Évaluation Labos #5 et #6 : Structures Linéaires

Noms : Courbat Guillaume – Streckeisen Jarod - Van Hove Timothée

Points : 45 / 50 **Note : 5.5**

Labo 5 : Buffer circulaire de capacité variable 23 / 25

Ajout/suppression d'éléments (push_front, push_back, pop_front et pop_back) **6 / 6**

✔

Accesseurs (front, back et operator[]) et autres méthodes (empty, size et capacity) **3 / 4**

Pour front, utilisez `debut` plutôt que `i_physique(0)` (-1pt)

Pour operator[], vous auriez plus simplement pu écrire : `return buffer[i_physique(i)];`. (pas pénalisé)

Mémoire (constructeur de copie, opérateur d'affectation, destructeur, shrink_to_fit) **9 / 10**

Pour l'opérateur d'affectation, votre ligne `reallocate(dq.capacite);` n'est pas utile. Elle réalloue et déplace les éléments de this dans cette nouvelle mémoire allouée. Ceci est inutile vu que la mémoire va de toute façon être désallouée par la suite. Vous pouvez enlever cette ligne de code. (-1pt)

Factorisation, rendu et propreté **5 / 5**

✔

Labo 6 : Liste simplement chaînée 22 / 25

Ajout/suppression d'éléments (insert_after, erase_after, push_front et pop_front) **6 / 6**

✔

Mémoire (constructeur de copie, opérateur d'affectation, destructeur) **5 / 6**

Pour le constructeur de copie, appelez le constructeur par défaut, cela permettra l'appel au destructeur de l'objet si une insertion d'objet lance une exception, car la liste est déjà créée par ce constructeur par défaut. Avec votre code, le destructeur ne serait pas appelé. (-1pt)

Accesseurs (end, before_begin, front) et autres méthodes (empty, swap) **4 / 4**

✔

Tri de liste et splice_after **2 / 4**

Votre tri tourne indéfiniment chez moi. Utilisez de préférence la fonction next pour retourner un itérateur sur le maillon suivant. Cela fonctionne si on change votre appel à splice_after : `splice_after(j, suivant, next(suivant));` (-1pt)

On demandait que votre tri utilise uniquement les itérateurs et non les maillons. (-1pt)

Par exemple, remplacez : `i.m->suivant != nullptr` par `next(i) != end()`

Votre tri doit aussi être marqué comme noexcept (pas pénalisé)

Factorisation, rendu et propreté **5 / 5**

✔