Cours/TD — Pile (suite) :

1. Fonctionnement du programme `infix2postfix` du précédent support ;
2. Écrire une structure (la bibliothèque) de pile stockant des `int` et dont la taille est dynamique ;
3. Donner la possibilité de gérer plusieurs piles en parallèle ;
4. Réfléchir à la généricité (quelque soit le type de donnée empilé) de la bibliothèque et écrire les prototypes (faire la conception).
5. Exemple d'usage d'une pile générique ;
6. Écriture de la pile générique ;

Cours/TD — Listes chaînées :

1. Gestion des rationnels avec tableau dynamique (doublement de la mémoire allouée si besoin), pile et garbage collector ;
2. Liste chaînée simple : structure, parcours, insertion et destruction ;
3. Liste doublement chaînée et circulaire (problème de Joséphus) ;
4. Piles multiples en utilisant des listes ...

TP — Listes chaînées :

1. Écrire une structure de données `node_t` utile au stockage d'un nombre indéfini de couples noms — prénoms ;
2. Écrire la fonction `node_t * new_node(char * first_name, char * last_name)` qui crée un élément nom-prénom ;
3. Écrire la fonction `void add_node(node_t ** here, node_t * n)` qui ajoute l'élément `n` dans `here` ;
4. En ayant un pointeur vers le début d'une liste de noms et prénoms :
   (a) Comment ajouter un nouvel élément en début de liste ? L'écrire ;
   (b) Comment ajouter un nouvel élément en fin de liste ? L'écrire ;
   (c) Comment ajouter un nouvel élément en respectant un ordre pré-établi (liste déjà triée dans l'ordre croissant ou décroissant) ? L'écrire ;
   (d) Comment supprimer un élément de la liste ? L'écrire.
   (e) Comment libérer la liste ? L'écrire.
5. Et un modèle générique ? L'écrire ? Autres types de listes chaînées ...

Devoir 02 :

— En utilisant la fonction `infix2postfix` et une structure de Pile (support $n^o 2$), réaliser le programme *miniCalculatrice* (votre archive miniCalculatrice.tgz contient uniquement vos sources (.c et .h) et un Makefile ; ce dernier commence avec 3 lignes de commentaires indiquant respectivement **NOM : <nom et prénom>, NUMERO : <numéro d'étudiant>, EMAIL : <email>**) capable de calculer la valeur d'une expression infixée et parenthésée dans laquelle nous utilisons des opérateurs binaires (deux arguments) tels que "+, -, *, /" et des entiers naturels..

— Utiliser la bibliothèque `ratio` (modifier le support $n^o 1$ ou utiliser le code "pile de ratio" qui sera disponible en ligne) afin d'obtenir au préalable un résultat sous la forme d'une fraction rationnelle. Voici comment doît se dérouler une exécution :

— CAS 1 :

```
bash$ ./miniCalculatrice
(((1+2)*3)+4*(((5+6)/7)+8))
1 2 + 3 * 4 5 6 + 7 / 8 + * +
(331 / 7) = 47.286
```

— CAS 2 :

```
bash$ echo "(((1+2)*3)+4*(((5+6)/7)+8))" | ./miniCalculatrice
1 2 + 3 * 4 5 6 + 7 / 8 + * +
(331 / 7) = 47.286
bash$
```

La procédure de remise du devoir 02 sera disponible en ligne (sur la page du cours) du 11 au 22 octobre 2018.