Correction : Examen

Exercice 1 (On suppose $a \geq b$).

```
1) static int PcgdR (int a, int b)
                    {if (b == 0) return a; else return PcgdR(b, a % b);}

2) static int PcgdI (int a, int b){
     double r;
     while(0 != (r = a % b)) {a = b; b = r;}
     return b;
}
```

Exercice 2 1) L'algorithme de Strassen calcule d'abord récursivement 7 sous-produits de matrices $n/2 \times n/2$ puis le produit AB avec un nombre constant d'additions en temps $\Theta(n^2)$. Puisque $2 < \log_2 7$, $\exists \varepsilon > 0$ tel que $n^2 = O(n^{\log_2 7 - \varepsilon})$, donc le terme $\Theta(n^{\log_2 7})$ s'impose.

2) $AD + BC = (A + B)(C + D) - AC - BD$ donc à partir des 3 produits matriciels $(A + B)(C + D)$, AC, et BD on peut recomposer avec un nombre constant d'additions le terme $(AC - BD, AD + BC)$. Toutefois cela n'a ici aucun impact sur la complexité puisque l'on effectuerait 3 appels à l'algorithme de Strassen au lieu de 4 et $\Theta(3 \times n^{\log_2 7}) = \Theta(4 \times n^{\log_2 7})$.

Exercice 3

1. $\texttt{algo}(8) = 64$.
2. $\texttt{algo}(2^k) = 4^k = 2^{2k}$.
3. $T(n) = 4T(n/2) + f(n)$
   (a) $T(n) = \Theta(n^2)$.
   (b) $T(n) = \Theta(n^2 \log n)$.
   (c) $T(n) = \Theta(n^4)$.
4. L'implémentation suivante a un temps $T(n) = T(n/2) + O(1)$, soit $\Theta(\log n)$.

   $\texttt{algo}(n)$
   Si $n = 1$ retourner 1
   Retourner $4 \times \texttt{algo}(n/2)$
5. Vrai pour $k = 0$; $\texttt{algo}(2^{k+1}) = 4 \times \texttt{algo}(2^k) = 4 \times 4^k = 4^{k+1}$.

Exercice 4

1. Supposons que ce soit faux et choisissons un contre-exemple avec n minimum. On a alors $n \geq 2$ et $\lfloor \frac{n}{2} \rfloor \notin \{1, \ldots, n-1\}$; c'est impossible.
2. Pour tout x, il existe un unique entier k tel que $x \in \{2^k, \ldots, 2^{k+1} - 1\}$. Il suffit de montrer que $p^k(x) = 1$. Par induction sur k: C'est vrai pour $k = 1$ et, pour tout $x \in \{2^{k+1}, \ldots, 2^{k+2} - 1\}$, on a $p(x) = \lfloor x/2 \rfloor = 2^k$, donc $p^{k+1}(x) = 1$.

3. $p(x) = x - 1$.

Exercice 5

1. `salade2fruit(12568,1000)`$= 12568000$.
2. `salade2fruit(x,y)`$= xy$.
3. Soient a et b les valeurs de `poire` et `mangue` données en paramètre d'entrée.
   On a $a = \sum_{i=0}^{i=n} a_i 2^i$ avec $a_i \in \{0,1\}$ pour tout $i = 0, \ldots, n$. Après un nombre k de passages dans la boucle while, on a les trois invariants suivants:
   (a) `poire` $= \sum_{i=k}^{i=n} a_i 2^{i-k}$
   (b) `mangue` $= b2^k$
   (c) `pomme` $= \sum_{i=0}^{i=k-1} a_i b 2^i$

   En effet, pour $k = 0$ c'est clairement vrai puisque `poire` $= a$, `mangue` $= b$, et `pomme` $= 0$. Puis l'invariant 1 se conserve car faire `poire/=2` revient à faire $2^{i-k} \leftarrow 2^{i-k-1}$ et $a_k \leftarrow 0$ dans l'égalité (a). L'invariant 2 se conserve clairement par `mangue*=2`. L'invariant 3 se conserve car `poire%2`$= a_k$.
   Donc l'algorithme retourne $\sum_{i=0}^{i=n} a_i b 2^i = ab$.