Structures de données en C
TC-IA – S3
2024/2025

TP3

○ Objectifs :

- Concevoir et implémenter des structures de données non linéaires.
- Résoudre des problèmes pratiques nécessitant l'utilisation d'arbres binaires de recherche et de tas.
- Gérer efficacement la mémoire dynamique en C.

○ Contexte : Gestion d'un Système de Priorités

Un gestionnaire de tâches souhaite organiser ses activités à l'aide d'un programme informatique.

Les tâches sont représentées par des identifiants uniques (entiers) et des priorités (entiers positifs).

L'objectif est d'implémenter deux structures de données :

- Un **arbre binaire de recherche** pour stocker les tâches en fonction de leurs identifiants.
- Un **tas binaire (Min-Heap)** pour organiser les tâches selon leurs priorités.

○ Partie 1 : Arbre Binaire de Recherche (BST)

1. Structure :

Implémentez une structure représentant un nœud contenant :

- Identifiant de la tâche (entier).
- Pointeur vers les nœuds gauche et droit.

2. Insertion :

Écrivez une fonction pour ajouter une tâche à l'arbre en respectant les propriétés du BST.

3. Recherche :

Implémentez une fonction pour rechercher une tâche par son identifiant.

Si la tâche existe, affichez son identifiant.

4. Affichage :

Écrivez une fonction pour afficher toutes les tâches dans l'ordre croissant de leurs identifiants (parcours in-order).

5. **Suppression :**

Implémentez une fonction pour supprimer une tâche du BST.

○ Partie 2 : Tas Binaire (Heap)

1. **Structure :**

Implémentez une structure représentant un tas binaire sous forme de tableau dynamique (utilisez malloc pour gérer la mémoire).

2. **Insertion :**

Écrivez une fonction pour insérer une tâche dans le tas en fonction de sa priorité.

3. **Extraction de la tâche de plus haute priorité :**

Implémentez une fonction pour retirer la tâche ayant la priorité la plus élevée (la plus basse valeur).

4. **Affichage :**

Écrivez une fonction pour afficher toutes les priorités dans l'ordre croissant.

5. **Question supplémentaire :**

Implémentez une fonction heapify() pour transformer un tableau de priorités en un tas.

○ Partie 3 : Intégration

Créez un programme principal qui propose les fonctionnalités suivantes :

- Ajouter une tâche dans le BST et dans le tas.
- Rechercher une tâche par identifiant dans le BST.
- Supprimer une tâche du BST.
- Extraire la tâche avec la plus haute priorité du tas.
- Afficher toutes les tâches stockées dans le BST et le tas.

➢ **Consignes :**

- Utilisez des pointeurs pour les structures dynamiques.
- Protégez vos allocations mémoire avec des vérifications (e.g., vérifiez que malloc ne retourne pas NULL).
- Ajoutez des commentaires dans votre code pour expliquer vos choix d'implémentation.

➢ **Livrables :**

- Code source complet avec les tests pour chaque fonctionnalité.