Algorithmique
Partiel n° 1

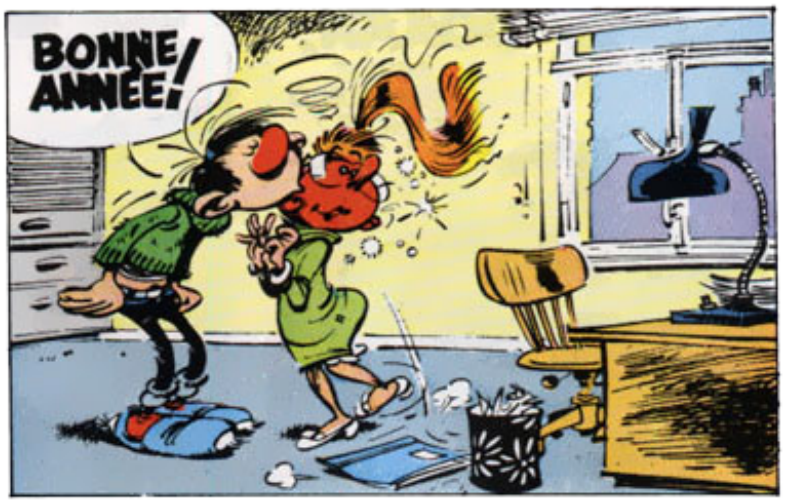
Info-Sup – Epita

D.S. 310020.31 BW (08 jan 2013 - 10 :00)

Remarques (à lire !) :

- Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- **Les algorithmes :**
  - Tout algorithme doit être écrit dans le langage Algo (pas de C#, Caml ou autre).
  - Tout code Algo non indenté ne sera pas corrigé.
  - En dehors d'indication dans les énoncés, vous ne pouvez utiliser aucune routine (fonction ou procédure) supplémentaire.
  - Tout ce dont vous avez besoin (opérations de types abtraits, types) est donné en annexe.
  - **Rappel :** pour chaque algorithme, lorsque demandé sur les feuilles de réponses, vous devez donner :
    - **Les spécifications** c'est à dire ce qu'il fait, les paramètres (types et significations) et les éventuelles conditions d'utilisation.
    - **Le principe algorithmique** c'est à dire en clair la méthode retenue pour résoudre le problème.

    Attention le principe et les spécifications sont notés, ne pas les envisager revient à sacrifier directement des points.
- Durée : 2h.



Des arbres binaires

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

Soit l'arbre `B` représenté figure 1.

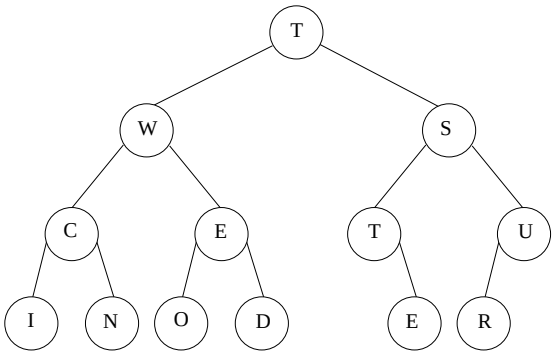


FIGURE 1 – Arbre binaire.

1. Quel noeud est la racine de `B` ?
2. Quels sont `(en ordre hiérarchique)` les noeuds internes de l'arbre `B` ?
3. Quelle est la taille de l'arbre `B` ?
4. Quelle est la longueur de cheminement externe de l'arbre `B` ?
5. Donner le parcours *infixe* de l'arbre `B`.

Des piles

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

1. Dans la résolution d'un problème, quelle raison principale motive l'utilisation d'une pile ?
2. Supposons une pile `P`, quelles seront, dans l'ordre, les valeurs dépilées pour la série d'opérations suivante ?

```
P<-pilevide, P<-empiler(8,p), P<-empiler(3,p), P<-dépiler(P),
P<-empiler(2,p), P<-empiler(5,p), P<-dépiler(P), P<-dépiler(P),
P<-empiler(9,p), P<-empiler(1,p), P<-dépiler(P), P<-empiler(7,p),
P<-empiler(6,p), P<-dépiler(P), P<-dépiler(P), P<-empiler(4,p),
P<-dépiler(P), P<-dépiler(P), P<-dépiler(P).
```

3. Donner le principe d'un algorithme qui utiliserait ***obligatoirement*** une pile pour tester qu'un mot donné sous la forme d'une chaîne de caractères est ou non un palindrome (un mot qui peut se lire dans les deux sens, exemple : radar).

Des vecteurs

Dans les exercices qui suivent, le type utilisé sera le suivant :

```
types
    t_vect11entiers = 11 entier
```

Exercice 3 (Recherche – *3 points*)

Écrire la fonction `cherche` qui recherche une valeur x dans un vecteur d'entiers V à partir de la position d ($1 \leq d \leq 11$). La fonction retourne la position du premier x trouvé, la valeur 0 si aucun x n'a été trouvé.

Par exemple dans le vecteur V suivant :

1	2	3	4	5	6	7	8	9	10	11
2	4	**0**	7	9	1	5	3	**0**	8	6

- l'appel **cherche (0, V, 1)** retournera la position du premier 0 : 3
- l'appel **cherche (0, V, 4)** retournera la position du deuxième 0 : 9

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

Soit un vecteur de 11 entiers contenant exactement 2 valeurs 0. Utiliser la fonction `cherche` de l'exercice précédent pour écrire la fonction `position0` qui retourne

- la valeur 0 si les deux 0 se suivent et sont précédés de 9 (si on trouve dans cet ordre 9, 0, 0),
- sinon la position de la première valeur 0 non précédée de la valeur 9.

Par exemple :

Dans le vecteur de l'exercice précédent, la fonction retournera 3.

Dans le vecteur suivant :

1	2	3	4	5	6	7	8	9	10	11
1	4	5	7	9	**0**	2	3	**0**	8	6

la fonction retournera 9.

Dans le vecteur suivant :

1	2	3	4	5	6	7	8	9	10	11
1	2	5	7	9	**0**	**0**	3	4	8	6

la fonction retournera 0.

Exercice 5 (1, 2, 3... – *3 points*)

Soit un vecteur de 11 entiers contenant les chiffres de 1 à 9 et deux valeurs nulles (0). Écrire la fonction `verifie` qui vérifie si les valeurs de 1 à 9 sont dans l'ordre croissant dans les 9 premières cases du vecteur (et donc les deux valeurs 0 sont dans les deux dernières cases).

Exercice 6 (Réussite – *3 points*)

Soit un tableau de 11 entiers dans lequel on a rangé aléatoirement les chiffres de 1 à 9 et deux valeurs nulles (0 : qui symbolise une case vide).

Le principe de la réussite est le suivant :
- Échanger une valeur nulle (la première trouvée) avec le successeur de la valeur située dans la case qui précède celle de la valeur nulle. Dans l'exemple donné en annexe, la première valeur nulle est juste après la valeur 4, on peut donc y placer la valeur 5.
  - Si la valeur nulle est en première position, alors c'est la valeur 1 qui pourra la remplacer.
  - Si la valeur nulle est précédée de 9, alors on ne peut pas la remplacer !
- Recommencer ce processus jusqu'à ce que l'on soit bloqué (les deux valeurs nulles sont placées après le 9 qui n'a pas de successeur).

On a "gagné" si on a pu classer les valeurs de 1 à 9 dans l'ordre dans les 9 premières cases du tableau.

Utiliser les fonctions `position0` et `verifie` des exercices précédents ainsi que la procédure `deplace` donnée en annexe pour compléter la fonction `reussite` qui tente de réussir la réussite (elle retourne un booléen indiquant la réussite ou l'échec).

Annexes

Exemple de déroulement de la réussite :

2	4	**0**	7	9	1	5	3	0	8	6

On échange le premier '0' avec '5'

2	4	5	7	9	1	**0**	3	0	8	6

On échange le premier '0' avec '2'

0	4	5	7	9	1	2	3	0	8	6

Le premier '0' est position 1, on l'échange avec '1'

1	4	5	7	9	**0**	2	3	**0**	8	6

Le premier '0' ne peut pas être remplacé.
On échange le deuxième '0' avec '4'

1	**0**	5	7	9	0	2	3	4	8	6

On échange le premier '0' et '2'

1	2	5	7	9	**0**	**0**	3	4	8	6

Plus aucun échange possible ! La réussite a échoué.

Spécifications :

La procédure `deplace` prend en paramètre le tableau de jeu, ainsi que la position de la valeur nulle (0) a échanger. Elle échange ce 0 avec la valeur qui suit (dans l'ordre des entiers) celle qui le précède dans le tableau, 1 s'il est en première position.

```
algorithme procedure deplace
   parametres globaux
      t_vect11entiers   jeu
   parametres locaux
      entier            pos

   variables
      entier   val
debut
   si pos0 = 1 alors
      val ← 1
   sinon
      val ← jeu[pos0-1] + 1
   fin si
   jeu[cherche(val, jeu, 1)] ← 0
   jeu[pos0] ← val
fin algorithme procedure deplace
```