamdelbrot set~~; shadows of the Cantor Aerogel; quilt plots; ~~some image transformations~~; De Bruijn graphs; ~~the Riemann series theorem~~; finite type invariants and the Willerton fish.

The Issue from Last Time

```
?? PermutationProduct
```

PermutationProduct[*a*, *b*, *c*] gives the product of permutations *a*, *b*, *c*. ≫

```
Attributes[PermutationProduct] = {Flat, OneIdentity, Protected}
```

```
? Flat
```

Flat is an attribute that can be assigned to a symbol *f* to indicate that all expressions involving nested functions *f* should be flattened out. This property is accounted for in pattern matching. ≫

```
? OneIdentity
```

OneIdentity is an attribute that can be assigned to a symbol *f* to indicate that *f*[*x*], *f*[*f*[*x*]], etc. are all equivalent to *x* for the purpose of pattern matching. ≫

```
SetAttributes[f, Flat];
f[x_] := x;
```

```
f[x]
```

$IterationLimit::itlim : Iteration limit of 4096 exceeded. ≫

```
Hold[f[x]]
```

```
f[x, y]
```

$IterationLimit::itlim : Iteration limit of 4096 exceeded. ≫

```
Hold[f[x, y]]
```

The Package

```
BeginPackage["Permutations`"]
```

```
Permutations`
```

```
Permutation::usage =
  "Permutation[i1,i2,...] represents a permutation mapping 1->i1, 2->i2, ...";

Begin["`Private`"]
```

```
Permutations`Private`
```

```
Permutation /: PermutationProduct[a_Permutation, b_Permutation,
    rest___Permutation] := PermutationProduct[b[[List @@ a]], rest];

Permutation /: InversePermutation[a_Permutation] :=
 (t = a; Do[t[[a[[i]]]] = i, {i, Length@a}]; t)

Permutation /: PermutationSupport[a_Permutation] := List @@ Select[a, # ≠ a[[#]] &]

Permutation /: PermutationReplace[expr_, a_Permutation] :=
 expr /. Table[i → a[[i]], {i, a}]

End[]; EndPackage[]
```

Basic Testing

```
PermutationProduct[Permutation[3, 2, 1]]
```

```
Permutation[3, 2, 1]
```

```
⊙
```

```
Permutation[3, 2, 1] ⊙ Permutation[2, 1, 3]
```

```
Permutation[3, 1, 2]
```

```
PermutationProduct[dror]
```

```
dror
```

To do

~~Use the head "Permutation" to represent permutations!~~
~~Use old operation names "PermutationProduct", "InversePermutation", "PermutationSupport", "PermutationReplace"!~~

Exercises:
1. Allow mixing old and new notations!
2. Complete the two commands below:

```
? PermutationList
```

PermutationList[*perm*] returns a permutation list representation of permutation *perm*.

PermutationList[*perm*, *len*] returns a permutation list of length *len*. >>

```
? PermutationCycles
```

PermutationCycles[*perm*] gives a disjoint cycle representation of permutation *perm*. ≫