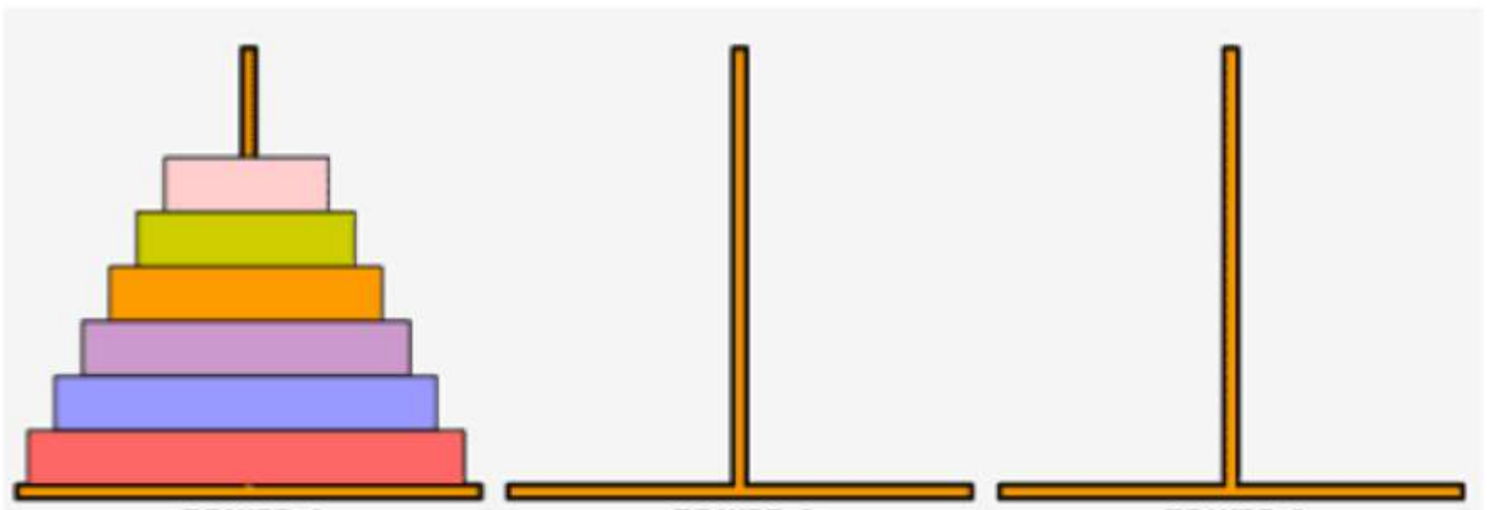
PRÉ-ING2
SEMESTRE 1 - 2024/2025

DS N° 1 INFORMATIQUE III - DOUBLE DIPLOME

Calculatrice et documents non autorisés

Le langage utilisé doit être le langage C obligatoirement
Attention à la robustesse du programme
Vous pouvez définir autant de fonctions que vous le souhaitez dans chaque exercice.
Il n'y a pas nécessairement l'équivalence : une question <=> une fonction.
Attention : Votre code doit être robuste. Vous n'êtes cependant pas obligés de vérifier l'integralité des données des structures vues en CM (définir et utiliser la fonction `verifierFile()` par exemple n'est pas obligatoire).
2 pt bonus vous seront cependant accordés si vous faites correctement toutes les vérifications d'intégrité des données des structures utilisées.
Le barême est donné à titre indicatif mais peut être sujet à changement.

Exercice 1 (Tours de Hanoï) (10,0 pts)



Les tours de Hanoï est un jeu de réflexion consistant à déplacer des disques de diamètres différents d'une tour de «départ» à une tour d'«arrivée» en passant par une ou plusieurs tours «intermédiaires», et ceci en un minimum de coups, tout en respectant les règles suivantes :

- On ne peut déplacer plus d'un disque à la fois
- On ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide

On suppose que cette dernière règle est également respectée dans la configuration de départ illustrée par la figure ci-dessous :

Dans cet exercice, nous représenterons les disques par des entiers dont la valeur représente le diamètre.

1. On considérera que le nombre de disques peut varier d'une partie à l'autre et n'est pas limité.
2. Justifiez en détail l'utilisation d'une `Pile dynamique` et justifiez également le choix d'une `Pile statique` pour représenter au mieux le fonctionnement d'une tour de Hanoi.
   Quelle que soit votre réponse, dans la suite de l'exercice nous utiliserons une `Pile dynamique`.
3. Définir la structure Tour qui devra, en plus des champs habituels, inclure un champ taille représentant le nombre de disques présents dans une `Tour`.
4. Ecrire une fonction enleverDisque(...) permettant de retirer un disque d'une Tour dont l'adresse sera passée en paramètre. Cette fonction doit vérifier le bon fonctionnement de la `Tour` et des règles du jeu. Le reste du programme doit pouvoir avoir accès à la valeur du diamètre du disque retiré.
5. Ecrire une fonction ajouterDisque(...) permettant d'ajouter à une Tour (dont l'adresse sera passée en paramètre) un disque de diamètre `d` (également passé en paramètre). Cette fonction doit vérifier le bon fonctionnement de la Tour et des règles du jeu.
6. Ecrire une procédure permettant d'afficher le contenu d'une Tour (les valeurs des diamètres de ses disques) verticalement avec le disque le plus petit au-dessus, comme on pourrait le visualiser sur une vraie tour.

Exemple possible :

```
2
3
8
13
87
```

7. Ecriture du programme principal :
   - Déclarer 3 `Tour` a, b, c qui seront initialement vides.
   - Demander à l'utilisateur le nombre de disques du jeu, sachant que ce nombre doit être compris entre 3 et 10.
   - Les disques de jeu, représentés par des entiers, auront des valeurs entre 1 et le nombre de disque choisis. Remplir la `Tour` a avec tous les disques du jeu en respectant les règles.
   - Le jeu enchaine donc les cycles de jeu en respectant les règles. Un cycle de jeu se déroule comme suit :
     - On demande à l'utilisateur sur quelle `Tour` il souhaite retirer un disque puis sur quel `Tour` il souhaite le replacer.
     - Si cela est possible le déplacement du disque est effectué
     - On affiche le contenu des trois `Tour` de gauche à droite (a puis b puis c)
   - Le jeu s'arrête lorsque tous les disques sont sur la `Tour` c (celle qui est le plus à droite)
   - Ecrire le programme permettant donc de jouer au jeu et de s'arrêter en cas de victoire.
8. **[BONUS] (+2,0 pts) Réécrire le programme principal pour que le jeu ne se fasse plus sur 3 Tour mais sur un nombre** de `Tour` dynamique entre 3 et 20 qui sera choisi par l'utilisateur. On utilisera alors un tableau de `Tour` pour gérer le jeu. Le nombre de disques quant à lui pourra être compris entre 3 et 500.

Exercice 2 (*Structure d'une entreprise*) **(10,0 pts)**

Dans cet exercice, nous allons construire un organigramme d'une entreprise que sera sous forme d'abre. Chaque hauteur de l'arbre representera un niveau hierarchique de l'entreprise. Les noeuds seront des membres de l'entreprise. Les *enfants* d'un noeud seront les employés sous la responsabilité directe de cet employé. La racine de l'arbre sera donc le chef d'entreprise.

1. Définir une structure `Employe` qui représentera un membre de l'entreprise. Cette structure doit contenir :
   - Le nom de l'employé
   - Son salaire
   - Son année d'arrivée dans l'entreprise.
2. Pour des raisons pratiques, on considérera que chaque membre de l'entreprise possède au maximum **trois** employés sous sa responsabilité. Définir la structure `Arbre` permettant de construire l'arbre de hiérarchie de l'entreprise.
3. Ecrire une procédure `afficherSubordonnes(...)` qui prend en paramètre un Noeud de l'arbre et afffiche le nom de tous les employés sous sa responsabilité directe.
4. Ecrire une procédure `AfficherN2(...)` qui va va prendre en paramètre un noeud de l'arbre et afficher tous les employés qui sont deux niveaux en dessous de lui.
5. Ecrire une fonction `niveauEntreprise(...)` qui va calculer et retourner le nombre de niveaux hiérarchiques de l'entreprise passée en paramètre.
6. Ecrire une fonction `rechercheChef` qui va prendre en paramètre
   - Un nom
   - un Arbre d'employé représentant une entreprise.

   Cette fonction va rechercher si un employé de ce nom existe et va retourner **l'adresse du noeud de cet employé.** Si l'employé n'existe pas la fonction devra retourner `NULL`.
7. On souhaite afficher les membres de l'intégralité de l'entreprise en affichant dans l'ordre hiérarchique : d'abord le chef puis les employés directement sous-lui, pour finir avec les employés les plus bas dans la hierarchie.
   (a) Quel type de parcours permet d'afficher les employés dans l'ordre souhaité ?
   (b) Définir la ou les structure qui permettront de gérer une file d'employés.
   (c) On suppose que la fonction `enfiler` pour ce type de file existe. Ecrire la fonction `défiler`.
   (d) Ecrire la ou les fonctions/procédures permettant d'effectuer le parcours de l'arbre désiré.