

CORRECTION Partiel (Sujet 2) - CYBER1 (2h00)

Algo et Structure de Données 1

NOM :

PRÉNOM :

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

- Lisez le sujet en entier avec attention
- Répondez sur le sujet
- Ne détachez pas les agrafes du sujet
- Écrivez lisiblement vos réponses (si nécessaire en majuscules)
- Vous devez écrire dans le langage algorithmique ou en C (donc pas de Python ou autre)
- Ne trichez pas

Dans cet examen, vous allez implémenter avec précisions quelques opérations basiques d'utilisation d'une liste (taille de la liste, récupération du premier élément, récupération du dernier élément, test si la liste est vide, ajout d'un élément à un emplacement précis en poussant les éléments suivants, suppression d'un élément suivi d'une copie des éléments suivants, vider la liste), et vous utiliserez ces opérations pour implémenter une pile et une file.

1 Questions (6 points)

36	15	42			
0	1	2	3	4	5

1.1 (1 point) Donnez les caractéristiques de cette liste :

Taille de la liste :

Taille maximale de la
liste :

Tête de la liste :

Queue de la liste :

1.2 (2,5 points) Indiquez l'état de la liste après chacune de ces opérations, puis précisez à quelle structure de données ce comportement correspond.

36	15	42			
0	1	2	3	4	5

Ajout de 55 en première position en décalant les valeurs vers la fin

55	36	15	42		
0	1	2	3	4	5

Ajout de 14 en première position en décalant les valeurs vers la fin

14	55	36	15	42	
0	1	2	3	4	5

Ajout de 21 en première position en décalant les valeurs vers la fin

21	14	55	36	15	42
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

14	55	36	15	42	
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

55	36	15	42		
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

36	15	42			
0	1	2	3	4	5

Structure de données imitée : **Pile**

1.3 (2,5 points) Indiquez l'état de la liste après chacune de ces opérations, puis précisez à quelle structure de données ce comportement correspond.

36	15	42			
0	1	2	3	4	5

Ajout de 55 en *dernière* position

36	15	42	55		
0	1	2	3	4	5

Ajout de 14 en *dernière* position

36	15	42	55	14	
0	1	2	3	4	5

Ajout de 21 en *dernière* position

36	15	42	55	14	21
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

15	42	55	14	21	
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

42	55	14	21		
0	1	2	3	4	5

Suppression de la première position en décalant les valeurs vers le début

55	14	21			
0	1	2	3	4	5

Structure de données imitée : **File**

2 Algorithmes (14 points)

Le but de l'implémentation est de réutiliser les fonctions que vous écrirez. Si vous ne savez pas implémenter une des fonctions, vous pouvez tout de même l'utiliser dans les exercices suivants.

A - Liste à tableau

- 2.1 (1 point) Écrivez une structure de données « *my_list_t* » pouvant servir de liste contenant des éléments étant des entiers positifs. Cette liste doit être stockée dans un tableau (et non pas une chaîne de pointeurs).



- 2.2 (2 points) Écrivez l'implémentation des fonctions suivantes :

- *length_list* : renvoie la taille de la liste, c'est-à-dire le nombre d'éléments contenus
- *get_first_elt* : renvoie le premier élément de la liste
- *get_last_elt* : renvoie le dernier élément de la liste
- *list_is_empty* : renvoie *vrai* si la liste est vide, sinon *faux*
- *list_is_full* : renvoie *vrai* si la liste est pleine, sinon *faux*

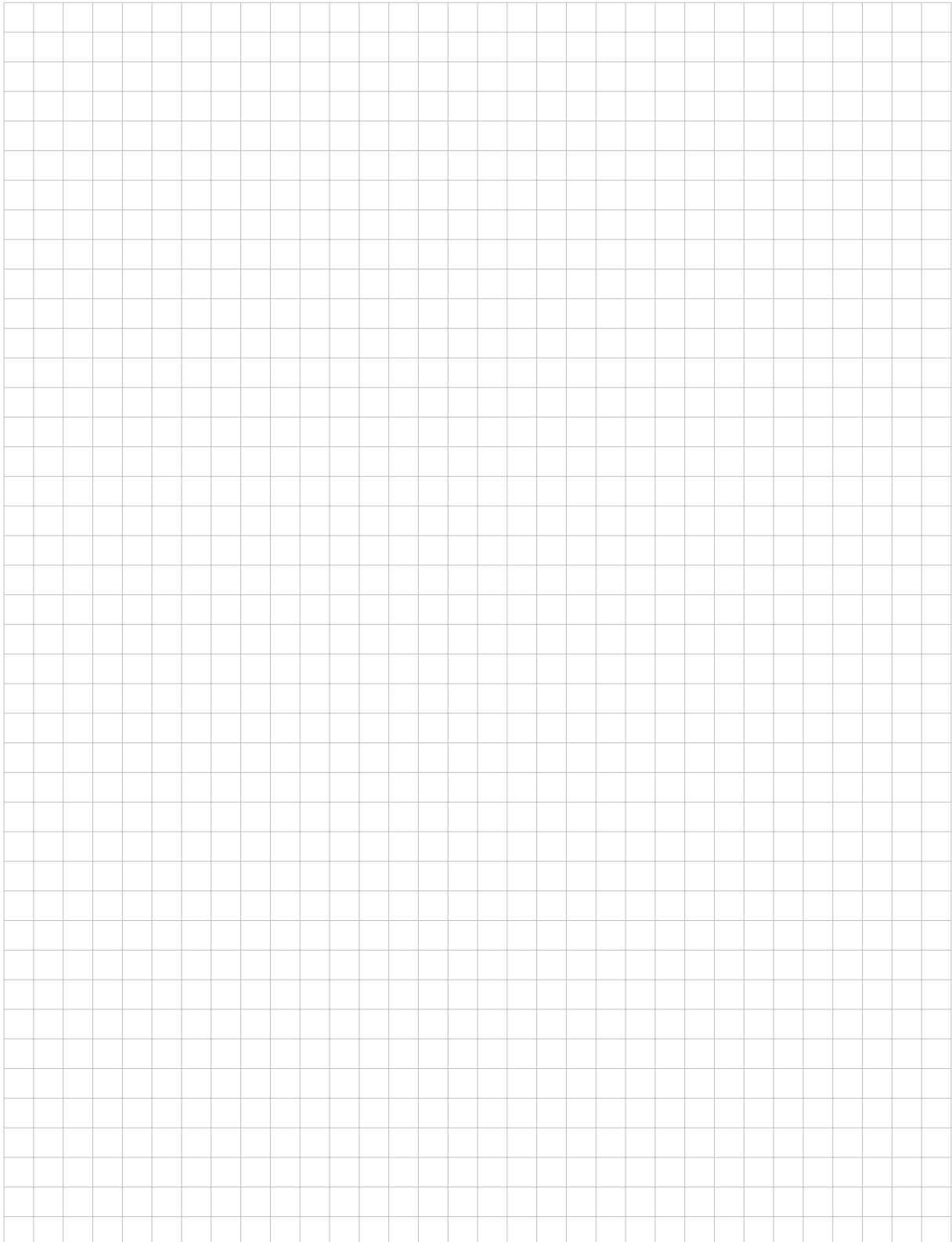
```
int length_list(my_list_t *liste)
```



```
int get_first_elt(my_list_t *liste)
```

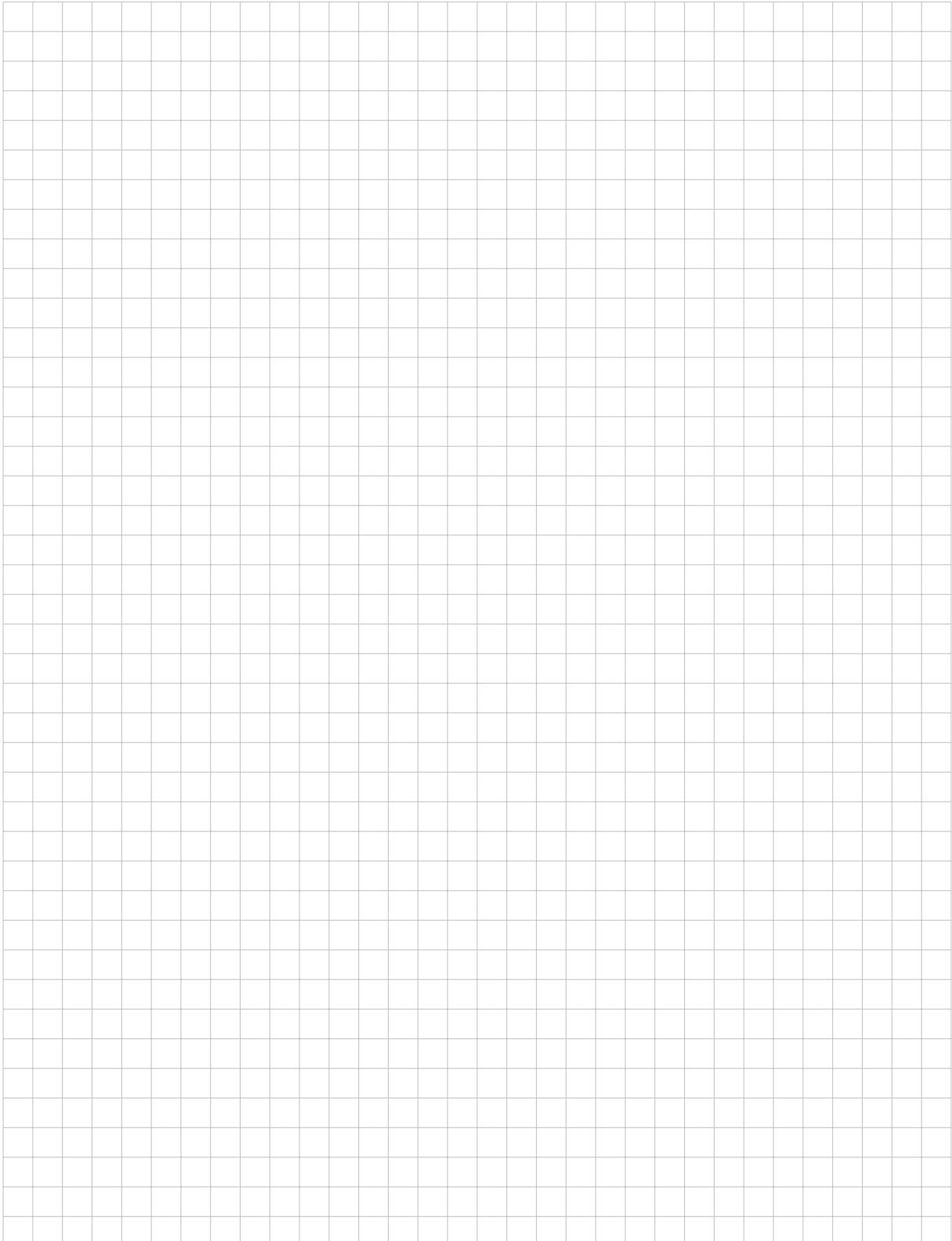
- 2.3 (3 points) Écrivez une fonction « *add_elt_at_pos* » qui ajoute un élément *elt* à la position *pos* dans la liste *liste*, puis renvoie *vrai*. Si la position est déjà occupée, vous décalerez son contenu et celui des cellules suivantes d'un cran vers la fin. Si la liste est pleine ou si la position n'existe pas, vous renverrez **faux** sans rien faire. La position 0 est considérée comme la première position de la liste.

```
bool add_elt_at_pos(my_list_t *liste, int elt, int pos)
```

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

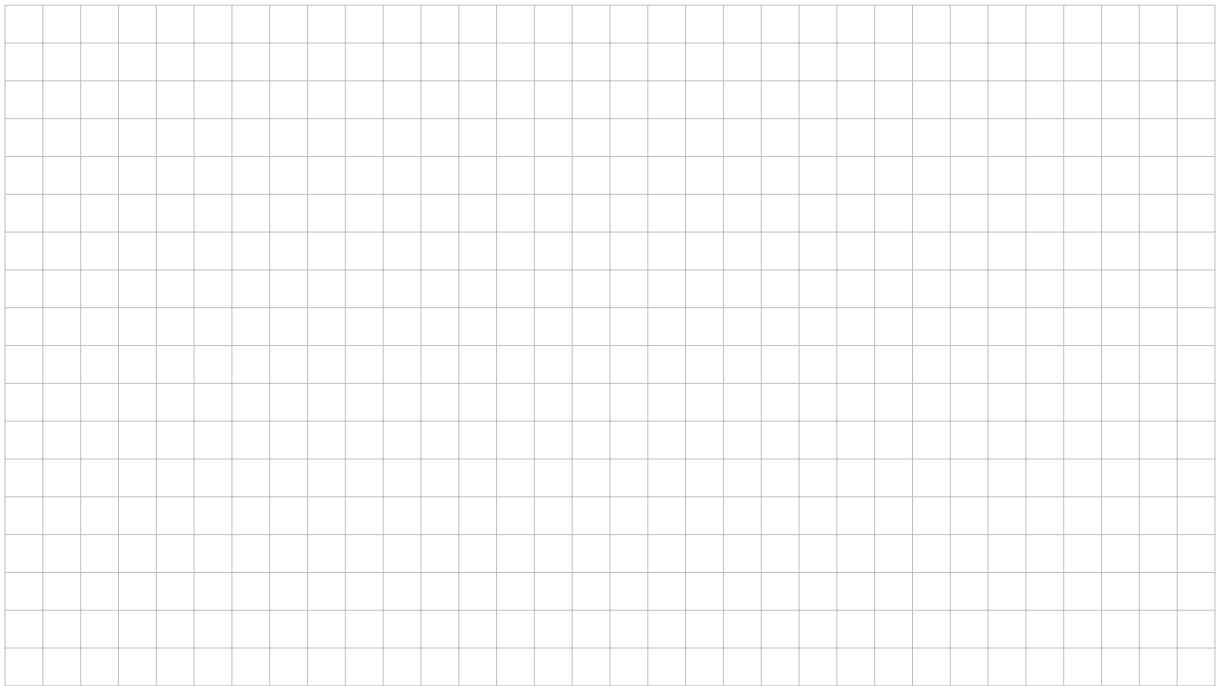
- 2.4 (3 points) Écrivez une fonction « *del_elt_at_pos* » qui supprime l'élément de la position *pos* dans la liste *liste*, puis renvoie *vrai*. Vous décalerez les cellules suivantes d'un cran vers le début. Si la liste est vide ou si la position n'existe pas, vous renverrez **faux** sans rien faire. La position 0 est considérée comme la première position de la liste.

```
bool del_elt_at_pos(my_list_t *liste, int pos)
```



2.5 (1 point) Écrivez une fonction « *remove_list* » qui vide la liste *liste* en supprimant tous les éléments contenus dedans.

```
void remove_list(my_list_t *liste)
```



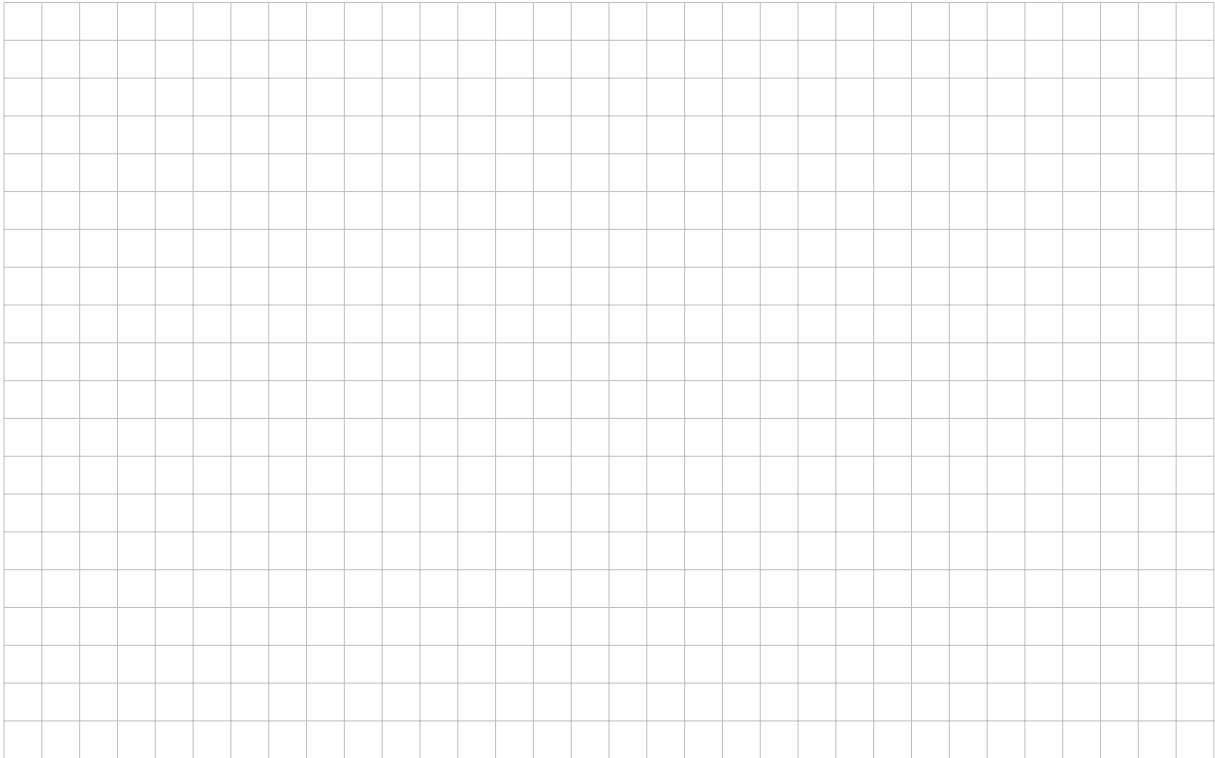
B - Pile et File avec liste à tableau

Vous allez maintenant utiliser les fonctions que vous venez d'écrire pour implémenter des piles et des files. Vous devez réutiliser les fonctions existantes (non pas en les réécrivant, mais en les appelant). N'hésitez pas à relire vos réponses de la partie *Questions* pour vous aider.

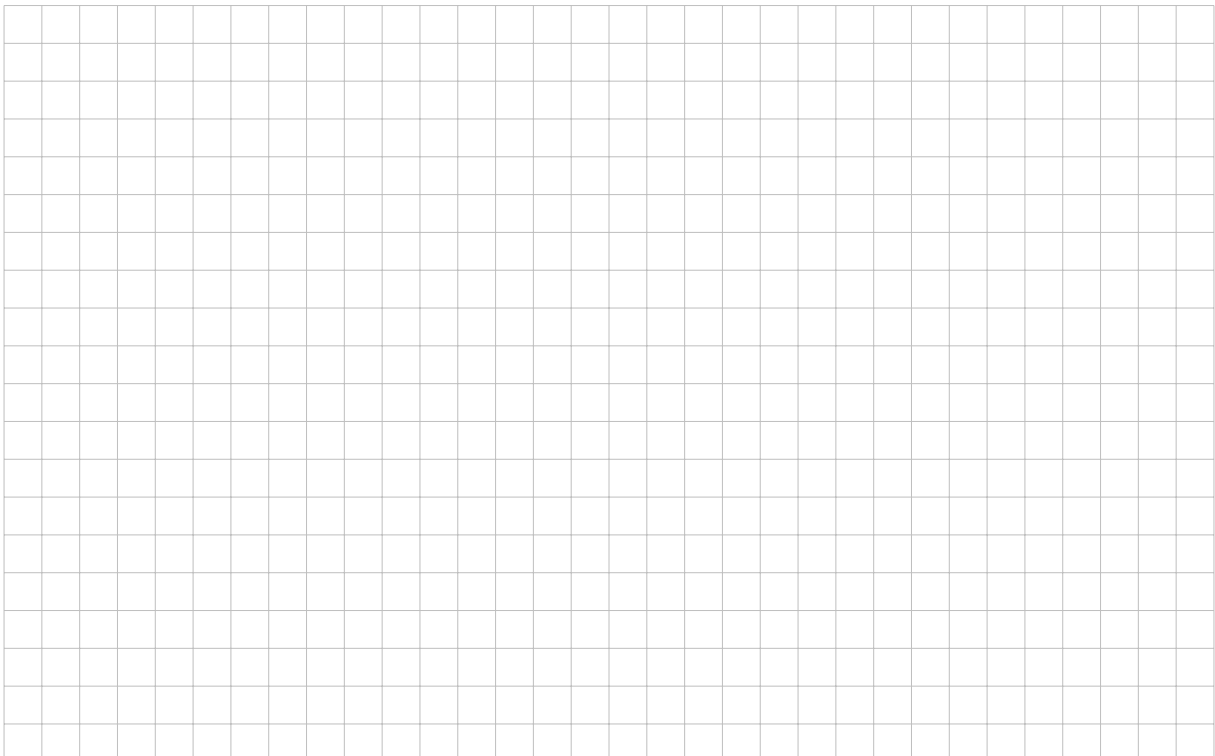
2.6 (1 point) Écrivez une fonction « *push* » pouvant servir à empiler un élément dans une liste.



2.7 (1 point) Écrivez une fonction « *pop* » pouvant servir à dépiler un élément depuis une liste.



2.8 (1 point) Écrivez une fonction « *enqueue* » pouvant servir à enfiler un élément dans une liste.



2.9 (1 point) Écrivez une fonction « *dequeue* » pouvant servir à défiler un élément depuis une liste.

