Université Paris Diderot **Langages à objets avancés**
M1 **Année 2015-2016**

TP n° 4

Exercice 1 Implémentez ou complétez votre implémentation des listes simplement chaînée d'entiers :

1. On définira une classe `CelluleChainee` pour représenter les cellules de la liste et une classe `ListeChainee` qui contienne uniquement un pointeur vers une cellule. (tête de liste). On rappelle que le pointeur nul est appelé `nullptr` en `C++` (C++11).
2. Écrivez les méthodes qui ajoute un élément au début d'une liste, qui ajoyte en fin d'une liste et qui affiche une liste. (Il faudra éventuellement écrire ds méthodes dans `CelluleChainee`.)
3. Construisez à la main une liste chaînée de 3 éléments.
4. On souhaite faire particulièrement attention aux ressources mémoire, modifiez le destructeur de vos cellules pour qu'il affiche, par exemple, "destruction de la cellule de valeur 1". Assurez vous que la mémoire soit bien totalement restituée en fin de programme.

Gestion de tâches : Modélisation

On considère les tâches que doivent exécuter un jour donné des employés d'une petite entreprise.

Pour chaque tâche, on déclarera un certain nombre de dépendances : c'est-à-dire des tâches qui doivent être exécutées avant celle-ci. Par exemple, Jean ne peut mettre sous enveloppe la facture pour la société "Truc" que si Armand l'a rédigée et qu'elle a été validée par Béatrice. Et Judith ne peut aller à la poste que lorsque tous les courriers prévus ce jour-là ont été mis sous enveloppe.

Les dépendances induites ne seront pas forcément stockées.

Il est conseillé de tester au fur et à mesure, le diffrentes classes et méthodes.

— Décrivez des classes `Tache` et `Employe`.
On utilisera les listes chaînées vues précédemment (et adaptées) pour stocker les tâche à faire et les dépendances. Une tâche donnée sera représentée par un objet unique. On comparera donc les tâches entre elles en comparant leurs adresses. Les tâches auront un attribut booléan `faite` qui indiquera si la tâche a été effectuée ou non.
— Écrivez ces classes.
— On veut maintenant faire une classe `Entreprise`. Cette classe comportera entre autres des méthodes permettant d'ajouter une tâche à un employé et aussi d'ajouter une dépendance à une tâche.
— Écrivez ensuite une méthode permettant de pour un employé donné une des tâches (s'il en existe une) que l'on peut faire tout de suite, i.e. qui ne dépend d'aucune autre.
— On va maintenant dans `Entreprise` écrire un algorithme qui va donner un ordonnancement possible de ses tâches : On regarde si le premier employé peut faire une tâche, si oui il l'exécute. Puis on passe à l'mployé suivant, et ainsi de suite jusqu'à ce qu'il n'y ait plus tâche à faire.
On se contentera d'afficher les différentes tâches effectuées et les employés qui les font.

— Écrire une méthode qui appelée à la fin de la journée, va détruire toutes les tâches.

S'il reste du temps...

Représentez les tâches et leur dépendances par un graphe implémenté par liste d'ajacence. Et trouvez un ordonancement avec la méthode du tri topologique : Parcours en profondeur du graphe avec numérotation (priorité) de la dernière visite du sommet et tri par odre de priorité décroissante.