UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 339

Travail pratique#3 : Liste chaînée de liste chaînée

Dans ce travail, vous allez implémenter une liste avec une liste chaînée de liste chaînée. Quelques modalités :

— Ce travail compte pour **5.5 %** de la session.

— La date de remise de ce travail est le **jeudi 10 octobre** jusqu'à 23h59. Nous allons **tolérer un retard** de 4 jours ouvrables et les remises seront acceptées jusqu'au 16 octobre (mais pas plus tard).
Vous devez remettre sur `https://turnin.dinf.usherbrooke.ca/`.

Partie 1 : liste via liste de liste

Comme dans le devoir 2, vous devez implémenter le type de données abstrait *liste*, qui permet : d'accéder au i-ème élément ; d'ajouter/retirer en début/fin ; d'obtenir la taille. Cette fois-ci, nous voulons utiliser les *listes chaînées*.

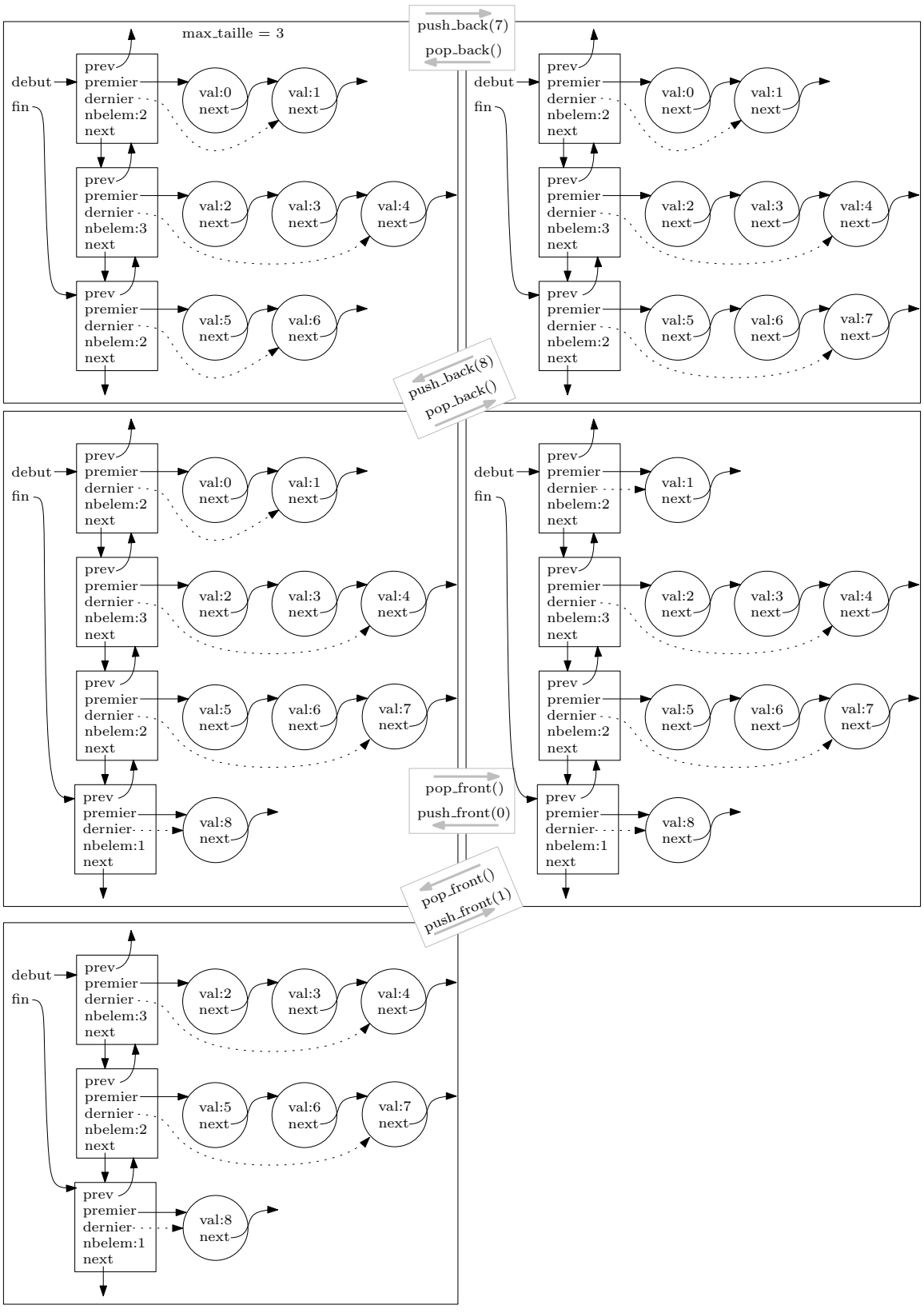
Par contre, on a vu que l'accès à l'élément i prend un temps $O(n)$. Une façon d'y remédier partiellement est d'implémenter une **liste chaînée à deux niveaux**. On a une liste chaînée, dans laquelle chaque noeud contient une liste chaînée. Ces sous-listes contiennent les éléments à stocker et ont chacune une longueur maximum spécifiée à la construction. Ceci permet de "sauter" des groupes d'éléments pour l'opérateur [ ]. La page suivante illuste un exemple des opérations de push et pop, avec max_taille = 3. Les noeuds carrés sont ceux de la liste principale (type SuperNoeud) et les noeuds ronds ceux des sous-liste (type Noeud).

Votre classe listchain (pour "chaîne de listes") utilise deux types de noeud :

— Un Noeud fait partie d'une liste simplement chaînée. Un objet Noeud contient une valeur val de l'utilisateur, et un pointeur next vers le prochain noeud. Le dernier noeud aura next = nullptr.

— Un SuperNoeud fait partie d'une liste doublement chaînée. Un tel objet contient deux pointeurs premier_noeud et dernier_noeud sur des Noeud, qui sont le début et la fin de la sous-liste du super noeud. Un SuperNoeud contient aussi deux pointeurs prev, next vers les super noeuds suivants ou précédent (prev = nullptr pour le premier super noeud, et next = nullptr pour le dernier super noeud). Une variable nbelem indique combien de noeuds il y a dans la sous-liste du super noeud.

— Votre classe listchain a deux variables membre debut, fin, qui sont des pointeurs vers le premier et dernier super noeud. Ils sont tous les deux à nullptr si la liste est vide. Une variable max_taille spécifie aussi la taille maximum des sous-listes.


max_taille = 3
push_back(7)
pop_back()
push_back(8)
pop_back()
pop_front()
push_front(0)
pop_front()
push_front(1)
debut
fin
prev
premier
dernier
nbelem:2
nbelem:3
nbelem:1
next
val:0
val:1
val:2
val:3
val:4
val:5
val:6
val:7
val:8

Afin de réaliser la liste, Vous devez implémenter les fonctions C++ suivantes :
constructeurs : la liste est vide à la construction. La taille maximum d'une sous-liste peut être spécifiée à la construction, sinon le défaut est à 10.
constructeur par copie : le copieur doit copier les éléments stockés à la source. Par contre, il n'est pas nécessaire de recopier l'état exact de la source passée, c'est-à-dire, la taille des super-noeuds de *this peut différer de la taille des super-noeuds de la source.
operator= : surcharge de l'affectateur, qui copie la liste passée et retourne une référence vers l'objet courant. Comme le constructeur par copie, vous n'avez pas à reproduire l'état exact de la source.
destructeur : nettoie la mémoire allouée dynamiquement.
clear : remet la liste vide, en effaçant tous les éléments.
size : retourne le nombre d'éléments stockés par l'utilisateur.
push_back(val) : ajoute val en fin. On ajoute un noeud au dernier super noeud si son nombre d'éléments est inférieur à max_taille, et sinon on créé un nouveau super noeud.
push_front(val) : ajoute val en début. On ajoute un noeud au premier super noeud si possible, et sinon on en créé un nouveau.
pop_back : retire le dernier élément. Si le dernier super noeud devient vide, il faut le supprimer et mettre à jour la structure. Il est déjà implémenté.
pop_front : retire le premier élément. Si le premier super noeud devient vide, on doit le supprimer et mettre à jour.
operator[](i) : retourne une référence vers le i-ème élément de l'utilisateur. Demande de parcourir les super noeuds adéquatement.

Vous pouvez ajouter des méthodes **privées**, au besoin, mais pas publiques. **N'ajoutez pas de variable membre**.

Un fichier exemple main.cpp vous est aussi fourni, ainsi que la sortie attendue en l'exécutant une fois votre implémentation complétée. Notez que le fait que vous obteniez la sortie attendue n'implique pas une note de 100% : vous êtes responsable de fournir un code libre de bugs, même s'ils ne sont pas identifés par le main.cpp fourni.

Partie 2 : quelques petites questions

Puisque le temps avant l'intra est serré, aucune implémentation n'est demandée pour la partie 2. Vous devez toutefois fournir une réponse textuelle à la question suivante :
On définit n comme le nombre d'éléments stockés dans votre liste, et m comme max_taille, la taille maximum d'une sous-liste. Quelle est la complexité en temps des fonctions push_front, pop_front, push_back, pop_back, operator[ ], size() ?

Vous devez brièvement justifier vos réponses. Une phrase devrait suffire (par exemple, une réponse du style "on a une boucle qui fait x tours, puis on change $O(1)$ variables, donc $O(x)$" est

suffisant). Vous pouvez fournir vos réponses dans l'en-tête de votre fichier .h, ou bien dans un fichier .txt ou .doc.

Évaluation

Comme pour les deux derniers devoirs, vous serez pénalisés pour les erreurs et imprécisions selon les critères suivants : Respect des spécifications (45 points) ; Qualité du code (45 points) ; Respect des normes de programmation (10 points).

Remise du travail

Vous devez remettre les fichiers listchain.h et main.cpp, et possiblement un fichier pour vos réponses à la partie 2. Ne remettez pas d'autre fichier ni surtout d'exécutable. Il ne doit y avoir qu'une seule remise par équipe, à partir d'un seul CIP. Les noms des co-équipiers doivent être clairement indiqués en entête de chaque fichier soumis. Notez qu'il n'y a pas de script de correction automatique - votre code sera inspecté à la correction.