TD 03 : Piles et Files (2)

Consignes générales : N'oubliez pas pour ce TD comme pour les suivants de vous créer un répertoire consacré au TD et d'enregistrer vos codes dedans.

On rappelle que les commandes à taper dans le terminal pour compiler puis éxécuter votre programme C :
— Pour compiler : *gcc -o nom_executable nom_programme.c*
— Pour exécuter : *./nom_executable*

Exercice 1 (*Liste doublement chaînée*)
On rapelle que les `Chainons` des listes doublement chaînées possèdent un pointeur vers le `Chainon` suivant et un pointeur vers le `Chainon` précédent. Il est donc possible de parcourir une telle liste dans les deux sens.

1. Déclarer une structure de liste doublement chaînée permettant de stocker des mots de 20 caractères maximum.
2. Écrire une fonction `compareMot(char *mot1, char* mot2 )` qui retourne 1 si `mot1` est avant `mot2` dans l'ordre alphabétique et 0 sinon.
3. Écrire une fonction `insertListe(char *mot1, liste *pliste )` qui ajoute à la liste doublement chaînée pointée par `pliste` de mots classés dans l'ordre alphabetique un nouveau `Chainon` contenant `mot1`.
4. Écrire une procédure `affiche(liste *pliste )` qui affiche dans l'ordre les mots contenus dans la liste.
5. Écrire une procédure `afficheInv(liste *pliste )` qui affiche dans l'ordre inverse les mots contenus dans la liste.
6. Déclarer un nouvelle liste et la remplir avec une vingtaine de mots. Afficher cette liste de mots dans l'ordre alphabétique et anti-alphabétique.

Exercice 2 (*Tri de crèpes(extrait du rattrapage de 2021-2022)*)
On souhaite écrire un algorithme qui va pouvoir modéliser le comportement d'un tas de crêpes qui sera trié par un humain (par ordre croissant de diamètre, la plus petite sur le dessus, la plus grande tout en bas du tas).

L'humain ne dispose que d'un seul outil pour trier ses crêpes, c'est une spatule qu'il peut insérer entre 2 crêpes (sans les abîmer), et retourner le tas au-dessus de la spatule. Cette opération inverse donc bien la partie supérieure du tas uniquement.
Exemple de tri avec une spatule

```
  3
  6
  2    ____
  9 __/
  7
  4
 \_5_/
```

En plaçant la spatule entre la crêpe de diamètre 9 (la plus grande) et la crêpe de diamètre 7, on peut inverser l'ordre des crêpes placées **au-dessus** de la spatule.
La crêpe de diamètre le plus grand (9) se retrouve donc au-dessus de la pile comme indiqué ci-dessous :

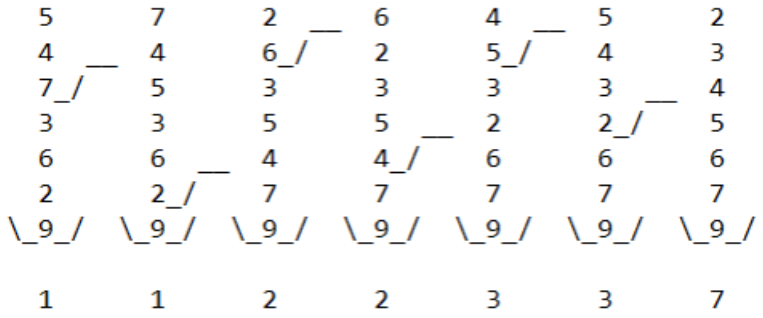
```
  9
  2
  6
  3
  7
  4
 \_5_/
```

Il faut ensuite placer la crêpe 9 tout en bas : pour cela, on réalise la même opération pour tout le tas de crêpes (en plaçant la spatule sous la crêpe de diamètre 5 ici dans l'exemple).
Ceci fait nous avons donc, en 2 coups de spatule, la crêpe la plus grande tout en bas, comme indiqué ci-dessous :

```
  5
  4
  7
  3
  6
  2
\_9_/
```

Il ne reste plus qu'à vérifier si le tas de crêpe est correctement trié. Si ce n'est pas le cas, il faut recommencer avec les N-1 crêpes supérieures (car la crêpe de diamètre 9 n'a plus besoin d'être touchée, elle est déjà à la bonne place), sinon le tri est terminé.

Voici les étapes restantes pour trier correctement le tas, avec le nb de crêpes correctement triées. Voici la liste des différentes étapes suivantes pour trier le tas de crêpes précédent.



Dans notre algorithme, on définira notre tas de crêpe par une pile d'entiers.

1. Définir une structure `Crepe` qui contient un entier (le diamètre de la crêpes) et un pointeur vers la crêpe suivante : cette structure nous permet donc d'avoir une liste chainée de crêpes. Définir un également un nouveau type `Pcrepe` pointeur sur `Crepe`.
2. Créer une fonction `Pcrepe inserFile(Pcrepre tete, Crepe c)` qui simule l'insertion d'une crèpe dans une **FILE** dont la tête est pointée par `tete`.
3. Créer une fonction `Pcrepe suppFile(Pcrepre tete)` qui simule le retrait d'une crêpe dans une **FILE** dont la tête est pointée par `tete`.
4. Créer une fonction `Pcrepe inserPile(Pcrepre tete)` qui simule l'insertion d'une crêpe dans une **PILE** dont la tête est pointée par `tete`.
5. Créer une fonction `Pcrepe suppPile(Pcrepre tete)` qui simule le retrait d'une crêpe dans une **PILE** dont le premier élément est pointé par `tete`.
6. Créer une fonction `int triCrepe(Pcrepre tete)` qui retourne 1 si la PILE de crêpes dont la tête est indiquée par tete est triée par ordre croissant, 0 sinon.
7. Créer une fonction `Pcrepe invCrepe(Pcrepre tet, int M)` qui va inverser les M premiers éléments d'une **PILE** (on pourra utiliser une FILE temporaire pour cela) et qui retourne la nouvelle tête de la pile.
8. Créer une fonction `int indMax(Pcrepe tete)` qui retourne l'indice de l'élément le plus grand d'une **PILE** de crêpe (- 1 si la pile est vide).
9. Créer une fonction `Pcrepe spatule(Pcrepre tete, int M)` qui va rechercher la crêpe la plus grande parmi les M premiers éléments d'une **PILE** et qui va inverser ces M éléments. La fonction retourne le pointeur sur la tête de la PILE modifiée.
10. Créer une fonction/procédure qui va utiliser les fonctions précédentes afin de réaliser l'objectif demandé, c'est à dire qu'à partir d'une PILE de crêpes quelconque, on souhaite, une à une dans l'ordre, placer les crêpes les plus grandes tout en bas de la pile.