**Exercice I :**

1 - Trouver le résultat fourni par l’algorithme :  
Procédure SomCar ( →X1 : numérique, → X2 : numérique, ↔S : numérique)  
Début  
X1 ← X1 \* X1  
X2 ← X2 \* X2  
S ← X1 + X2  
Fin  
Programme principal :  
Variables X, Y, Z : numériques  
X ← 3  
Y ← 4  
Z ← 0  
SomCar(X, Y, Z)  
Ecrire X, « ^2 + », Y, « ^2 = », Z  
Fin du programme principal

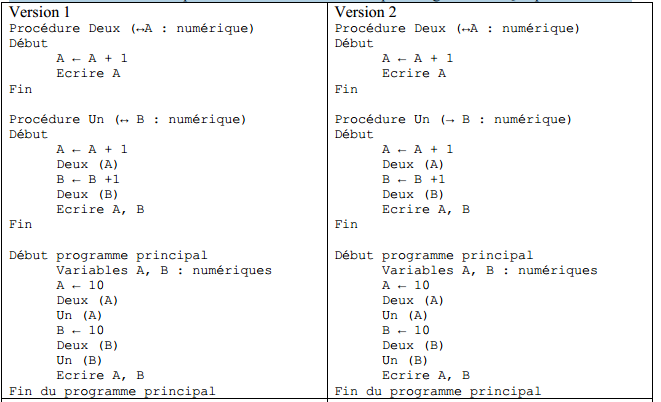
2 - Remplacer dans ce programme la procédure par une fonction

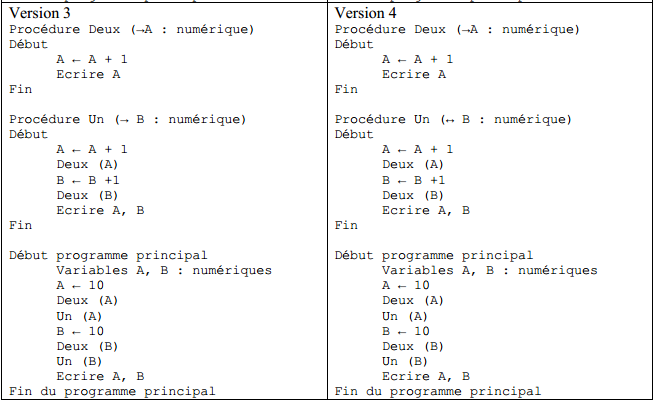
**Exercice II**

Quels sont les résultats produits par l’algorithme suivant :  
Procédure Max (→X : numérique, →Y : numérique, ↔ M : numérique)  
Début  
A ← X  
Si A < Y alors  
A ← Y  
Fsi  
M ← A  
Fin // de la procédure  
Début du programme principal  
Variables A, B, C : numériques  
A ← 3  
B ← 7  
C ← 0  
Max (A, B, C)  
Ecrire « Le maximum de », A, « et », B, « est », C  
Fin du programme principal

**Exercice III**

Les algorithmes suivants ont été écrits par un mauvais programmeur, particulièrement maladroit dans les choix des noms des paramètres et peu soucieux d’éviter les effets de bords. Seuls les modes de transmission des paramètres diffèrent entre ces quatre algorithmes. Que produisent ils ?





**Exercice IV**

Ecrire une fonction qui permet de savoir si un entier est divisible par un autre. On pourra utiliser un nouveau type nommé logique afin de renvoyer le résultat

**Exercice V**

Créer un petit ensemble de procédures et de fonctions permettant de manipuler facilement les heures et les minutes et composé de :

1. La fonction *Minutes*, qui calcule le nombre des minutes correspondant à un nombre d’heures et un nombre de minutes donnés.
2. La fonction ou la procédure HeuresMinutes qui réalise la transformation inverse de la fonction Minute.
3. La procédure AjouteTemps qui additionne deux couples de données heures et minutes en utilisant les deux fonctions précédentes.

**Correction des exercices**

**Exercice I :**

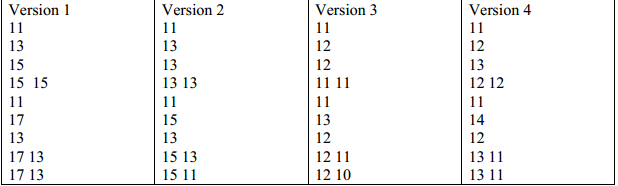
1. Ce programme affiche pour résultat « 3^2+4^2 = 25 ». On remarquera que la procédure, bien qu’elle ait modifié les valeurs des paramètres formels X1 et X2, n’a pas modifié la valeur des paramètres effectifs X et Y car ils étaient en entrée (passés par valeur).
2. Avec une fonction, ce programme devient :  
   Fonction SomCar ( →X1 : numérique, → X2 : numérique) : numérique  
   Début  
   X1 ← X1 \* X1  
   X2 ← X2 \* X2  
   Résultat X1 + X2  
   Fin  
   Programme principal :  
   Variables X, Y, Z : numériques  
   X ← 3  
   Y ← 4  
   Z ← SomCar(X, Y)  
   Ecrire X, « ^2 + », Y, « ^2 = », Z  
   Fin du programme principal  
   Remarque :  
   On peut aussi écrire :  
   Ecrire X, « ^2 + », Y, « ^2 = », SomCar(X, Y)

**Exercice II**

On aurait souhaité que cet algorithme fournisse pour résultat « Le maximum de 3 et 7 est 7 », mais il donne « Le maximum de 7 et 7 est 7 ». A n’est pas le nom d’un paramètre ni d’une variable locale pour la procédure qui utilise donc la variable A du programme principal comme variable globale et la modifie en lui affectant d’abord 3, puis 7. Si par malchance les valeurs initiales de A et B avaient été respectivement de 7 et 3, on ne se serait aperçu de rien et on n’aurait découvert ce problème que le jour où les valeurs de A et B auraient été dans l’ordre croissant. La modification subreptice d’une variable globale par une procédure ou une fonction (effet de bord) est une cause de mauvais fonctionnement de certains algorithmes souvent difficile à repérer et même à déceler.

**Exercice III :**

Les quatre algorithmes produisent des résultats différents en raison de la modification des paramètres effectifs lors des passages par adresse, et de leur conservation lors des passages par valeur.



**Exercice IV**

fonction logique Divise (var entier a, var entier b)  
début  
si (a mod b = 0)  
retourner vrai;  
sinon  
retourner faux;  
fin  
qui peut s'utiliser : si (Divise (x, y) = vrai)

**Exercice V**

1. Fonction Minutes ( →H : numérique, → M : numérique) : numérique  
   Début  
   Résultat H \* 60 + M  
   Fin
2. Pour HeuresMinutes, il y a deux résultats à fournir, une fonction ne peut convenir, il faut donc écrire une procédure comportant trois paramètres : La durée (entrée), l’heure (sortie) et les minutes (sortie).  
   Procédure HeuresMinutes (→ Durée : numérique, ↔ H : numérique, ↔  
   M : numérique)  
   Début  
   H ← Durée Div 60 // division entière  
   M ← Durée – 60 \* H  
   Fin
3. La procédure AjouteTemps reçoit quatre paramètres en entrée, fournit deux paramètres en sortie. La variable locale MinuteEnTout sert à stocker un résultat intermédiaire, mais elle n’est pas indispensable.  
   Procédure AjouteTemps (→H1 : numérique, → M1 : numérique, → H2 :  
   numérique, → M2 : numérique, ↔Hsomme : numérique, ↔Msomme :  
   numérique)  
   Variables MinuteEnTout : numérique  
   Début  
   MinuteEnTout ← Minutes (H1, M1) + Minutes (H2, M2)  
   HeuresMinutes(MinuteEnTout, Hsomme , Msomme)  
   Fin