Correction TD1

Data Structure & Algorithm I

Matthieu Jimenez

Été 2015

Correction TD1

Data Structure & Algorithm I

Exercice I

Enoncé:

Proposer un algorithme qui prend en entrée une série de nombre et qui renvoie l'index du premier plus petit élément de cette série.

Correction:

```
int firstMinimumInList(List list){
int indexOfTheMinimumElement = 1;
for (int i=2;i<=list.size();i++){
     if(list[i]<list[indexOfTheMinimumElement])
          indexOfTheMinimumElement= i;
}
return indexOfTheMinimumElement;
}
```

Exercice II

Enoncé:

Soit les 2 façons de gérer les notes suivantes:

Méthode I	**Méthode II**
moyEleveI= 12 moyEleveII= 15 moyEleveIII= 14 moyEleveIV= 11 moyEleveV= 10	listMoyenneEleve= [12,15,14,11,10]

a)Proposer deux pseudo-codes permettant de calculer la moyenne de la classe, lune prendra en entrée la première façon de gérer les données, l'autre la seconde.

Oups, on a oublié de rentrer la moyenne de l'élève 6 …

Qu'est ce que cela change à vos pseudo codes? Conclusions?

Correction:

a) Méthode I:

$$\frac{\text{moyEleveI} + \text{moyEleveII} + \text{moyEleveIII} + \text{moyEleveIV} + \text{moyEleveV}}{5}$$

Methode II:

```
int sum = 0;
for (int i=1;i<listMoyenneEleve.size();i++)
 sum+=listMoyenneEleve[i];
return sum/listMoyenneEleveList;
}
```

b) Dans le cas de la première méthode, il va falloir rajouter moyEleveVI en numérateur et remplacer 5 par 6 pour le dénominateur

Dans le cas de la seconde méthode, aucun changement n'est nécessaire, l'algorithme n'est pas fixe et s'adapte.

En conclusion, une structure de donnée est beaucoup plus pratique à utiliser et si la modification paraît bénigne pour la première méthode, demandez vous comment cela se passerai si on ajoute 10 élèves et qu'on en retire 3 …

Exercice III

Enoncé:

Les mots de passe de compte utilisateurs sont des données très critiques que l'on ne souhaite absolument pas voir compromises. Si l'effort de renforcer la sécurité de ces mots de passe est à la charge des entreprises, il convient également de conseiller les utilisateurs afin qu'ils ne choisissent pas des mots de passe trop simple. Une méthode possible est d'obliger l'utilisateur à choisir un mot de passe comprenant au minimum une majuscule, une minuscule et un chiffre.

L'entreprise pour laquelle vous travaillez, a décidé d'appliquer cette méthode. Elle vous a donc chargé de concevoir l'algorithme permettant cette vérification.

Proposez un pseudo-code reprenant le style de celui vu en cours:
input: Chaine de caractères

output: boolean

Correction:

```
boolean verificationMdP(String A){
    // on considère la chaine de caractère A comme un tableau de
taille n débutant à 1
    boolean majuscule, minuscule, nombre=false
    for (int i=1; i<=n; i++){
        if (isMajuscule(A[i])
            majuscule=true;
        else    if (isMinuscule(A[i])
                minuscule=true
            else    if (isNumber(A[i])
                    number=true;
        if (majuscule & minuscule & number)
            return true;
    }
    return false;
}
```

Exercice IV

Enoncé:

a) Reprenez votre algorithme de l'exercice 1 et calculer sa fonction de complexité best case et worst case.

b) Reprenez votre algorithme de l'exercice 3 et calculer sa fonction de complexité best case et worst case.

Correction:

a) complexité Best Case = 2 cn & Worst Case = 3cn - c

a) complexité Best Case = 19 c & Worst Case = 4c + 6cn