IFT339 - Exercices sur les tableaux, files et liste chaînées

Exercice 1.
Écrivez une version récursive de la fouille dichotomique, avec une fonction qui reçoit un tableau *tab* trié (avec accès à $tab.size()$), un élément x à trouver, et les indices *min* et *max* qui déterminent la région du tableau à chercher.

Votre code devrait se rappeler lui-même récursivement, et ne devrait pas avoir besoin d'une boucle. En classe, on a argumenté que le nombre de tours de boucle de la version non-récursive était $O(\log n)$. Ici, il n'y a pas de boucle, alors comment peut-on analyser le temps O de la fonction?

Exercice 2.
Vous voulez trier les personne étudiantes de votre classe en ordre croissant de leur note en IFT339. Une personne est représentée par une struct

```
struct Etudiant{
    string nom;
    string prenom;
    string cip;
    uint32_t note_IFT339;
};
```

où $note\_IFT339$ est un entier entre 0 et 100. Notez que pour bien faire les choses, cette classe devrait s'appeler PersonneEtudiante, mais c'est plus long à écrire.

Écrivez une fonction qui reçoit un vecteur de Etudiant, et qui retourne un autre vecteur du même type, dans lequel les personnes sont triées en ordre non-décroissant de leur note en IFT339 (si deux personnes ont la même note, leur ordre relatif n'est pas important). Le vecteur retourné devrait contenir des copies des personnes.

Exercice 3.
Donnez le code pour la fonction erase_batch de la classe vector. Cette fonction reçoit un argument vector<size_t> pos, qui est une liste **triée** de positions à effacer dans le vecteur. Visez $O(n)$, où n est la taille du vecteur courant.

Par exemple, si *tab* contient [1 5 2 9 7 2] et que *pos* = [2 4], alors après l'appel *tab* contiendra [1 5 9 2].

Exercice 4.
Reconsidérez l'exercice précédent, mais dans le cas où la liste *pos* n'est **pas triée**. Pouvez-vous tout de même atteindre un temps de $O(n)$?

Exercice 5.
Vous maintenez le compte d'une population infectée par un virus. On suppose que la population est infinie. Au début de chaque jour, s'il y a p personnes infectées, il y aura p nouvelles personnes infectées à la fin de la journée (une façon de voir est que chaque personne infectée va en infecter une autre dans la journée). Toutefois, toute nouvelle personne infectée guérira dans k jours, et ne sera plus infectée.

Par exemple, si $k = 3$ et que Robert se fait infecter au jour 5, alors aux jours $6, 7, 8$, Robert infectera une nouvelle personne, puis ne sera plus sur la liste des infectés au jour 9.

Étant donné un nombre d'infectés p, le nombre de jours de guérison k, et un entier n, donnez un algorithm qui calcule le nombre de personnes infectée après n jours. Quelle est votre complexité?

Exercice 6.
Implémentez l'opérateur == d'une liste simplement chaînée (deux listes sont égales si tous leur éléments aux positions correspondantes sont égaux). Cet opérateur reçoit une autre liste chaînée.