« Tu l'as trop écrasé, César, ce Port-Salut. »

est un alexandrin attribué à Victot Hugo qui se lit dans les deux sens (si on ne tient pas compte des espaces et de la ponctuation).
Une telle phrase est un *palindrome*. On dira qu'un tableau `B` de nombres indexés de p à q est un palindrome si et seulement si

$$\texttt{[B[p],B[p+1],}\cdots\texttt{,B[q]]} = \texttt{[B[q],B[q-1],}\cdots\texttt{,B[p]]}$$

Le nom `test_pal` désigne une procédure *récursive* prenant trois arguments : le premier `A` désigne un tableau de nombres entiers, les deux derniers `p` et `q` désignent des clés pour le tableau `A` avec `p` inférieur ou égal à `q` .
L'appel `test_pal(A,p,q)` renvoie vrai si le tableau

$$\texttt{[A[p],A[p+1]}\cdots\texttt{A[q]]}$$

extrait de `A` est un palindrome et renvoie faux sinon.

1. Écrire en langage Maple une procédure *récursive* `test_pal`.
2. Assigner à `B` un tableau de nombres indexé de 0 à 5 et préciser la syntaxe de l'appel de la procédure permettant de savoir si `B` est un palindrome.

Corrigé

1. On peut implémenter la procédure `test_pal` par le code suivant

```
test_pal := proc(A,p,q)
  if p >= q then return true fi;
  if A[p] <> A[q] then return false fi;
  return test_pal(A,p+1,q-1);
end proc:
```

   Remarquer le test `p>=q` qui, lorsqu'il est réalisé, fait renvoyer la valeur vraie à la procédure. Cela peut sembler étrange mais c'est conforme à ce que l'on exige de la procédure. Elle n'a pas à être appelée dans le cas où `p>q`, on est donc libre de lui faire renvoyer ce qui nous arrange.

2. On assigne un tableau et on teste s'il est un palindrome avec le code suivant

```
B:= array(0..5,[1,2,3,3,2,1]);
test_pal(B,0,5);
```

   L'appel à `test_pal` renvoie la valeur vraie dans ce cas particulier.