

# [CYBER1][2024-2025] CORRECTION Examen (Sujet A)

## Algorithmique 2

NOM : \_\_\_\_\_


PRÉNOM : \_\_\_\_\_

Vous devez respecter les consignes suivantes, sous peine de 0 :

- I) Lisez le sujet en entier avec attention
- II) Répondez sur le sujet
- III) Ne trichez pas
- IV) Ne détachez pas les agrafes du sujet
- V) Écrivez lisiblement vos réponses (si nécessaire en majuscules)
- VI) Vous devez écrire les algorithmes et structures en langage C (donc pas de Python ou autre)

### 1 Listes chaînées (9 points)

1.1 Écrivez la structure d'une liste chaînée d'entiers (0,5 point)



1.2 Écrivez la fonction *IsEmpty* testant si une liste est vide (0,5 point)

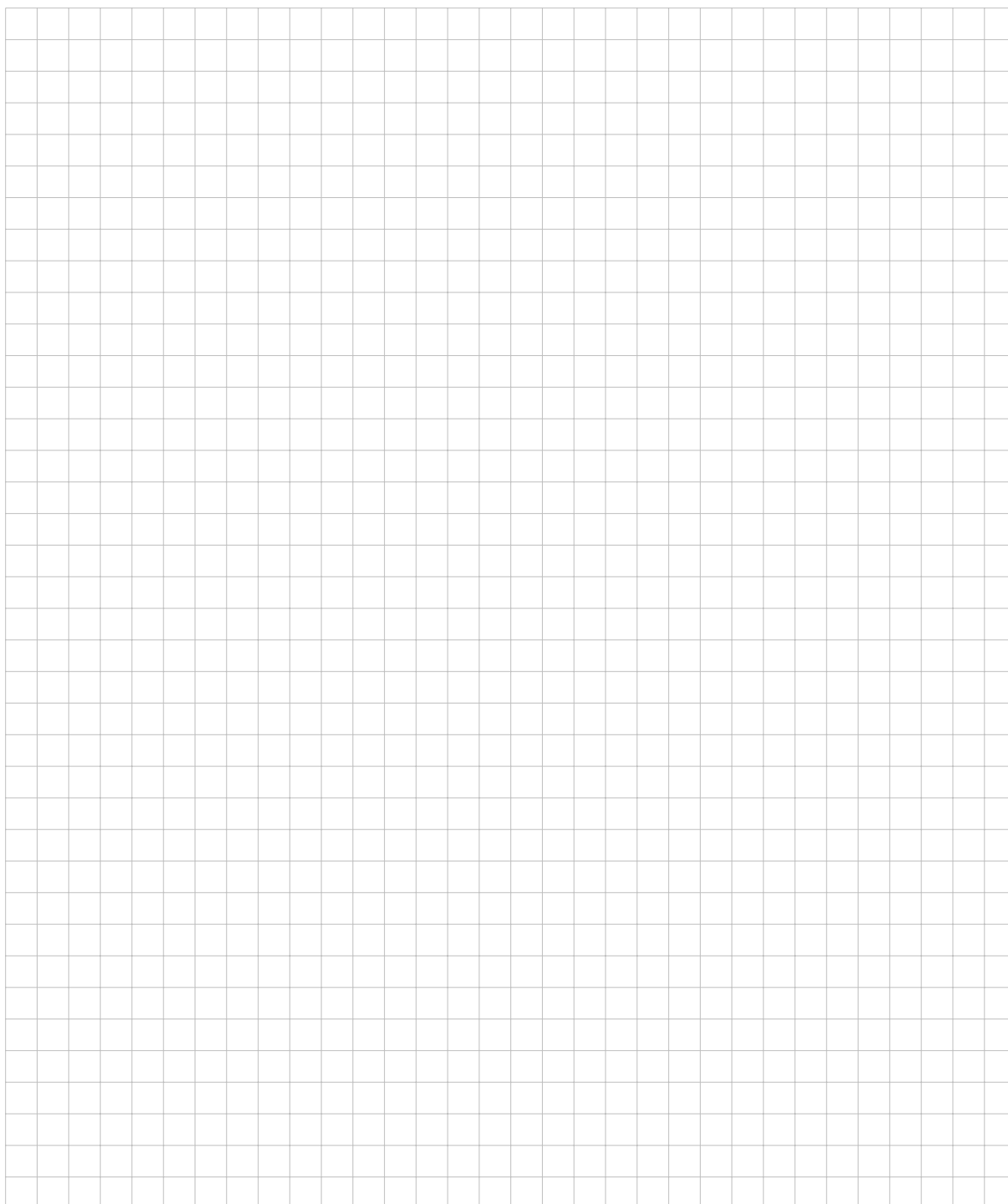


1.3 (1 point) Écrivez la fonction *LengthList* retournant la longueur d'une liste



**1.4 (5 points) Écrivez une fonction *insert\_list* insérant un élément *elt* à la position *pos* dans une liste chaînée *L* et respectant les exigences suivantes**

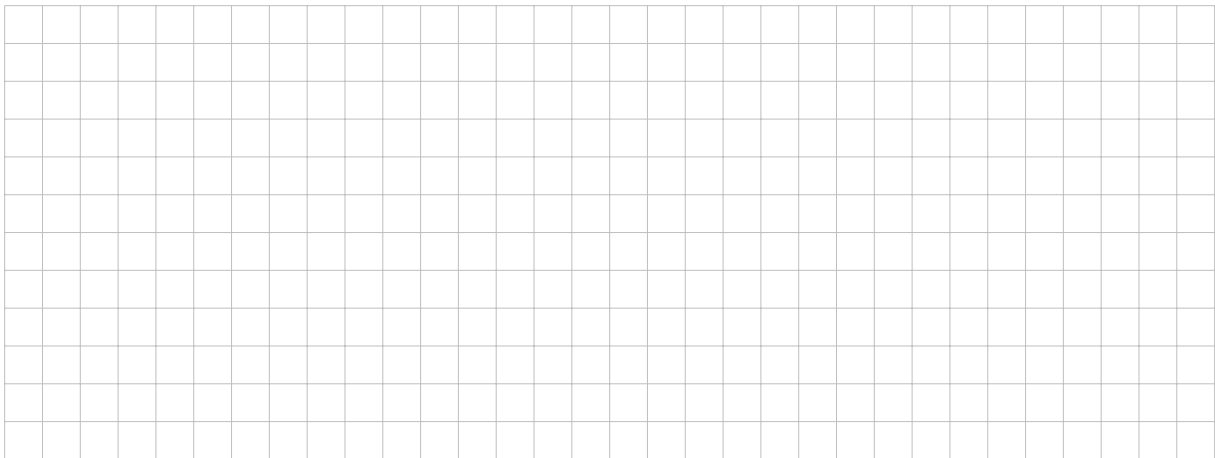
- La fonction doit renvoyer la tête de la liste (éventuellement la nouvelle tête)
- Les entiers insérés doivent être positifs, sinon la fonction ne fait rien et retourne *NULL*
- Le premier élément est considéré comme étant en position 1
- Si la liste est vide, l'élément sera inséré en première position
- Lors de l'ajout, si un élément est déjà présent à la position *pos* donnée en paramètre, alors il faut pousser l'élément existant en position  $pos + 1$
- Si la position *pos* donnée en paramètre est supérieure à la longueur, alors on doit insérer l'élément en dernière position de la liste
- Si la position *pos* donnée en paramètre est inférieure ou égale à 1, alors on doit insérer en première position et décaler l'élément déjà présent s'il y en a un





**1.5** En réutilisant les fonctions précédentes, et en considérant que vous disposez de la fonction *remove\_list* qui supprime l'élément à une position donnée d'une liste chaînée, réécrivez les fonctions *push* et *pop* d'une pile

**1.5.1** (1 point) Push

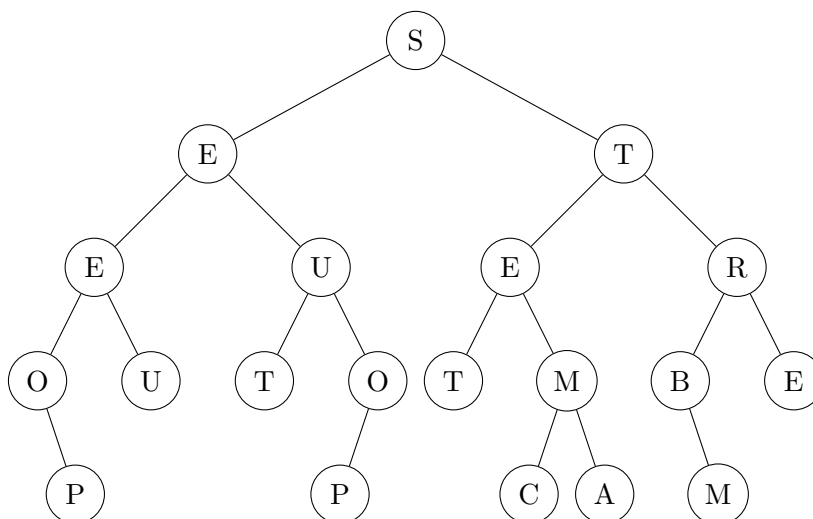


**1.5.2** (1 point) Pop



## 2 Arbres Binaires (11 points)

### 2.1 Répondez aux différentes questions concernant l'arbre suivant (4 points)



#### 2.1.1 (1,5 point) Indiquez toutes les propriétés que possède cet arbre :

Arité : 2

Taille : 20

Hauteur : 4

Nb feuilles : 9

☐ Arbre binaire strict / localement complet

☐ Arbre binaire (presque) complet

☐ Arbre binaire parfait

☐ Arbre filiforme

☐ Peigne gauche

☐ Peigne droit

#### 2.1.2 (2 points) Écrivez les clés lors d'un parcours profondeur main gauche de l'arbre dans les 3 ordres ainsi que lors d'un parcours largeur :

Parcours profondeur :

ordre préfixe : S E E O P U U T O P T E T M C A R B M E

ordre infixe : O P E U E T U P O S T E C M A T B M R E

ordre suffixe : P O U E T P O U E T C A M E M B E R T S

Parcours largeur :

ordre : S E T E U E R O U T O T M B E P P C A M

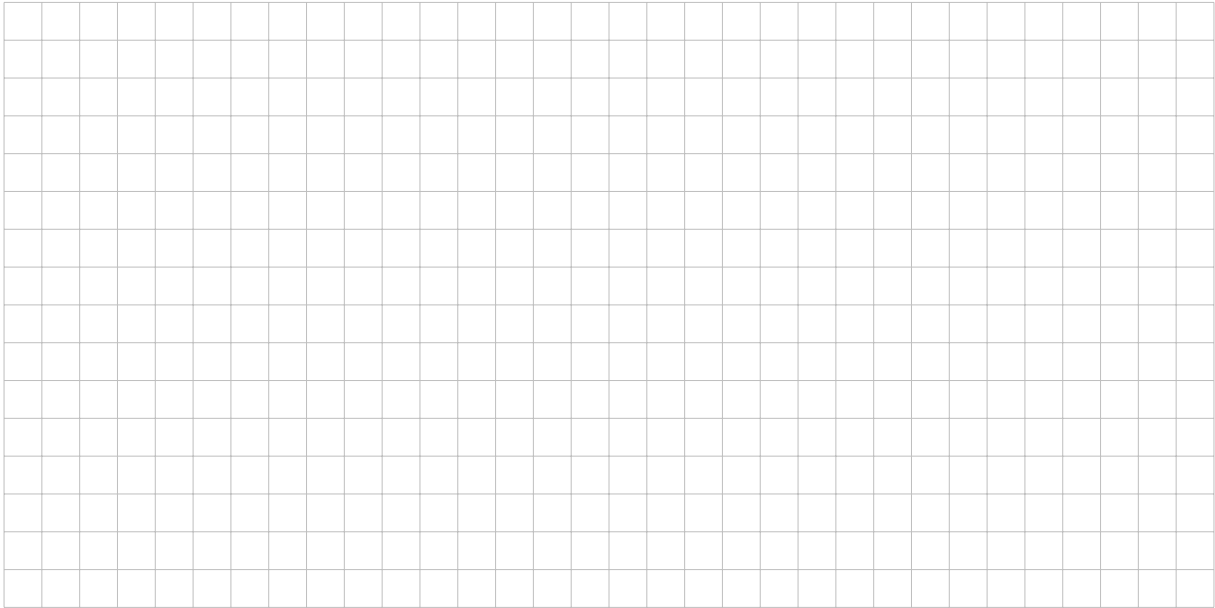
#### 2.1.3 (0,5 point) Indiquez la profondeur et le numéro hiérarchique des nœuds suivants :

|   | Profondeur | N° hiérarchique |
|---|------------|-----------------|
| B | 3          | 14              |

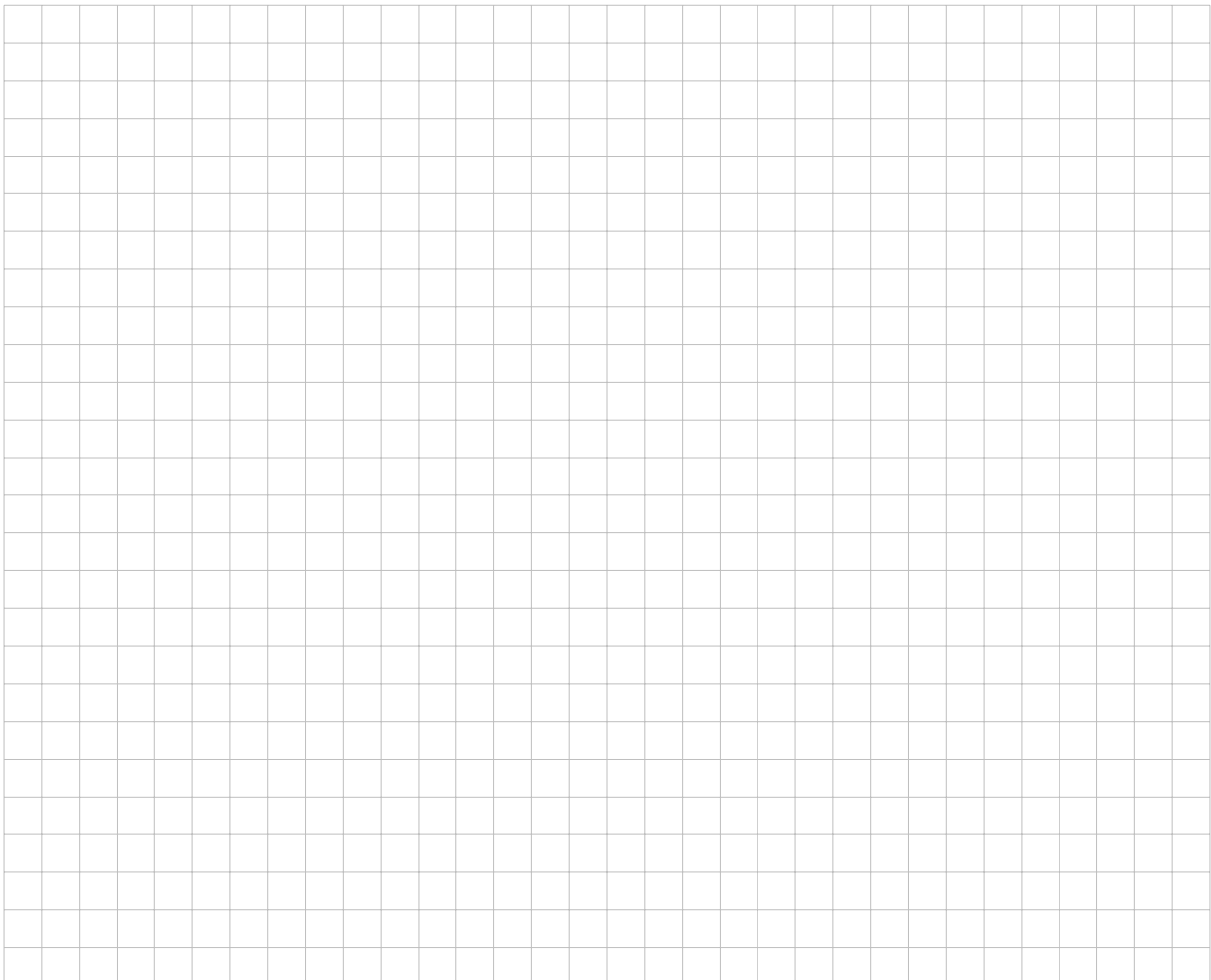
|   | Profondeur | N° hiérarchique |
|---|------------|-----------------|
| C | 4          | 26              |

**2.2 Algorithmes (7 points)**

**2.3 (0,5 point) Écrivez la structure récursive *node* permettant de représenter des arbres binaires de nombres entiers :**



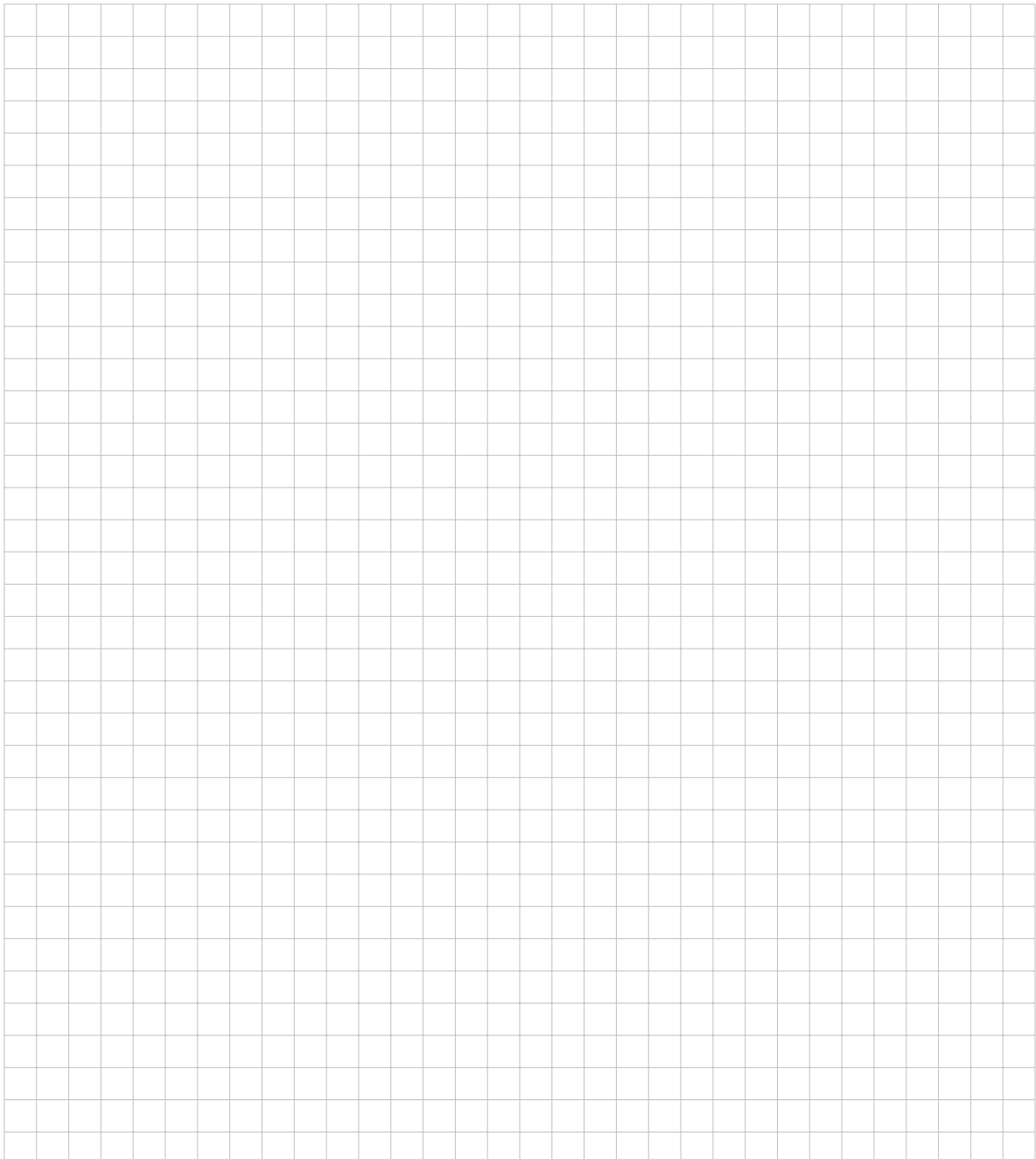
**2.4 (2 points) Écrivez une fonction récursive « *parc\_prof\_rec* » effectuant un parcours profondeur main gauche dans un arbre binaire, et affichant les nœuds dans l'ordre *suffixe* (l'arbre est de type *node\**) :**



**2.5 (2 points) Écrivez une fonction itérative « *parc\_larg\_iter* » effectuant un parcours largeur dans un arbre binaire, et affichant les nœuds (l'arbre est de type *node\**) :**

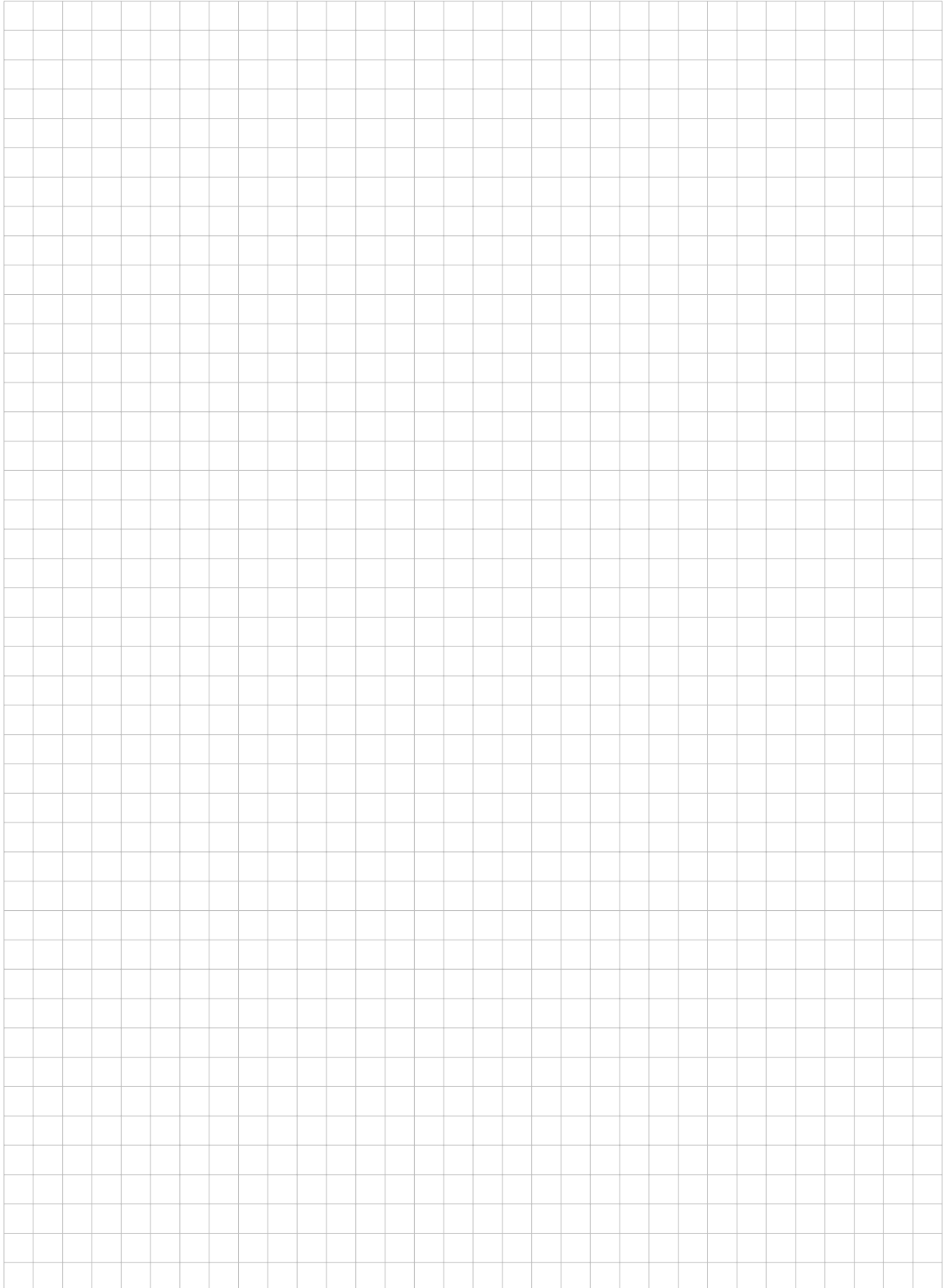
Vous pouvez utiliser les conteneurs externes suivants avec leurs opérations :

| Liste         | File           | Pile           |
|---------------|----------------|----------------|
| <i>list_p</i> | <i>queue_p</i> | <i>stack_p</i> |
| Create        | Create         | Create         |
| Length        | Length         | Length         |
| IsEmpty       | IsEmpty        | IsEmpty        |
| Insert        | Enqueue        | Push           |
| Remove        | Dequeue        | Pop            |
| Clear         | Clear          | Clear          |
| Delete        | Delete         | Delete         |



**2.6 (2,5 points) Écrivez une fonction « *node\_to\_array* » transformant un arbre au format *node\** vers le format tableau *int\** :**

Le tableau est donné en paramètre et est déjà alloué avec la bonne taille : votre fonction ne doit *que* le remplir avec les bonnes valeurs. La taille du tableau est évidemment fournie en paramètre. Un nœud vide doit être représenté par «  $-1$  ».

A large grid of graph paper, consisting of 30 columns and 40 rows of small squares, intended for the student to write their code.

## **CORRECTION SUJET A ALGORITHMIQUE 2**