Université de Sherbrooke
Département d'informatique

IFT339
Structures de données

Hiver 2025

Examen intratrimestriel Type

Professeure:
Aïda Ouangraoua

Date
Durée : 1 h 50

Toute documentation est permise.

Cet examen comporte 3 questions sur 6 pages (+ 2 pages additionnelles).

Question 1:	30 points
Question 2:	30 points
Question 3:	40 points

Total:	100 points

NOM : _____.

PRÉNOM : _____.

CIP : _____.

SIGNATURE : _____.

Question 1 (30 points)

a) Révision sur la gestion de la mémoire

Soit le programme ci-dessous.

```
int max(int i, int j){
      int m(i);
      if(j > m) m = j;
      return m;
      }

int max_vector(vector<int> V){
      int m(V[0]);
      for(int i = 0; i < V.size(); i++){
            if(V[i] > m) m = V[i];
      }
      return m;
      }

int main(){
      int m1, m2, m;
      vector<int> V1;
      vector<int> V2;
      . . .
      m1 = max_vector(V1);
      m2 = max_vector(V2);
      m = max(m1,m2);

      cout<<m<<endl;
      return 0;
}
```

Dessinez toutes les étapes (empilement et dépilement) du cadre de pile pour ce programme suivant le modèle de la pile d'exécution. (15 points)

b) Différence entre portée et durée de vie d'un objet

Expliquez la différence entre la portée et la durée de vie d'un objet dans un programme. (15 points)

Question 2 (30 points)

Considérez la représentation ci-après du **vector**, différente de celle de la bibliothèque standard (SL). Cette représentation comporte trois membres : un pointeur vers le début du tableau dynamique, un `size_t` qui donne la dimension logique, et un `size_t` qui donne la capacité totale du vector.

```
template <typename TYPE>
class vector{
private:
        TYPE* DEBUT;
        size_t DIM;
        size_t CAP;
public:
        ...
        void insert(size_t i, TYPE val);
        void delete(size_t i);
        TYPE max();
        ...
}
```

a) Définissez la fonction **`insert`** du vector (différente de celle de la SL) qui reçoit en paramètres une position `i` dans le vector et une valeur `val`, et qui insère l'élément `val` à la position `i` du vector appelant. Par exemple si le vector appelant `V` contient les éléments `[3,10,5,-1,6]` et que l'on appelle `V.insert(2,3)`, alors les éléments de `V` deviennent `[3,10,3,5,-1,6]`. (10 points)

```
void vector<TYPE>::insert(size_t i, TYPE val){




}
```

b) Définissez la fonction **`delete`** du vector qui reçoit en paramètre une position **`i`** dans le vector et supprime l'élément à cette position du vector. (10 points)

```
void vector<TYPE>::delete(size_t i){

}
```

c) Définissez la fonction **`max`** du vector qui retourne la valeur maximum des éléments du vector, en supposant que les éléments du vector possèdent des opérateurs relationnels (<, >, <=, >=,). (10 points)

```
TYPE vector<TYPE>::max(){

}
```

Question 3 (40 points)

Révision sur le choix de structure de données et la complexité

a) Proposez une représentation pour la classe générique de conteneur ci-dessous permettant d'avoir une opération d'insertion et une opération de suppression en O(n), une opération d'accès à partir de la position en O(1), et une opération de recherche par valeur en O(log(n)). (20pts).

```
template <typename TYPE>
class Conteneur{
private:
     . . .

public:
   ...
   void insert(const TYPE& x);     // insertion en O(n)
   void remove(const TYPE& x);     // suppression de toutes les
                                   // occurrences de l'élément x
                                   // en O(n)
   TYPE& operator[](size_t i);     // accès à l'élément en position i
                                   // en O(1)
   size_t search(const TYPE& x);   // recherche de la position
                                   // de l'élément x en O(log(n))
   size_t size();                  // dimension du conteneur
}
```

b) Définissez la fonction **insert** (20pts)

```
template <typename TYPE>
void Conteneur<TYPE>::insert(const TYPE& x){









}
```

FIN DE L'EXAMEN

Page additionnelle

Page additionnelle