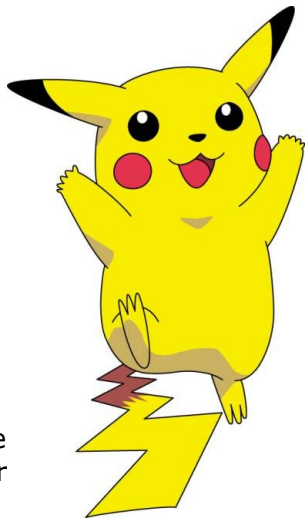
TD d'algorithmique avancée n°3



1 SAUVEZ-LES TOUS !

On continue le TD n°1 sur les Pokemons. On décide que la structure de données Pokemon contiendra un champs Pokemon, permettant de lier les Pokemons les uns aux autres...

```
structure Pokemon
{
	String nom;
	String type;
	int puissance;
	Pokemon suivant;
}
```

De cette manière, on créé ce qu'on appelle une liste chaînée, c'est à dire un ensemble d'éléments liés les uns aux autres. Chaque élément de la liste (ou de la chaîne) est appelé un maillon.

1. Quel élément permet de retrouver tous les autres éléments de la liste ?

Imaginez que vous possédez une liste de Pokemons.

2. Comment supprimer un élément de cette liste (réfléchissez à la donnée d'entrée) ?

3. Et si la liste est triée ?

4. Comment ajouter un élément à la liste ?

5. Comment connaître la taille de la liste ? Voyez-vous un moyen plus simple pour obtenir la taille de la liste (qui inclurait de rajouter une structure de données) ?

6. Comment accéder (et renvoyer en sortie) au i° élément de la liste (par exemple comment accéder aux 4° élément de la liste) ? Et si la collection de Pokemons n'était pas une liste mais un tableau ?

2 4CHAN

4chan est un forum assez populaire sur Internet. Sa particularité est de garantir l'anonymat complet de ses contributeurs : aucune adresse IP n'est conservée en mémoire, et les sujets les moins populaires sont remplacés par les plus populaires (pas de stockage à long terme des messages).

Nous allons tenter de reproduire le comportement de ce forum avec des listes. Dans un premier temps, nous définirons une liste **SujetDeDiscussion**, qui permettra de stocker tous les messages relatifs à un sujet de conversation précis. Puis, nous devrons trouver un moyen de stocker au maximum 150 **SujetDeDiscussion**, et prévoir les fonctions permettant d'ajouter un **SujetDeDiscussion** tout en en supprimant un.

1. Définissez la structure **ListeMessage** qui contient un champ *Auteur* (de type String), un champ *Contenu* (de type String), et ?... Puis, définissez la structure **SujetDeDiscussion**, qui contient un champ *DateDeDerniereContribution* (de type entier), un champ *Sujet* (de type String), et un champ *Messages* (de type **ListeMessage***)* qui pointera vers le premier message du sujet de discussion.

2. Faîtes une fonction **AjouterMessage**, qui prend en paramètre un **SujetDeDiscussion**, un nom d'auteur et un message, et qui ajoute un message au sujet de discussion (vous devrez utiliser la fonction fictive Instant(), qui renvoie un entier avec la date et l'heure du jour (à la milliseconde près).

3. Vous devez trouver maintenant un moyen de stocker au plus 150 messages de discussion dans une structure. Avant de vous lancer, réfléchissez au fait que vous devrez faire une fonction **AjouterSujetDeDiscussion**, qui prendra en paramètre un sujet, un auteur et un message, qui recherchera dans la structure choisie le sujet de discussion avec la date de contribution la plus ancienne, et le remplacera par le nouveau sujet de discussion.

4. Imaginons que chaque **SujetDeDiscussion** possède un nouveau champ *NumeroIndentiteUnique*, qui l'identifie de façon unique par rapport aux autres **SujetDeDiscussion**. De quoi auriez vous besoin pour faire une fonction qui ajoute rapidement un message à un sujet de discussion qu'on ne possède pas (c'est à dire que l'on ne possède pas la structure **SujetDeDiscussion** à proprement dit, mais on peut posséder autre chose, comme par exemple le *NumeroIndentiteUnique* du sujet de discussion auquel on souhaite ajouter un message.

Attention : prenez en compte le fait que pleins de gens sont connectés en même temps sur votre site et y font des opérations.