Énoncé 2
Fonctions, passage d'arguments, renvoi de résultats

élaboré par : Łukasz Starzak
Department of Microelectronics and Computer Science, Lodz University of Technology

Remarques

A. Les sous-points marqués avec des lettres (a, b, ...) désignent des modifications qui doivent être introduites de façon progressive, c'est-à-dire après avoir vérifié que le sous-point précédent a été réalisé correctement et sans supprimer la fonctionnalité actuelle (à moins que ce soit demandé). Par contre, les points sans numérotation (•) désignent des fonctionnalités ou caractéristiques qui doivent être achevées et présentées simultanément.

B. Lorsque vous modifiez une fonction écrite dans un point ou sous-point antérieur, toujours gardez l'ancienne version.

C. Pensez à optimaliser le nombre d'opérations : d'un côté, il faut le minimiser ; de l'autre, le code doit être lisible ce qui peut justifier la création de variables additionnelles. Rendez les formules mathématiques les plus courtes possible. (Cela est laissé à votre invention.)

D. Appliquez l'indentation conventionnelle du code.

E. S'il n'est pas demandé explicitement d'afficher quelque chose, alors la fonction ne doit pas l'afficher. S'il est demandé d'afficher quelque chose, alors la fonction doit l'afficher une fois et non plus ni rien de plus.

F. Les noms de fonctions doivent être évocatrices et suivre les règles et conventions appropriées. Vous pouvez toujours ajouter le numéro de l'exercice/énoncé pour différentier.

G. Il est interdit de donner à une variable un nom que porte déjà une fonction (créée par vous ou intégrée en Octave) ou une instruction du langage Matlab, et *vice versa*.

H. Il est interdit de donner un même nom à des variables qui se trouvent dans de différents espaces (p. ex. l'espace de base et l'espace d'une fonction). Cela s'applique aussi aux démonstrations dans le compte rendu.

I. Parfois l'ancienne version d'une fonction peut toujours résider dans la mémoire de l'ordinateur. Vous pouvez le constater quand vous modifiez le fichier M sur le disque mais la fonction fonctionne comme avant (pourtant ce n'est pas l'unique cause possible). Utilisez la fonction `clear` pour l'effacer (la fonction en cause seulement et non toutes les fonctions et variables), ensuite invoquez la fonction encore une fois.

J. Vous pouvez placer les définitions des matrices et vecteurs d'entrée dans un fichier M séparé (fichier programme et non fonction ; il doit contenir juste des affectations), ce qui facilitera votre travail si la variable est effacée par accident ou si vous devez interrompre votre travail et le reprendre plus tard.

Interaction avec l'utilisateur à l'intérieur d'une fonction

1. Créez un fichier M pour réaliser le calcul du salaire brut à l'aide d'une fonction. Cette fonction devrait :
   - demander le salaire net à l'utilisateur à l'aide de la fonction `input` ;
   - afficher le résultat, précédé du texte « Le salaire brut vaut: », à l'aide de la fonction `disp`.

   Pour la formule de calcul, reportez-vous à votre solution de l'exercice 1 de l'énoncé 1.

Arguments et résultats de fonctions

2. Réalisez le même calcul (que dans l'exercice 1) à l'aide d'une autre fonction qui devrait :
   - obtenir le salaire net comme son argument (au lieu de le demander à l'utilisateur avec `input`) ;
   - renvoyer le résultat de façon qu'une variable externe (dans l'espace de base) puisse être affectée avec lui (au lieu de juste l'afficher avec `disp`).

   Ne pas modifier la formule de calcul conçue dans l'exercice 1.

   Assurez-vous que vous êtes capable d'effectuer d'opérations arithmétiques sur la variable affectée avec le résultat, p. ex. la multiplier par un nombre de mois. Est-ce que le même est possible avec la fonction

développée dans l'exercice 1 ?

3. Modifiez la fonction de l'exercice 2 :
   a) rendez-la plus générique en introduisant un deuxième argument : le taux de cotisations comme pourcentage (c'est-à-dire, on voudrait être capable d'introduire simplement 13,7 au lieu de 0,137) ;
   b) en plus, définissez le taux de cotisations standard de 13,7% comme valeur par défaut ;
   c) assurez que la fonction fonctionne aussi quand son premier argument est un vecteur (au lieu d'un nombre simple) ainsi que quand ses deux arguments sont des vecteurs d'une même taille (s'il y a des erreurs, revoir la solution de l'exercice 3 de l'énoncé 1).
4. Créez une fonction qui :
   - prendrait un vecteur de salaires de base et un vecteur de taux de primes (exprimées comme pourcentages) comme arguments (deux arguments alors) ;
   - renverrait un vecteur de salaires finaux (salaire net plus prime) comme résultat.

   Appuyez-vous sur une partie appropriée de votre solution de l'exercice 3 de l'énoncé 1. (La solution sera très pareille à celle de l'exercice 3 ici ; c'est normal.)

Invocation d'une fonction depuis une autre fonction

5. Une entreprise doit calculer les salaires d'un employé comme dans l'exercice 3 de l'énoncé 1 (voir le schéma y donné). Créez une fonction qui :
   - prendrait une matrice à 3 colonnes comme un premier argument : semaine, salaire de base (net), prime (tout comme dans l'exercice 3 de l'énonce 1) ;
   - prendrait un taux de cotisations salariales comme un deuxième argument (un nombre simple commun : assumer que ce taux ne changeait pas) ;
   - renverrait comme résultat une matrice à 4 colonnes, dans cet ordre : semaine, salaire de base (net), salaire final net, salaire final brut.

   Pour effectuer tous les calculs nécessaires, cette fonction ne doit rien calculer par soi-même mais appeler les fonctions créées dans les exercices 3 et 4. C'est-à-dire, il faut juste invoquer leurs noms et y passer les arguments appropriés, sans aucune modification de leurs codes et sans recopier leur codes dans la nouvelle fonction.