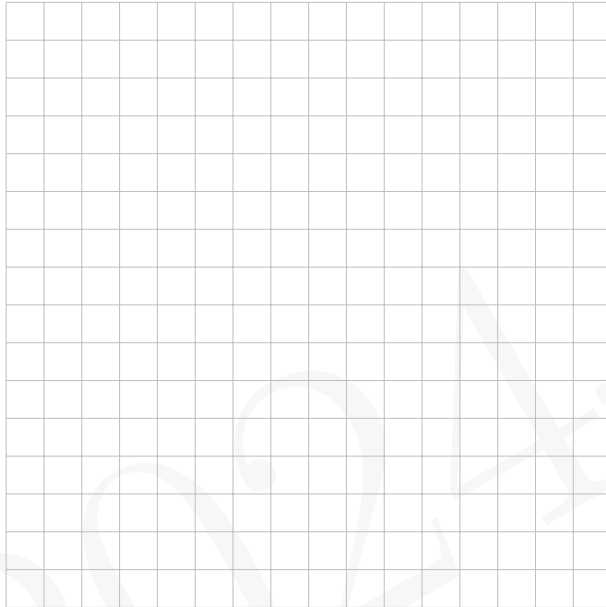
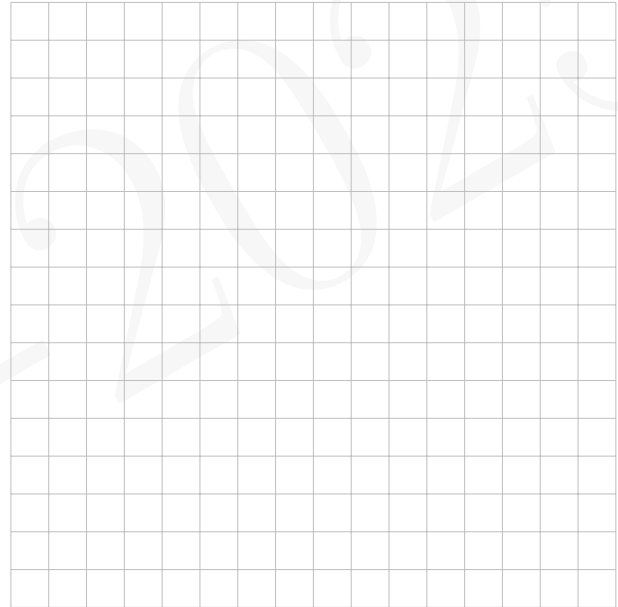


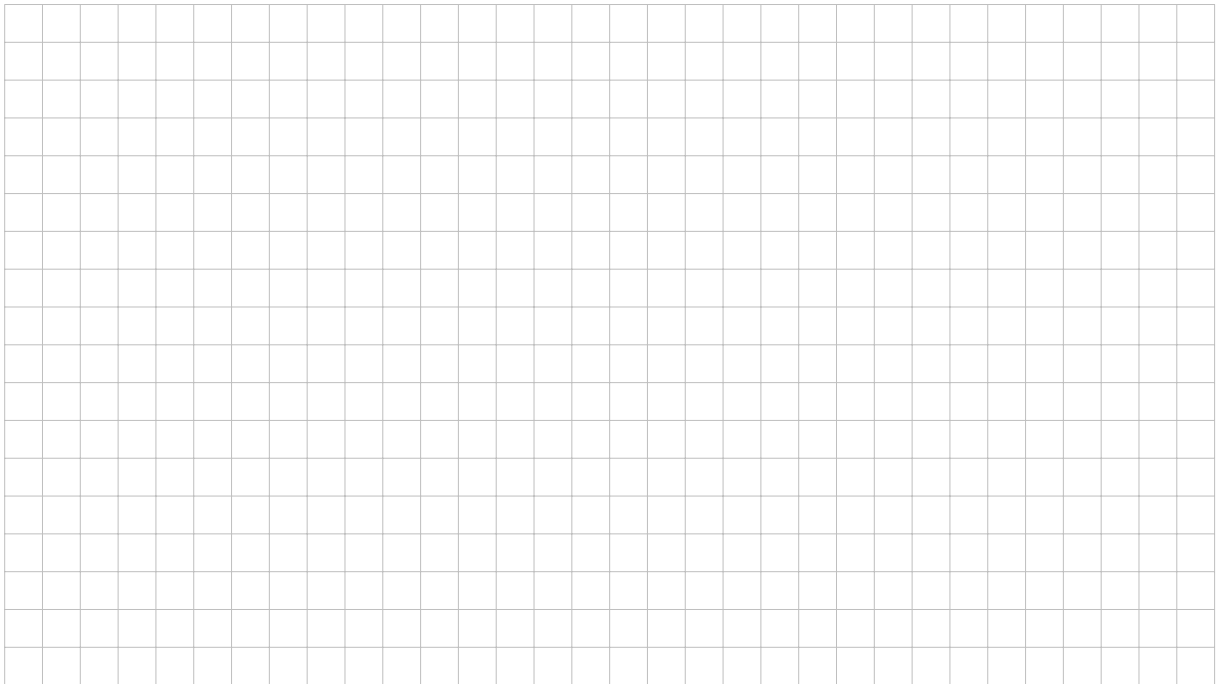
[CYBER1][2024-2025] Examen (Sujet A)
Algorithmique 2

NOM : _____**PRÉNOM :** _____

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

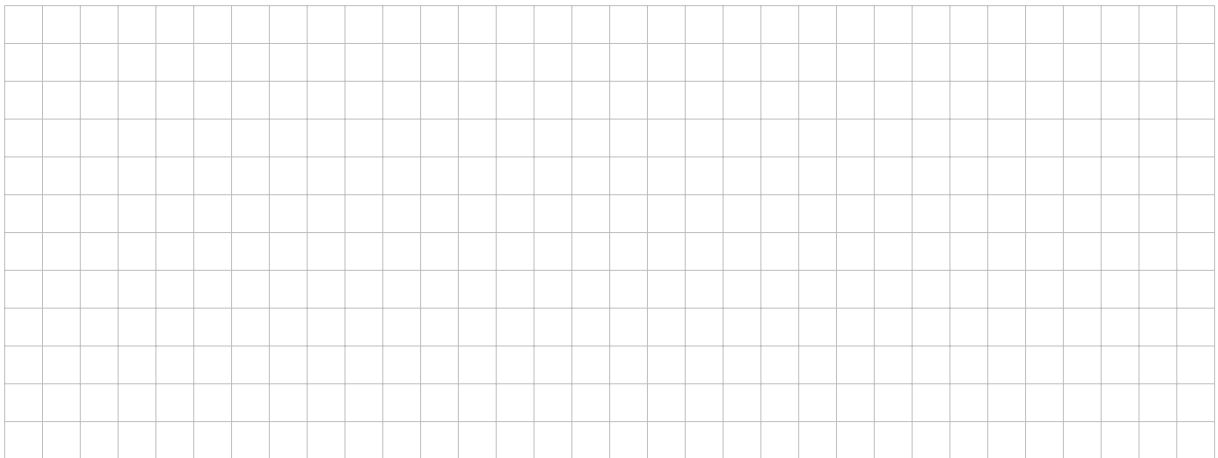
- | | |
|--|--|
| I) Lisez le sujet en entier avec attention | V) Écrivez lisiblement vos réponses (si nécessaire en majuscules) |
| II) Répondez sur le sujet | |
| III) Ne trichez pas | VI) Vous devez écrire les algorithmes et structures en langage C (donc pas de Python ou autre) |
| IV) Ne détachez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

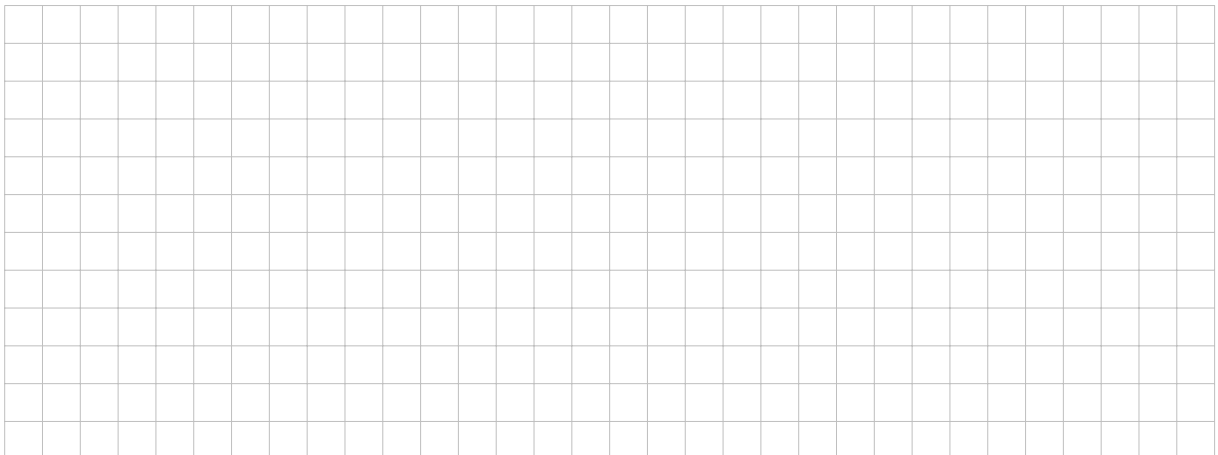


1.5 En réutilisant les fonctions précédentes, et en considérant que vous disposez de la fonction *remove_list* qui supprime l'élément à une position donnée d'une liste chaînée, réécrivez les fonctions *push* et *pop* d'une pile

1.5.1 (1 point) Push

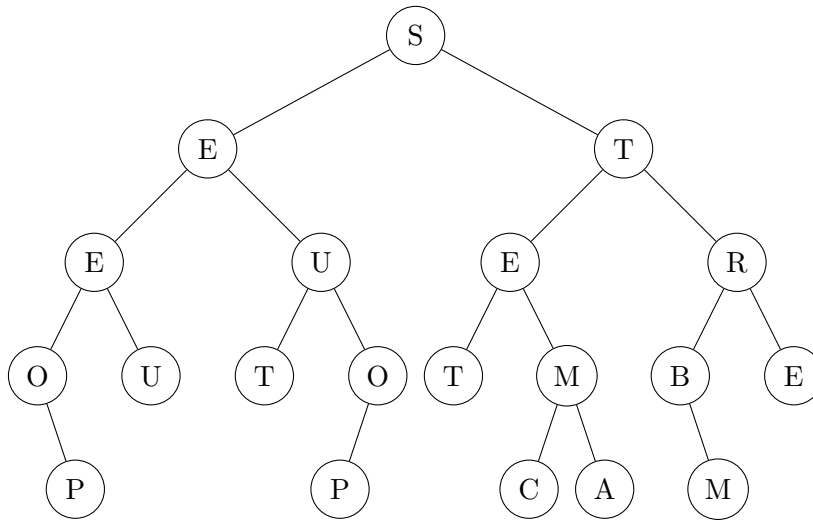


1.5.2 (1 point) Pop



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 :

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 : _ _ _ _ _

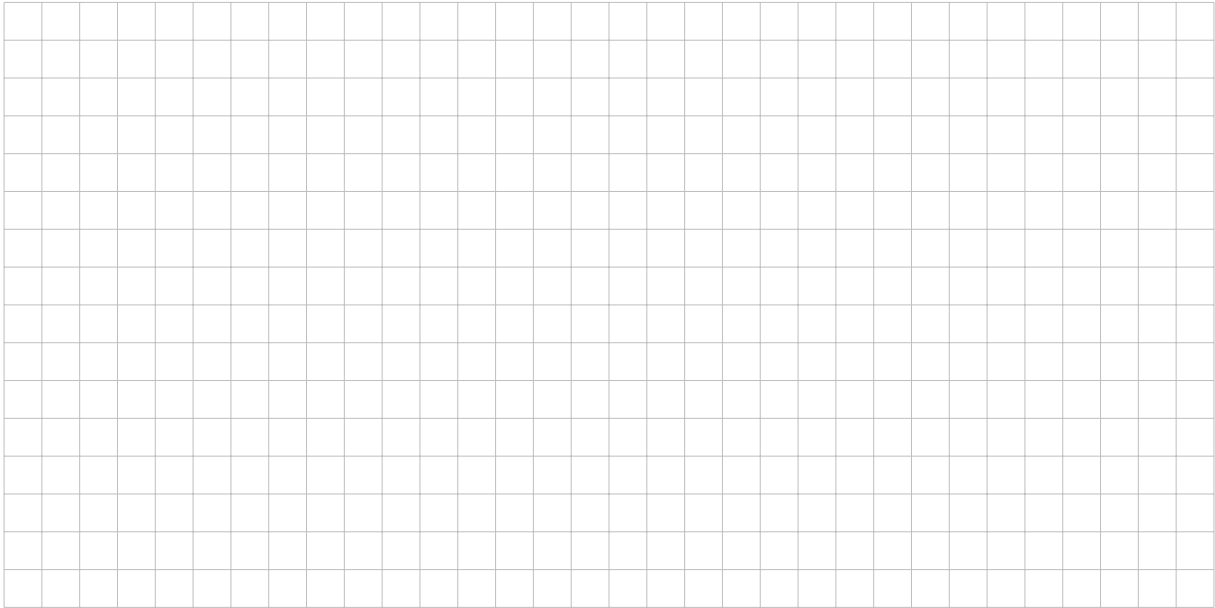
2.1.3 (0,5 point) Indiquez la profondeur et le numéro hiérarchique des nœuds suivants :

	Profondeur	N° hiérarchique
B		

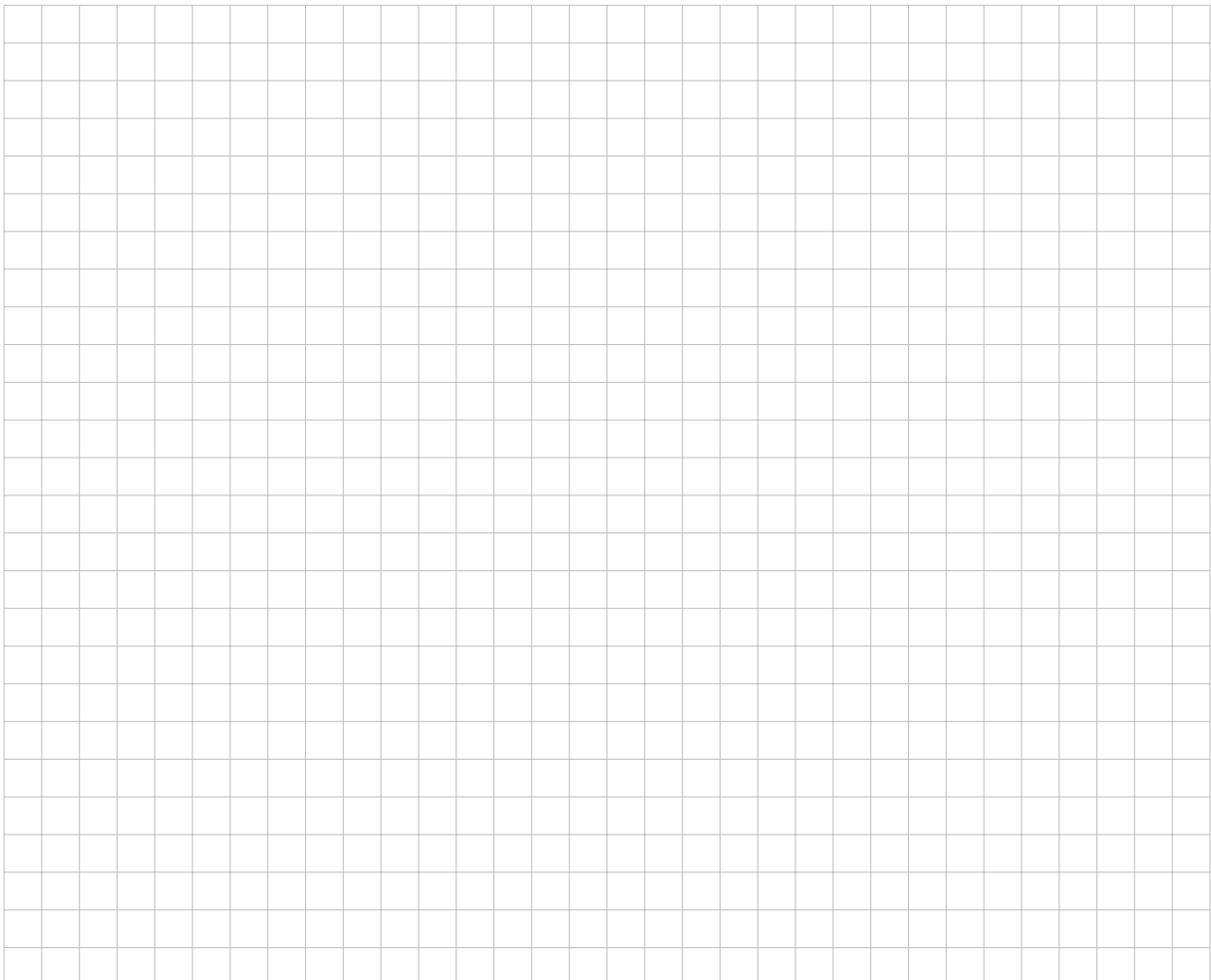
	Profondeur	N° hiérarchique
C		

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) :**



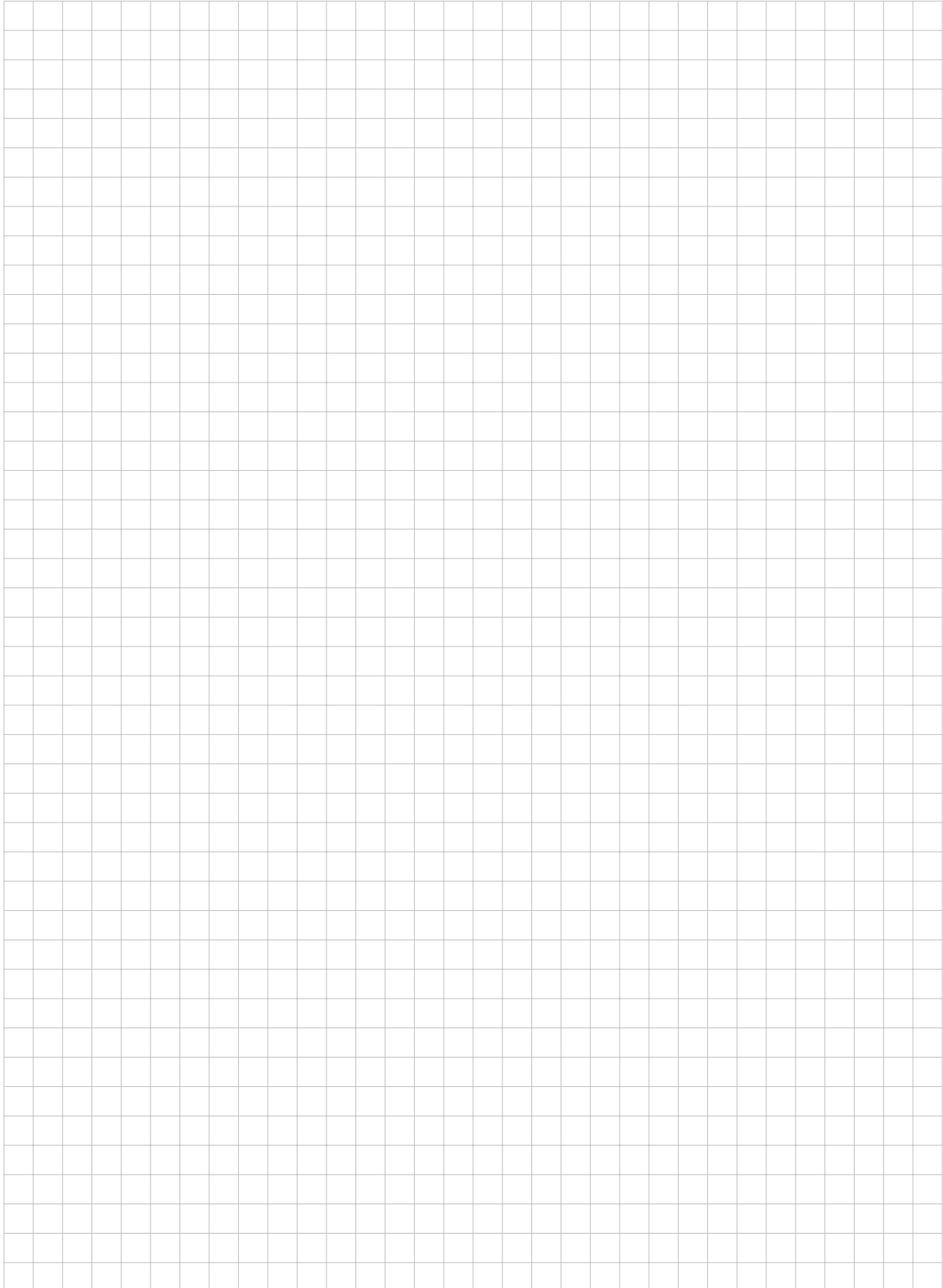
2.5 (2 points) Écrivez une fonction itérative « *parc_larg_iter* » effectuant un parcours largeur dans un arbre binaire, et affichant les nœuds (l'arbre est de type *node**) :

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 ».



SUJET A
ALGORITHMIQUE 2