## Le test et le debugging des algorithmes

1. Définir ce qu’est une donnée de test
2. Lister les deux fonctions de tests les plus utilisées, ainsi que les deux mesures les plus utilisées pour les tests
3. Définir la notion de classe d’équivalence de test
4. C’est quoi le problème de l’Oracle ?
5. C’est quoi le problème de la partition ?

## Exercices sur les tests

## Exercices sur les nombres

**Ex 1 :** Produire des données de tests pour une fonction calculant la racine carrée d’un réel donné

**Ex2 :** Produire des données de tests pour une fonction calculant le logarithme d’un réel donné

**Ex3 : P**roduire des données de tests pour une fonction calculant factorielle n où n est un entier

Ex4 : **P**roduire des données de tests pour une fonction calculant Ckn (combinaison de k dans n), où k et n sont des entiers donnés.

Ex5 : produire des données de test pour le calcul de xn, où n est un entier positif et x un réel.

## Exercices de base sur les dates

Ex1 : produire des données de tests pour une fonction fournissant le nombre de jours calendaires entre deux dates, défini par nc=d2-d1, où d2 et d1 sont des dates considérées sur une échelle de jours.

Ex2 : on considère une fonction qui pour une date donnée renvoie le jour de la semaine correspondant à cette date. Fournir les données de tests pour ladite fonction.

## Exercices sur les chaines de caractères

1. Fournir les données de test pour une fonction permettant de compresser un fichier donné, et créer un autre fichier correspondant au fichier compressé.
2. On suppose qu’on dispose d’un fichier d’enregistrements contenant les noms et les âges des étudiants. On souhaite une fonction de recherche permettant de retrouver tous les enregistrements correspondant à un nom donné. Fournir les données de test pour cette fonction.

## Exercices élémentaires sur les bases de données

## Exercices pluridimensionnels

**Ex1 :** Un programme prend en entrée trois entiers. Ces trois entiers sont interprétés comme représentant les longueurs des cotés d’un triangle. Le programme rend un résultat précisant s’il s’agit d’un triangle scalène, isocèle ou équilatéral.

1. Déterminer les dimensions des variables d’entrée du problème
2. Déterminer les contraintes associées au problème (sémantiques ou autres)
3. Fournir les classes d’équivalence pour chaque dimension
4. Déterminer la dimension de sortie et les classes d’équivalence correspondantes
5. Pour chaque classe d’équivalence de sortie, caractériser les limites correspondantes dans les données d’entrées.
6. En déduire pour chaque classe d’équivalence de sortie les données d’entrée de test aux limites
7. En déduire les données de tests complètes pour le problème posé, au sens de la couverture de toutes les classes d’équivalence identifiées.

**Exercice 2**

Supposons que nous élaborions un compilateur pour le langage BASIC. Un extrait des spécifications précise :

«L’instruction FOR n’accepte qu**’un seul paramètre** en tant que variable auxiliaire. Son nom ne doit pas dépasser **deux caractères non blancs**; Après le signe = est précisée aussi une borne supérieure et une borne inférieure. Les bornes sont des **entiers positifs** et on place entre eux le mot-clé TO. »

Proposer les données de tests.

Nbvaraux=1

0<Taille(nom(varaux))<=2 non blancs

0<Nbblancs entre nom et = >=1

0=Nbnonblancs entre nom et =

Il est préférable de traduire le problème à l’aide d’un automate d’états finis, et d’identifier pour chaque état les classes d’équivalence à étudier.

Expression sur la forme d’an automate d’états finis.