Examen du 16 Mars 2013 (2h)
TOP : Techniques and tOols for Programming
Telecom Nancy - Première année

La notation tiendra compte de la validité des réponses, mais aussi de la présentation et de la clarté de la rédaction.

Documents interdits, à l'exception d'une feuille A4 à rendre avec votre copie.

Questions de cours (2pts)

- Question 1 ($\frac{1}{2}$ pt). Démontrez que
$$n^2 \in O(10^{-5}n^3)$$
$$2^{n+100} \in O(2^n)$$
- Question 2. Quel est le rapport entre les notations O, Ω, Θ et les temps de calcul dans le meilleur des cas, le pire des cas et le cas moyen ? ($\frac{1}{2}$ pt)
- Question 3. À quelles classes de complexité (en notation Θ) appartiennent les algorithmes 1 et 2 suivants ? ($\frac{1}{2}$ pt)

algorithme 1
```
pour i = 1 à n faire
  pour j = 1 à n faire
    x += 3
```

algorithme 2
```
pour i = 1 à n faire
  pour j = 1 à n faire
    x += 3
pour i = 1 à n faire
  y = x + 5
```

- Question 4. Définissez en quelques mots le principe des tris (1) par insertion (2) par sélection en explicitant en français (sans équation) leurs invariants. ($\frac{1}{2}$)

La traversée du pont (4pts)

Un groupe de 4 personnes possédant une seule lampe torche souhaite traverser un vieux pont mal éclairé la nuit. Au maximum deux personnes peuvent traverser le pont à la fois. Chaque sous-groupe (une ou deux personnes) traversant le pont doit avoir avec lui la lampe torche. La lampe torche doit être utilisée dans le sens de la traversée vers le bon côté du pont et dans le sens de retour pour la ramener vers le mauvais côté du pont. La personne A a besoin d'une minute pour traverser le pont, la personne B a besoin de 2 minutes, la personne C a besoin de 5 minutes et la personne D a besoin de 10 minutes. Chaque paire de personnes traversant le pont doit suivre l'allure du plus lent des deux.

On cherche à trouver la séquence optimale de déplacements des 4 personnes pour qu'ils traversent le pont en un temps le plus court.

- Question 1 ($1pt$).Combien de temps faut-il au minium pour que les quatre personnes traversent le pont ? Décrivez la séquence de déplacements associée à ce temps minimum.
- Question 2 ($1pt$). Explicitez en français l'algorithme à écrire. Le fonctionnement en général en vous appuyant sur la séquence de déplacements dessinée à la question précédente, puis l'idée pour chaque étage de la récursion.
- Question 3 ($\frac{1}{2}pt$). Quel sera le prototype de votre fonction de résolution ? Quels sont les cas triviaux ?
- Question 4 ($1,5pt$). Écrivez le code de cette fonction de résolution.

Code récursif mystère (5pts) (D'après Baynat, Exercices et problèmes d'algorithmique.)

Considérez le code mystère suivant.

```
public int puzzle(int n) {
 if (i == 0)
   return 2;
 else
   return puzzle(i-1)*puzzle(i-1);
}
```

- Question 1 (1pt). Explicitez les appels récursifs effectués pour `puzzle(3)` en prenant garde à ne développer qu'un seul appel récursif par ligne.
  Calculez le résultat de `puzzle(4)` et `puzzle(5)`.
  Que semble calculer cette fonction ?
- Question 2 ($\frac{1}{2}pt$). Montrez la terminaison de cet algorithme.
- Question 3 (1pt). Démontrez par récurrence ce que calcule cette fonction, en fonction de n.
- Question 4 (1pt). Le nombre de multiplications effectuées par la fonction puzzle est la solution de l'équation de récurrence suivante.

$$m(n) = \begin{cases} 0 & \text{si n=0} \\ 1 + 2 \times m(n-1) & \text{si } n > 0 \end{cases}$$

  Montrez par récurrence que $m(n) = 2^n - 1$, et déduisez en la complexité algorithmique de `puzzle`.
- Question 5 (1pt). Modifiez l'algorithme pour ramener la complexité dans $\Theta(n)$.
  Il n'est pas demandé de démontrer formellement que le nouvel algorithme est effectivement dans cette classe de complexité, ni sa correction.
- Question 6 ($\frac{1}{2}pt$). Est-il possible de dérécursiver directement cette nouvelle fonction ? Pourquoi ?

Les tests (5pts)

Les fonctions ci-dessous appartiennent à une classe permettant de stocker des chaînes de caractères sous forme d'une pile.

- Question 1. Pour chaque méthode, indiquez les tests qu'il faudrait écrire pour vérifier leur bon fonctionnement. Écrivez simplement les tests à réaliser en français (vous n'avez pas à les écrire explicitement en utilisant JUnit ou une autre technique) (3pts).
- Question 2. Écrivez en JUnit les tests de la méthode `isEmpty()` (1pt).
- Question 3. Avez vous utilisé une approche whitebox, greybox, bluebox ou blackbox testing ? Justifiez votre réponse ? (1pt)

```
public class MyStack {
  /** Tests if this stack is empty.
   * @return True if and only if this stack contains no items; false otherwise
   */
  public boolean isEmpty()

  /**
   * Pushes an item onto the top of this stack.
   * @param item { the item to be pushed onto this stack
   * @return The item that was pushed onto the stack
   */
  public String push(String item)

  /**
   * Removes the object at the top of this stack and returns that object as the value of this
   * function.
   * @return The object at the top of this stack
   * @throws EmptyStackException if this stack is empty.
   */
  public String pop()
}
```

Preuve de programme (4 pts) (D'après Baynat, Exercices et problèmes d'algorithmique)

Soit l'algorithme itératif ci-contre. Nous allons calculer formellement la précondition nécessaire pour que la post-condition de cette algorithme soit $Q \equiv res = 2^{2^n}$ en utilisant les formules de calcul de la plus faible précondition rappelées en annexe.

```
public int puzzleIter(int n) {
  int res=2;
  int i=0;
  while (i<n) {
    res = res * res;
    i=i+1;
  }
  return res;
}
```

- Question 1(1pt). Explicitez l'invariant et le variant de la boucle. N'oubliez pas que cet invariant et variant vont servir ensuite pour les obligations de preuve.
- Question 2 (3pts). Calculez la plus faible précondition nécessaire pour que cet algorithme calcule 2^{2^n}. Explicitez ce que vous faites et pourquoi.

Annexe : Règles de calcul des préconditions.

- $\mathbf{WP}(nop, Q) \equiv Q$
- $\mathbf{WP}(x := E, Q) \equiv Q[x := E]$
- $\mathbf{WP}(C; D, Q) \equiv \mathbf{WP}(C, \mathbf{WP}(\mathrm{D,Q}))$
- $\mathbf{WP}(\texttt{if } Cond \texttt{ then } C \texttt{ else } D, Q) \equiv (Cond = \texttt{true} \Rightarrow \mathbf{WP}(C, Q)) \wedge (Cond = \texttt{false} \Rightarrow \mathbf{WP}(D, Q))$
- $\mathbf{WP}(\texttt{while } E \texttt{ do } C \texttt{ done } \{\text{inv I var V}\}, \mathrm{Q}) \equiv I$; Obligations de preuve :
  - $(E = \texttt{true} \wedge I \wedge V = z) \Rightarrow \mathbf{WP}(C, I \wedge V < z))$
  - $I \Rightarrow V \geq 0$
  - $(E = \texttt{false} \wedge I) \Rightarrow Q$