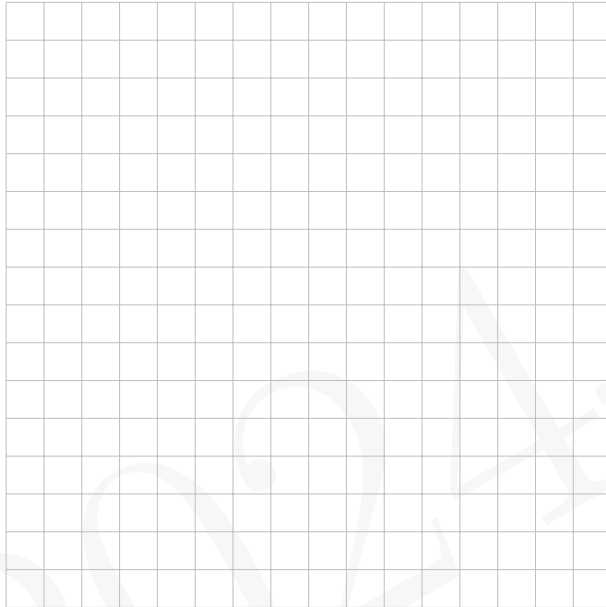
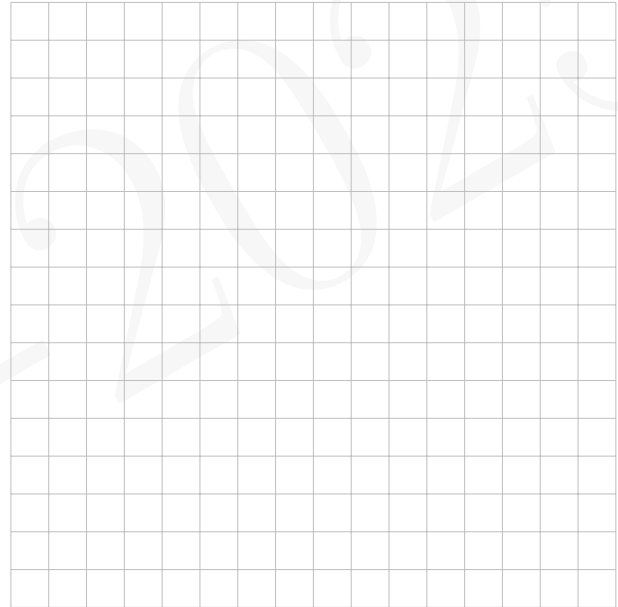


[CYBER1][2024-2025] Examen (Sujet B)
Algorithmique 2

NOM : _____**PRÉNOM :** _____

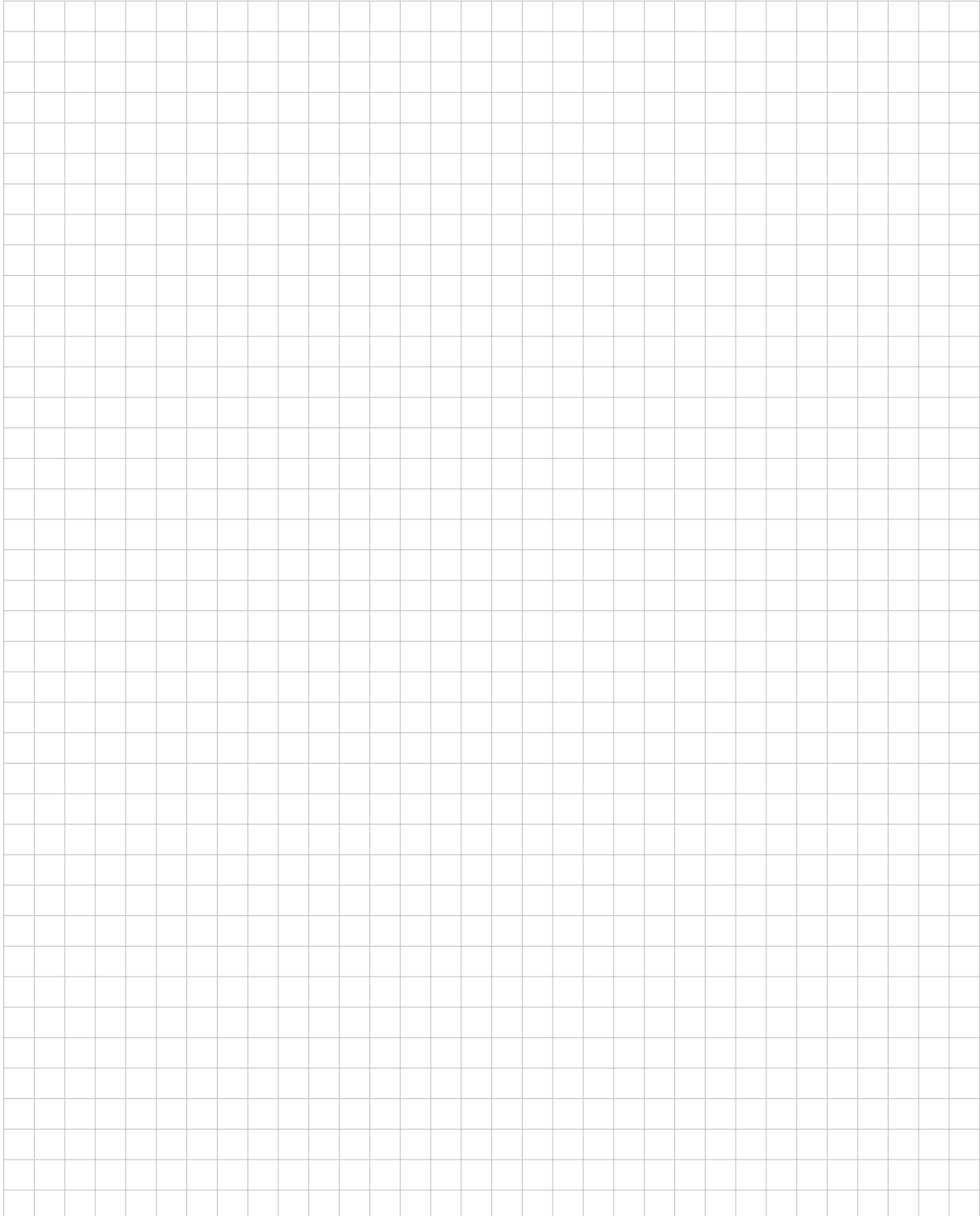
Vous devez respecter les consignes suivantes, sous peine de 0 :

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

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 *remove_list* supprimant l'élément présent 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)
- Le premier élément est considéré comme étant en position 1
- Si la liste est vide, la fonction ne fait rien et retourne *NULL*
- Si la position *pos* donnée en paramètre est supérieure à la longueur, alors on doit supprimer l'élément en dernière position de la liste
- Si la position *pos* donnée en paramètre est inférieure ou égale à 1, alors on doit supprimer l'élément en première position de la liste





1.5 En réutilisant les fonctions précédentes, et en considérant que vous disposez de la fonction *insert_list* qui insère un élément à une position donnée d'une liste chaînée en poussant le suivant à droite, réécrivez les fonctions *enqueue* et *dequeue* d'une file

1.5.1 (1 point) Enqueue

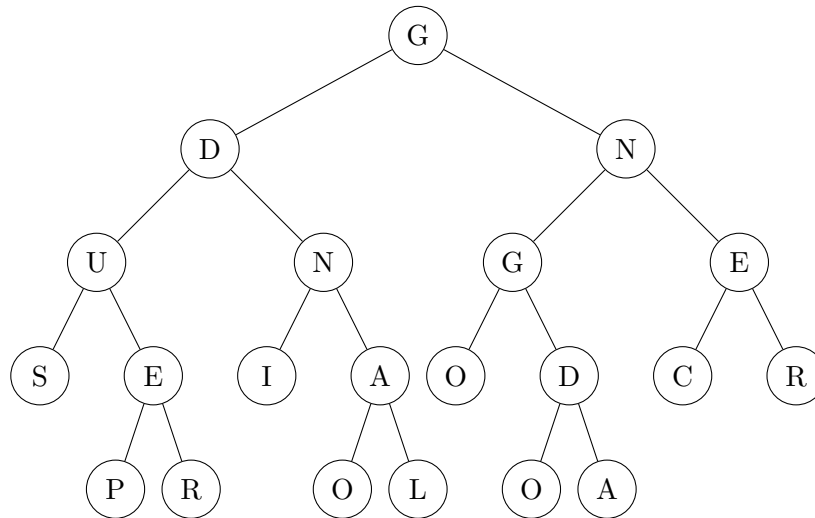


1.5.2 (1 point) Dequeue



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 :

Taille :

Hauteur :

Nb feuilles :

☐

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 : _ _ _ _ _

ordre infixe : _ _ _ _ _

ordre suffixe : _ _ _ _ _

Parcours largeur :

ordre : _ _ _ _ _

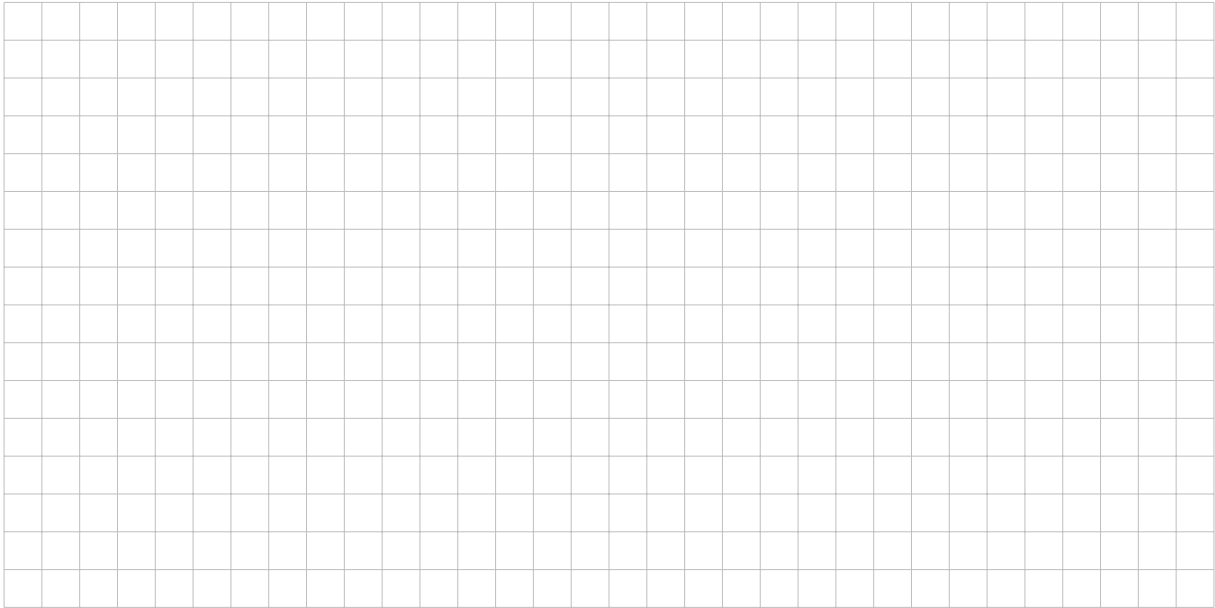
2.1.3 (0,5 point) Indiquez la profondeur et le numéro hiérarchique des nœuds suivants :

	Profondeur	N° hiérarchique
I		

	Profondeur	N° hiérarchique
P		

2.2 Algorithmes (7 points)

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



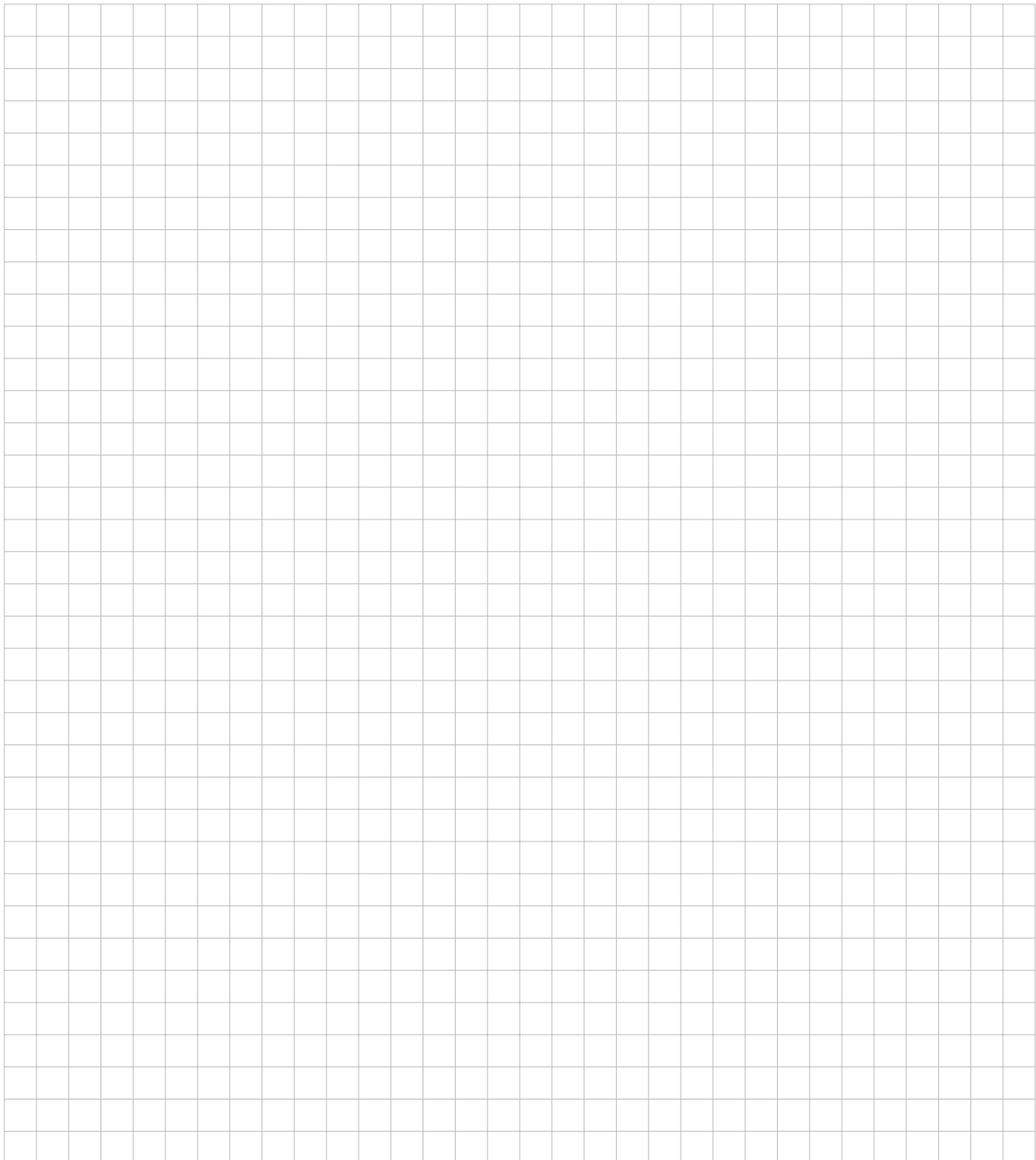
2.4 (2 points) Écrivez une fonction récursive « *size* » calculant la taille d'un arbre binaire (l'arbre est de type *node) :**



2.5 (2 points) Écrivez une fonction itérative « *parc_prof_iter* » effectuant un parcours profondeur main gauche dans un arbre binaire, et affichant les nœuds (l'arbre est de type *node**) :

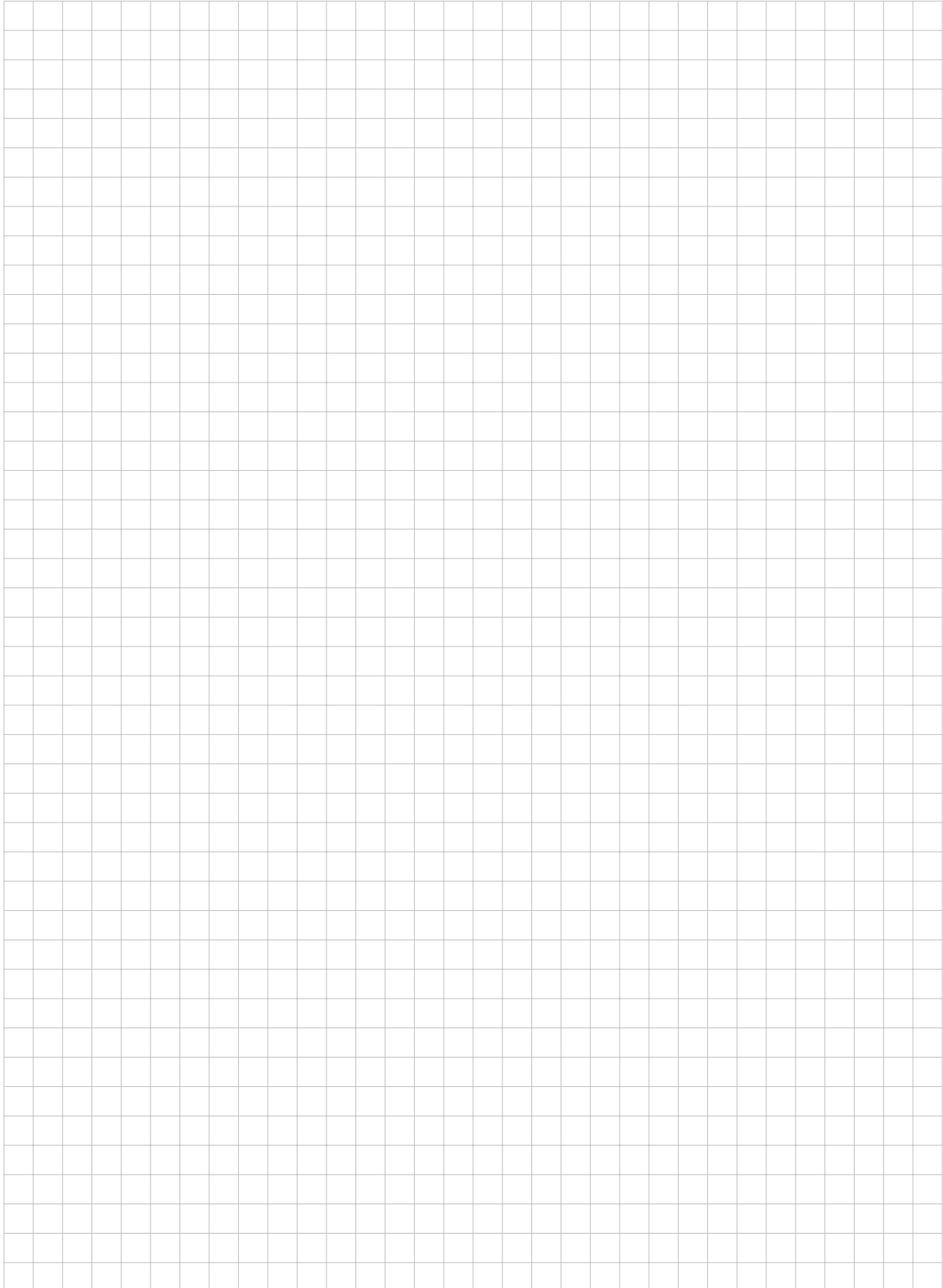
Vous pouvez utiliser les conteneurs externes suivants avec leurs opérations :

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 ».



SUJET B
ALGORITHMIQUE 2