```
BeginPackage["Perm`"]
```

```
Perm`
```

```
Perm::usage = "Perm[2,5,6,4] means the permutation that maps 1→2, 2→5, 3→6, 4→4.
   \n Perm[2,3,1\!\(\*SuperscriptBox[\(]\), \(-1\)]\) return the inverse of
   Perm[2,3,1]. \n Perm[c_Cycles, n_Integer]: converts a permutaion in cycle form
   to a permutation with head Perm where n is the size of the set being permutated. "
```

```
Perm[2,5,6,4] means the permutation that maps 1→2, 2→5, 3→6, 4→4.
 Perm[2,3,1]^-1 return the inverse of Perm[2,3,1].
 Perm[c_Cycles, n_Integer]: converts a permutaion in cycle form to a
  permutation with head Perm where n is the size of the set being permutated.
```

```
IdentityPermutation::usage =
 "IdentityPermutation[n_Integer] returns the identity permutation."
```

```
IdentityPermutation[n_Integer] returns the identity permutation.
```

```
PermutationsQ::usage =
 "PermutationsQ[σ_Perm] gives True if σ is a permutation, and False otherwise."
```

```
PermutationsQ[σ_Perm] gives True if σ is a permutation, and False otherwise.
```

```
Pivot::usage = "Pivot[σ_Perm] returns the minimum
   of the element moved by σ. This element is called the pivot of σ."
```

```
Pivot[σ_Perm] returns the minimum of the
  element moved by σ. This element is called the pivot of σ.
```

```
ElementsMoved::usage = "ElementsMoved[σ_Perm] returns the elements moved by σ "
```

```
ElementsMoved[σ_Perm] returns the elements moved by σ
```

```
Cycle::usage =
 "Cycle[p_Perm] converts permutaions with head Perm to cycle form with head Cycles."
```

```
Cycle[p_Perm] converts permutaions with head Perm to cycle form with head Cycles.
```

```
ProductTranspositions::usage =
 "ProductTranspositions[σ_Perm] converts a σ to products of transpostions."
```

```
ProductTranspositions[σ_Perm] converts a σ to products of transpostions.
```

```
EvenPermutationQ::usage =
 "EvenPermutationQ[σ_Perm] gives True if σ is an even permutation, and False otherwise"
```

```
EvenPermutationQ[σ_Perm] gives True if σ is an even permutation, and False otherwise
```

```
Begin["`private`"]
```

```
Perm`private`
```

```
PermutationsQ[σ_Perm] := Sort[List @@ σ] === Range[Length[σ]];
```

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

```
Perm /: (σ_Perm)^-1 /; PermutationsQ[σ] := (
τ = σ;
τ[[σ[[#]]]] & /@ Perm @@ Range[Length[σ]]
)

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

Perm[Cycles[{{}}], n_Integer] := IdentityPermutation[n]
Perm[c_Cycles, n_Integer] := Module[{s, v},
s = List @@ c[[1]];
v = ConstantArray[0, n];
For[i = 1, i <= Length[s], i++,
{If[Position[s[[i]], #][[1, 1]] === Length[s[[i]]], v[[#]] = s[[i]][[1]],
       v[[#]] = s[[i]][[Position[s[[i]], #][[1, 1]] + 1]]] } & /@ s[[i]]
];
If[# === 0, v[[Position[v, #][[1, 1]]]] = Position[v, #][[1, 1]]] & /@ v;
Return[Perm @@ v]
]

Cycle[p_Perm] /; PermutationsQ[p] := Module[{ls, cyc, tmp},
ls = List @@ p;
tmp = ls /. i_Integer :> {Position[ls, i][[1, 1]], i} /. {l_, l_} :> {} ;
cyc = tmp //.
    {r___, {k_, g___, l_}, s___, {l_, w___, m_}, t___} :> {r, s, {k, g, l, w, m}, t};
Cycles @@ {DeleteDuplicates[#] & /@ DeleteCases[cyc, {}]}
]

ElementsMoved[σ_Perm] /; PermutationsQ[σ] := (
elmv =
   Position[σ, #] & /@ Select[List @@ σ, Position[σ, #][[1, 1]] != # &] // Flatten;
elmv
)

Pivot[σ_Perm] /; PermutationsQ[σ] := (
spt =  Position[σ, #] & /@ Select[List @@ σ, Position[σ, #][[1, 1]] != # &] // Flatten;
If[spt != {}, Return[spt // Min]]
)

ProductTranspositions[σ_Perm] := Module[{g, t},
g = Tuples[{{#[[1]]}, Reverse[#[[2 ;;]]]}] & /@ Cycle[σ][[1]];
t = Flatten[g /. {a_Integer, j_Integer} :> v[a, j]];
List @@ # & /@ t
]

EvenPermutationQ[σ_Perm] := EvenQ[Length[ProductTranspositions[σ]]]

End[]; EndPackage[]

r = RandomPermutation[10]

Cycles[{{1, 6, 4, 3, 8, 2}, {9, 10}}]

p = Perm[r, 10]

Perm[6, 1, 8, 3, 5, 4, 7, 2, 10, 9]
```

```
Pivot[p]
```

```
1
```

```
p^-1
```

```
Perm[4, 6, 2, 8, 5, 3, 7, 1, 9, 10]
```

```
Perm[8, 6, 5, 2, 4, 3, 1, 9, 10, 7]
```

```
Perm[8, 6, 5, 2, 4, 3, 1, 9, 10, 7]
```

```
Cycle[p]
```

```
Cycles[{{1, 6, 4, 3, 8, 2}, {9, 10}}]
```

```
a = ElementsMoved[p]
```

```
{1, 2, 3, 4, 6, 8, 9, 10}
```

```
ProductTranspositions[p]
```

```
{{1, 2}, {1, 8}, {1, 3}, {1, 4}, {1, 6}, {9, 10}}
```

```
EvenPermutationQ[p]
```

```
True
```