Énoncé

Au démarrage d'un certain algorithme, on considère les variables suivantes :
- `n`, `m`, `M` désignent des nombres entiers avec `n>1` et `M-m > n`.
- `A` désigne un tableau indexé de 1 à `n` dont les valeurs sont des entiers entre `m` et `M`.
- `B` désigne un tableau vide indexé de 1 à `n`.
- `C` désigne un tableau indexé de `m` à `M` et dont toutes les valeurs sont nulles.

On présente l'algorithme en langage usuel.
- Pour `i` de 1 à `n`
  - `C[A[i]] <-- C[A[i]] +1`
- # Pause 1
- Pour `v` de `m+1` à `M`
  - `C[v] <-- C[v] + C[v-1]`
- # Pause 2
- `i <-- n`
- tant que `i >= 1`
  - `B[C[A[i]]] <-- A[i]`
  - `C[A[i]] <-- C[A[i]]-1`
  - `i <-- i-1`
- # Fin

Les lignes commençant par # sont des commentaires.

1. Implémenter en langage Maple l'algorithme présenté.
2. Dans cette question, `v` désigne un entier entre `m` et `M`.
   a. Que représente `C[v]` lors de la pause 1.
   b. Que représente `C[v]` lors de la pause 2.
3. Que représente `B` à la fin. Justifier.

Corrigé

1. On peut implémenter cet algorithme dans une procédure codée de la manière suivante.

```
tri_den := proc(A,n,m,M)
 local i,v,C,B;
 C := array(m..M);
 for v from m to M do C[v] := 0 od;
 B := array(1..n);
 for i from 1 to n do
   C[A[i]] := C[A[i]] +1;
 od;
 # Pause 1
 for v from m+1 to M do
   C[v] := C[v] + C[v-1];
 od;
 #Pause 2
 i:=n;
 while i >= 1 do
   B[C[A[i]]] := A[i];
   C[A[i]] := C[A[i]]-1;
   i := i-1;
 od;
 return(eval(B));
end proc:
```

2. a. Après la première boucle, lors de la pause 1, chaque `C[v]` désigne le nombre de fois où la valeur `v` figure dans `A`. Si les valeurs étaient supposées distinctes ce nombre serait 0 ou 1. Dans notre cas il peut être plus grand.

   b. Après la deuxième boucle, lors de la pause 2, chaque `C[v]` désigne le nombre de clés pour lesquelles les valeurs de `A` sont inférieures ou égales à `v`.

3. Soit `i` entre 1 et `n` alors `v=A[i]` est une valeur du tableau `A` donc `C[v]` désigne le nombre d'occurence compris entre 1 et `n` des valeurs du tableau inférieures à `v`. Si les valeurs étaient deux à deux distinctes ce serait la place (clé) de cette valeur dans le tableau obtenu à partir de `A` en triant par ordre croissant. On ne suppose pas ici les valeurs distinctes. Il faut donc gérer les « ex-aequo ». C'est ce que fait la dernière boucle en partant de la plus grande valeur possible et en décrémentant le nombre d'occurences.
   À la fin de l'algorithme `B` désigne donc ce tableau ordonné.

Rémy Nicolai Atridnomb