TD 01 : Listes chaînées

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 (*Listes des puissances de deux*)

1. Déclarer une structure `Chainon` permettant de construire une liste chaînée comportant des entiers.
2. Écrire une fonction `Chainon * creationChainon(int a)` qui retourne un pointeur vers un nouveau Chainon contenant l'entier `a`.
3. Écrire une fonction `Chainon * insertDebut(Chainon* pliste, int a)` permettant d'insérer un nouvel entier `a` en début de la liste chaînée pointée par `pliste`. Cette fonction retourne la tête de la liste chaînée.
4. Écrire une fonction `insertFin(Chainon* pliste, int a)` permettant d'insérer un nouvel entier `a` en fin de la liste chaînée pointée par `pliste`.
5. Écrire une procédure `afficheListe(Chainon* pliste)` permettant d'afficher les différents éléments de la liste chaînée pointée par `pliste`. Les entier seront séparés par le symbole "->".
   Ex : 1 -> 10 -> 3 -> 15
6. Dans le programme principal, initialiser un pointeur sur une liste vide.
7. On souhaite que ce programme affiche une liste des puissances de deux dans l'ordre. La taille de cette liste sera décidée par l'utilisateur ; on affiche la liste chaînée et on demande à l'utilisateur s'il souhaite voir la puissance de deux suivante, si c'est le cas, on rajoute un Chainon supplémentaire à la liste et on l'affiche à nouveau, sinon on arrête le programme.

Exercice 2 (*Liste croissante*)

Écrire un programme qui ajoute un élément dans une liste simplement chaînée et triée par ordre croissant.

Exercice 3 (*Suppression de Chaînon*)

1. Définir une liste chaînée d'entiers composée de 10 `Chainons` dont les valeurs seront aléatoires entre 0 et 5 compris.
2. Écrire des fonctions prenant en paramètre un pointeur sur liste et un entier qui permettent de supprimer dans une liste :
   — Le premier `Chainon` dont la valeur est égale au paramètre.
   — Tous les `Chainon` dont la valeur est égale au paramètre.
3. Tester ces fonctions avec la liste de la question 1.

Exercice 4 (*Inversion d'une liste*)

— Écrire un programme qui permet de créer une liste chaînée inverse d'une autre liste (premier élément en dernier, deuxième en avant dernier etc.).
— Écrire un programme qui permet d'inverser une liste chaînée sans passer par une liste intermédaire.

Exercice 5 (*Gestion des étudiants (extrait du devoir de 2021/2022)*) On souhaite gérer les notes et les informations de l'ensemble des étudiants de CY–Tech.

1. Un étudiant de deuxième année est défini par son nom, son prénom, son groupe et ses notes. Plus précisément, chaque étudiant reçoit toujours 10 (constante) notes. Pour plus de facilité, le groupe de l'étudiant sera défini par un entier correspondant au numéro de son groupe (exemple : groupe 1, groupe 6 etc...). Définir la structure Etudiant permettant de stocker ces informations.
2. Comme on peut avoir des retards à l'inscription ou des étudiants qui partent en cours d'année, les étudiants vont être rangés dans une liste dynamique. Définir la ou les structures nécessaires pour construire la liste dynamique `LstEtudiants` des étudiants.
3. Écrire une fonction `saisirEtudiant(LstEtudiants lst)` qui permet de saisir les données d'un nouvel étudiant et de l'ajouter à la fin de la liste. Les notes seront aléatoires.
4. Proposer une procédure `listeParGroupe(LstEtudiants lst, int groupe)` permettant d'afficher le nom de tous les étudiants du groupe passé en paramètre.

5. Écrire une fonction `moyTab( int *tab)` qui prend en argument un tableau statique de taille N et qui renvoie la moyenne de ses éléments.
6. Écrire une fonction `trouveEtudiant( char *nom, char * prenom, LstEtudiants lst)` qui recherche un étudiant par son nom et son prénom et renvoie un pointeur sur l'étudiant ou NULL s'il n'existe pas.
7. Proposer un algorithme permettant de calculer la moyenne de l'élève dont le nom est «Spiruline Barnabus». Affichez un message d'erreur si aucun élève de ce nom n'existe.
8. Calculer la moyenne de toute la promotion.
9. Afficher le nom de l'Etudiant ayant la plus mauvaise moyenne de la promotion.