Algorithmique
Correction Partiel n° 1

Info-Sup – Epita

08 jan 2013

Solution 1 **(Un peu de vocabulaire... –** ***5 points*****)**

1. La racine de `B` est : `T`
2. Les noeuds internes `(en ordre hiérarchique)` de l'arbre `B` sont : `TWSCETU`
3. La taille de l'arbre `B` est : `13`
4. La longueur de cheminement externe de l'arbre `B` est : `18`
5. Le parcours *infixe* de l'arbre `B` est : `ICNWOEDTTESRU`

Solution 2 **(Pile ou... Pile ? –** ***3 points*****)**

1. Nous utiliserons une pile lorsque des données doivent être mémorisées et traitées en ordre inverse de leur arrivée.

2. Les sorties sont (dans l'ordre) : `3 5 2 1 6 7 4 9 8`

3. Le principe d'un tel algorithme est le suivant : Un palindrome est un mot qui peut se lire de façon identique dans un sens comme dans l'autre (*ex :radar*). Le principe de fonctionnement d'une pile est de restituer les données dans l'ordre inverse de leur arrivée. Dès lors il suffit de lire le mot caractère par caractère et d'empiler ceux-ci au fur et à mesure. En les dépilant ensuite un par un et en les concaténant dans leur ordre de sortie de la pile (inverse de l'ordre d'arrivée dans celle-ci), nous créons un autre mot, inverse de l'original. Il ne nous reste plus qu'à comparer ces deux mots. S'ils sont égaux, nous avons affaire ou non à un palindrome.

   Une optimisation possible, est de ne faire cela que sur la moitié des caractères qui constituent le mot, ce dernier pouvant se lire identiquement dans les deux sens, la comparaison des deux moitiés (dont une inversée) suffira.

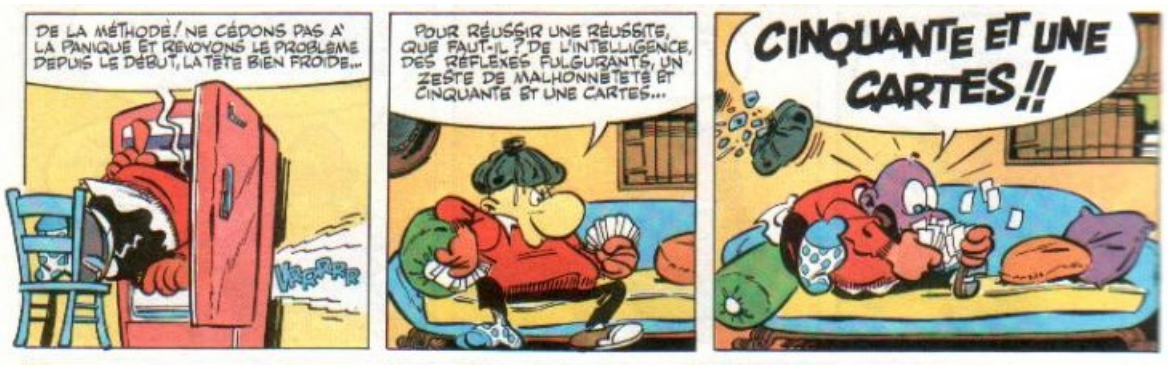
Solution 3 **(Recherche – *3 points*)**

Spécifications :

La fonction `cherche (entier` x`, t_vect11entiers` V`, entier` d`)` retourne la position de la première occurrence de x trouvée à partir de la position d ($1 \leq d \leq 11$)dans le vecteur V, la valeur 0 si x n'est pas présente.

```
algorithme fonction cherche : entier
   parametres locaux
      entier           x       /* valeur cherchée */
      t_vect11entiers  V
      entier           d       /* position de départ */

   variables
      entier   i
debut
   i ← d
   tant que (i <= 11) et (V[i] <> x) faire
      i ← i+1
   fin tant que
   retourne (i mod 12)
   /* ou
      si i < 12 alors
         retourne i
      sinon
         retourne 0
      fin si
   */
fin algorithme fonction cherche
```



Solution 4 **(Cherche le 0 –** ***3 points*****)**

Spécifications :

La fonction `position0 (t_vect11entiers V)` retourne la position de la première valeur 0 non précédée de 9 dans le vecteur V. Elle retourne 0 si deux valeurs 0 se suivent et sont précédées de 9.

```
algorithme fonction position0 : entier
   parametres locaux
      t_vect11entiers   jeu

   variables
      entier   pos
debut
   pos ← cherche (0, jeu, 1)          /* on cherche le premier 0 */
   si (pos = 1) ou (jeu[pos-1] <> 9) alors
      retourne pos
   sinon                              /* le 0 trouvé est précédé de 9 */
      pos ← cherche (0, jeu, pos+1)      /* on cherche le 2ème 0 */
      si jeu[pos-1] = 0 alors   /* Les deux 0 se suivent : c'est terminé */
         retourne 0
      sinon
         retourne pos
      fin si
   fin si
fin algorithme fonction position0
```

Solution 5 **(1, 2, 3... –** ***3 points*****)**

Spécifications :

La fonction `verifie (t_vect11entiers V)` retourne un booléen indiquant si les 9 entiers de 1 à 9 sont dans les premières cases dans l'ordre croissant suivies de deux valeurs 0.

```
algorithme fonction verifie : booleen
   parametres locaux
      t_vect11entiers   V

   variables
      entier   i
debut
   si (V[10] <> 0) ou (V[11] <> 0) alors
      retourne faux
   sinon
      i ← 1
      tant que (i <= 9) et (V[i] = i) faire
         i ← i + 1
      fin tant que
      retourne (i = 10)
   fin si
fin algorithme fonction verifie
```

Solution 6 **(Reussite –** ***3 points*****)**

Spécifications :

La fonction `reussite (t_vect11entiers V) : booleen` tente de réussir la réussite : elle retourne un booléen indiquant la réussite ou l'échec.

```
algorithme fonction reussite : booleen
   parametres locaux
      t_vect11entiers   jeu

   variables
      entier   pos0
debut
   pos0 ← position0 (jeu)
   tant que pos0 <> 0 faire
      deplace (jeu, pos0)
      pos0 ← position0
   fin tant que
   retourne (verifie (jeu))
fin algorithme fonction reussite
```