Les trois premiers exercices sont chacun donné à un élève puis les suivants sont donnés de manière gloutonne: Le premier à finir son exercice fait le premier exercice non fait. Une preuve ou une justification est systèmatiquement attendue.

Exercice 1 Expliquez comment construire deux piles avec un tableau. Les opérations push et pop doivent être en $\mathcal{O}(1)$. Si la somme des tailles de piles dépasse la taille du tableau la structure renvoie "Overflow". Expliquez comment construire une file avec deux piles avec une complexité amortie constante sur les push & pop.

Exercice 2 Donnez un pseudo-code C qui prends en entrée une liste-chainée et renvoie un pointeur sur le maillon au milieu de la liste (On suppose la liste de longueur impaire). On suppose que la taille de la liste est un entier si grand que vous ne pouvez pas le stocker, comment faire ?

Exercice 3 Donnez un algorithme qui prends en entrée une liste chaînée et qui détermine en espace et en temps linéaire si cette liste est un palindrome. Comment faire si l'espace doit être constant ?

Exercice 4 Décrire un algorithme qui prends en entrée n entiers dans l'ensemble $\{1, \ldots, k\}$, qui effectue un pré-traitement en $\mathcal{O}(n + k)$ puis est en mesure de déterminer en $\mathcal{O}(1)$ combien d'entiers (parmis les n en entrée) sont dans l'intervalle $[a, b]$ pour $a, b \in \mathbb{R}$

Exercice 5 Si toutes les clés d'un arbre binaire de recherche sont distinctes, le successeur d'un noeud x est le noeud avec la plus petite clé supérieure à celle de x. Donnez un algorithme qui trouve le successeur d'un noeud x dans un tel arbre.

Exercice 6 Expliquez comment trouver le min et le max d'un tableau d'entiers de taille arbitraire en faisant le moins de comparaison possible. (La réponse attendue est 0 mais la discussion peut être intéressante si l'élève ne pense pas au tri-base)

Exercice 7 Donnez une borne inférieure sur le nombre de comparaisons effectuées par un algorithme de tri par comparaisons dans le pire cas. $(n \log(n))$

Exercice 8 Donnez une borne inférieure sur le nombre de comparaisons effectuées par un algorithme de fusion de listes (de mêmes tailles) triées dans le pire cas. (Remarquer qu'il y a $\binom{2n}{n}$ façons de répartir les $2n$ éléments dans les deux listes: $2n - o(n)$)