```
> with(ArrayTools)
```

[AddAlongDimension, Alias, AllNonZero, AnyNonZeros, Append, BlockCopy, CircularShift, ComplexAsFloat, Compress, Concatenate, Copy, DataTranspose, Diagonal, Dimensions, ElementDivide, ElementMultiply, ElementPower, Extend, Fill, FlipDimension, GeneralInnerProduct, GeneralOuterProduct, HasNonZero, HasZero, Insert, IsEqual, IsMonotonic, IsZero, Lookup, LowerTriangle, MultiplyAlongDimension, NumElems, Partition, Permute, PermuteInverse, RandomArray, ReduceAlongDimension, RegularArray, Remove, RemoveSingletonDimensions, Replicate, Reshape, Reverse, ScanAlongDimension, SearchArray, Size, SuggestedDatatype, SuggestedOrder, SuggestedSubtype, Uncompress, UpperTriangle] **(1)**

```
> triapile := proc(T)
  local n, pile, taillepile, s, c, trie, i;
  n := Size(T)[2];
  trie := Array(1..n); #Tableau qui contient la permutation triée
  pile := Array(1..n);
  pile[1] := T[1];
  taillepile := 1;
  s := 2; #Compteur qui parcourt T
  c := 1;
  while s ≤ n do
    if (taillepile = 0 or T[s] < pile[taillepile]) then
        taillepile := taillepile + 1;
        pile[taillepile] := T[s]; #On empile
        s := s + 1
  else trie[c] := pile[taillepile]; taillepile := taillepile - 1; c := c + 1 #On dépile
  fi;
  od;
  for i from 1 to taillepile do #Quand on a fini de parcourir T on termine de vider la pile
      trie[c + i - 1] := pile[taillepile + 1 - i]
  od:
  trie
  end:
```

```
> T := Array([4, 2, 3, 1, 5])
```

$$T := \begin{bmatrix} 4 & 2 & 3 & 1 & 5 \end{bmatrix} \quad \textbf{(2)}$$

```
> triapile(T)
```

$$\begin{bmatrix} 2 & 1 & 3 & 4 & 5 \end{bmatrix} \quad \textbf{(3)}$$

```
> est_trie := proc(T)
  local i;
  for i from 1 to Size(T)[2] - 1 do
      if T[i] > T[i + 1] then
          return false fi; od;
  true
  end:
```

```
> est_trie([ 2  1  3  4  5 ])
```

false **(4)**

```
> perms := proc(n)
  local L, i, j, k, T, M, c;
  # Procédure qui génère toutes les permutations de taille n
  if n = 1 then M := [Array([1])]
  else
  L := perms(n - 1);
  M := [ ];
  for i from 1 to nops(L) do #On parcourt les permutations de longueur n-1
        for j from 1 to n do
            T := Array(1 ..n);
            c := 1;
        for k from 1 to n do
            if k = j then T[k] := n # On insère n à l'emplacement j
           else T[k] := L[i][c]; c := c + 1 fi;
        od;
        M := [op(M), T];
  od;od;
  fi;
  M;
  end proc:
```

```
> perms(4)
```

$$\left[\left[4\ 3\ 2\ 1 \right], \left[3\ 4\ 2\ 1 \right], \left[3\ 2\ 4\ 1 \right], \left[3\ 2\ 1\ 4 \right], \left[4\ 2\ 3\ 1 \right],\right.$$
$$\left[2\ 4\ 3\ 1 \right], \left[2\ 3\ 4\ 1 \right], \left[2\ 3\ 1\ 4 \right], \left[4\ 2\ 1\ 3 \right], \left[2\ 4\ 1\ 3 \right],$$
$$\left[2\ 1\ 4\ 3 \right], \left[2\ 1\ 3\ 4 \right], \left[4\ 3\ 1\ 2 \right], \left[3\ 4\ 1\ 2 \right], \left[3\ 1\ 4\ 2 \right],$$
$$\left[3\ 1\ 2\ 4 \right], \left[4\ 1\ 3\ 2 \right], \left[1\ 4\ 3\ 2 \right], \left[1\ 3\ 4\ 2 \right], \left[1\ 3\ 2\ 4 \right],$$
$$\left.\left[4\ 1\ 2\ 3 \right], \left[1\ 4\ 2\ 3 \right], \left[1\ 2\ 4\ 3 \right], \left[1\ 2\ 3\ 4 \right]\right] \quad \textbf{(5)}$$

```
> triable := proc(n)
  local L, i, c;
  L := perms(n);
  c := 0;
  for i from 1 to nops(L) do
        if est_trie(triapile(L[i])) then
            c := c + 1 fi;
  od; c
  end:
```

```
> seq(triable(n), n = 1 ..8)
```

$$1, 2, 5, 14, 42, 132, 429, 1430 \quad \textbf{(6)}$$

```
> # Ce sont les nombres de Catalan
>
```