```
BeginPackage["Perm`"];
Perm::usage =
  "Perm[5,2,3,1,4] means the permutation that maps 1→5, 2→2, 3→3, 4→1, 5→4. Perm[c]
    may also be used to convert a permutation c from cycles into standard form";
ComposePerm::usage = "ComposePerm[σ,τ] where σ and τ are permutations
    returns the permutation obtained by composing σ and τ where both
    σ and τ may be in cycle form or standard form. Alternatively
    ComposePerm[σ,A] will return the expression obtained by applying
    permutation σ to expression A. The alternate notation is σ∘τ";
PermPower::usage = "PermPower[τ,n] returns the permutaiton obtained
    by taking τ to n-th power. Alternate notation is τ^n";
PermutationQ::usage = "PermutationQ[τ] Returns True if τ is
    a permutation and false otherwise";
IdentityPerm::usage = "IdenityPerm[n] returns the identity permution of length n";
FindPerm::usage =
  "FindPerm[a,b] returns the permutation that takes expression a to expression b";
RandomPerm::usage = "RandomPerm[n] returns a random permutation of length n";
PermOrder::usage = "PermOrder[τ] returns the order of a permutation";
PermList::usage = "PermList[n] creates a list of all permutations of length n";


Begin["`private`"];

PermutationQ[σ_Perm] := Sort[List @@ σ] === Range[Length[σ]];
Perm /: σ_Perm ∘ τ_Perm /; Length[σ] == Length[τ] := σ[[List @@ τ]];
Perm /: (σ_Perm)^0 /; PermutationQ[σ] := IdentityPerm[Length[σ]];
Perm /: (σ_Perm)^n_Integer /; PermutationQ[σ] ∧ n ≥ 1 := σ ∘ σ^(n - 1);

Perm /: (σ_Perm)^-1 /; PermutationQ[σ] := (τ = σ;
Do[τ[[σ[[i]]]] = i, {i, Length[σ]}];
τ
);

IdentityPerm[n_Integer] := Perm @@ Range[n]

Perm /: (σ_Perm)^n_Integer /; PermutationQ[σ] ∧ n ≤ -1 := (σ^-n)^-1;

Perm[c_Cycles] := (τ1 = Range[Max[List @@ c]];
For[i = 1, i ≤ Length[c[[1]]], i++,
For[j = 1, j ≤ Length[c[[1]][[i]]], j++,
τ1 = ReplacePart[τ1,
       c[[1]][[i]][[j]] -> c[[1]][[i]][[If[j == Length[c[[1]][[i]]], 1, j + 1]]]];
]
];
Perm @@ τ1
)

σ_Perm ∘ τ_Cycles /; Length[σ] == Length[Perm[τ]] := σ ∘ Perm[τ];
τ_Cycles ∘ σ_Perm /; Length[σ] == Length[Perm[τ]] := Perm[τ] ∘ σ;
```

```
τ_Perm ∘ σ_ /; Length[σ] == Length[τ] := σ[[List @@ τ]];

ComposePerm[σ_, τ_] := σ ∘ τ
PermPower[τ_Perm, n_Integer] := τ^n

FindPerm[a_, b_] /;
  Sort[a] == Sort[b] And Length[Union[a]] == Length[a] And Length[Union[b]] == Length[b] :=
 Perm @@ Table[Position[a, n][[1]][[1]], {n, b}]

RandomPerm[n_Integer] := Module[{list = Range[n], a, τ2 = {}},
For[i = 1, i ≤ n, i++,
a = RandomChoice[list];
τ2 = Append[τ2, a];
list = DeleteCases[list, a]
];
Perm @@ τ2
];

PermOrder[p_Perm] := (τ3 = p;
  For[n = 1, List @@ τ3 ≠ Range[Length[p]], n++, τ3 = τ3 ∘ p];
  n
 )

PermList[1] := {Perm[1]}
PermList[n_Integer] :=
 Flatten@Table[Insert[i, n, j], {i, PermList[n - 1]}, {j, Range[n]}]


End[];
EndPackage[];
τ

PermutationQ[Perm[1, 2, 3, 4]]
True

Perm[1, 2, 3, 4] ∘ Perm[2, 1, 4, 3]
Perm[1, 4, 2, 3]^-1
Perm[4, 3, 2, 1]^4
Perm[2, 1, 4, 3]

Perm[1, 4, 2, 3]

Perm[1, 2, 3, 4]

IdentityPerm[4]
Perm[2, 1, 4, 3]^-3
Perm[1, 2, 3, 4]

Perm[2, 1, 4, 3]
```

```
r = RandomPermutation[4]
Perm[r]
r ∘ Perm[2, 1, 3, 4]
```

```
Cycles[{{1, 2, 3, 4}}]
```

```
Perm[2, 3, 4, 1]
```

```
Perm[3, 2, 4, 1]
```

```
s = RandomPerm[4]
PermOrder[s]
s[[2]]
s ∘ {a, b, c, d}
```

```
Perm[1, 3, 2, 4]
```

```
2
```

```
3
```

```
{a, c, b, d}
```

```
FindPerm[{a, b, c, d}, {b, a, c, d}]
```

```
FindPerm[{a, b, c, d}, {b, a, c, d}]
```

```
PermList[5]
```

```
{Perm[5, 4, 3, 2, 1], Perm[4, 5, 3, 2, 1], Perm[4, 3, 5, 2, 1], Perm[4, 3, 2, 5, 1],
 Perm[4, 3, 2, 1, 5], Perm[5, 3, 4, 2, 1], Perm[3, 5, 4, 2, 1], Perm[3, 4, 5, 2, 1],
 Perm[3, 4, 2, 5, 1], Perm[3, 4, 2, 1, 5], Perm[5, 3, 2, 4, 1], Perm[3, 5, 2, 4, 1],
 Perm[3, 2, 5, 4, 1], Perm[3, 2, 4, 5, 1], Perm[3, 2, 4, 1, 5], Perm[5, 3, 2, 1, 4],
 Perm[3, 5, 2, 1, 4], Perm[3, 2, 5, 1, 4], Perm[3, 2, 1, 5, 4], Perm[3, 2, 1, 4, 5],
 Perm[5, 4, 2, 3, 1], Perm[4, 5, 2, 3, 1], Perm[4, 2, 5, 3, 1], Perm[4, 2, 3, 5, 1],
 Perm[4, 2, 3, 1, 5], Perm[5, 2, 4, 3, 1], Perm[2, 5, 4, 3, 1], Perm[2, 4, 5, 3, 1],
 Perm[2, 4, 3, 5, 1], Perm[2, 4, 3, 1, 5], Perm[5, 2, 3, 4, 1], Perm[2, 5, 3, 4, 1],
 Perm[2, 3, 5, 4, 1], Perm[2, 3, 4, 5, 1], Perm[2, 3, 4, 1, 5], Perm[5, 2, 3, 1, 4],
 Perm[2, 5, 3, 1, 4], Perm[2, 3, 5, 1, 4], Perm[2, 3, 1, 5, 4], Perm[2, 3, 1, 4, 5],
 Perm[5, 4, 2, 1, 3], Perm[4, 5, 2, 1, 3], Perm[4, 2, 5, 1, 3], Perm[4, 2, 1, 5, 3],
 Perm[4, 2, 1, 3, 5], Perm[5, 2, 4, 1, 3], Perm[2, 5, 4, 1, 3], Perm[2, 4, 5, 1, 3],
 Perm[2, 4, 1, 5, 3], Perm[2, 4, 1, 3, 5], Perm[5, 2, 1, 4, 3], Perm[2, 5, 1, 4, 3],
 Perm[2, 1, 5, 4, 3], Perm[2, 1, 4, 5, 3], Perm[2, 1, 4, 3, 5], Perm[5, 2, 1, 3, 4],
 Perm[2, 5, 1, 3, 4], Perm[2, 1, 5, 3, 4], Perm[2, 1, 3, 5, 4], Perm[2, 1, 3, 4, 5],
 Perm[5, 4, 3, 1, 2], Perm[4, 5, 3, 1, 2], Perm[4, 3, 5, 1, 2], Perm[4, 3, 1, 5, 2],
 Perm[4, 3, 1, 2, 5], Perm[5, 3, 4, 1, 2], Perm[3, 5, 4, 1, 2], Perm[3, 4, 5, 1, 2],
 Perm[3, 4, 1, 5, 2], Perm[3, 4, 1, 2, 5], Perm[5, 3, 1, 4, 2], Perm[3, 5, 1, 4, 2],
 Perm[3, 1, 5, 4, 2], Perm[3, 1, 4, 5, 2], Perm[3, 1, 4, 2, 5], Perm[5, 3, 1, 2, 4],
 Perm[3, 5, 1, 2, 4], Perm[3, 1, 5, 2, 4], Perm[3, 1, 2, 5, 4], Perm[3, 1, 2, 4, 5],
 Perm[5, 4, 1, 3, 2], Perm[4, 5, 1, 3, 2], Perm[4, 1, 5, 3, 2], Perm[4, 1, 3, 5, 2],
 Perm[4, 1, 3, 2, 5], Perm[5, 1, 4, 3, 2], Perm[1, 5, 4, 3, 2], Perm[1, 4, 5, 3, 2],
 Perm[1, 4, 3, 5, 2], Perm[1, 4, 3, 2, 5], Perm[5, 1, 3, 4, 2], Perm[1, 5, 3, 4, 2],
 Perm[1, 3, 5, 4, 2], Perm[1, 3, 4, 5, 2], Perm[1, 3, 4, 2, 5], Perm[5, 1, 3, 2, 4],
 Perm[1, 5, 3, 2, 4], Perm[1, 3, 5, 2, 4], Perm[1, 3, 2, 5, 4], Perm[1, 3, 2, 4, 5],
 Perm[5, 4, 1, 2, 3], Perm[4, 5, 1, 2, 3], Perm[4, 1, 5, 2, 3], Perm[4, 1, 2, 5, 3],
 Perm[4, 1, 2, 3, 5], Perm[5, 1, 4, 2, 3], Perm[1, 5, 4, 2, 3], Perm[1, 4, 5, 2, 3],
 Perm[1, 4, 2, 5, 3], Perm[1, 4, 2, 3, 5], Perm[5, 1, 2, 4, 3], Perm[1, 5, 2, 4, 3],
 Perm[1, 2, 5, 4, 3], Perm[1, 2, 4, 5, 3], Perm[1, 2, 4, 3, 5], Perm[5, 1, 2, 3, 4],
 Perm[1, 5, 2, 3, 4], Perm[1, 2, 5, 3, 4], Perm[1, 2, 3, 5, 4], Perm[1, 2, 3, 4, 5]}
```