Les listes chaînées

Solution de l'exercice :

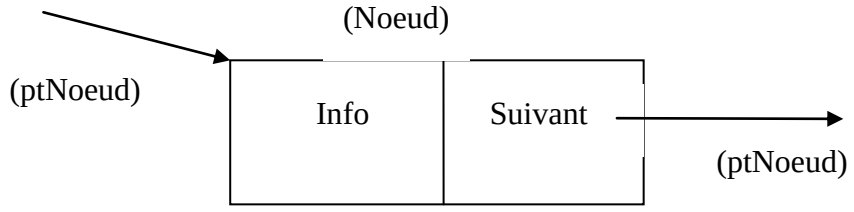
Soit les déclarations de types et la procédure XXXX suivantes :

```
Type ptNoeud = ^ Noeud
Type Noeud = Structure
          Info : Entier ;
          Suivant : ptNoeud
        finstructure

Procédure XXXX ( E/S tete : ptNoeud, E x : Entier)
    Déclaration p, p1 : ptNoeud
    Début
    p ← tete
    si (p=null) alors
         début
             tete ← allouer (Noeud)
             tete^.info ← x
             tete^.suivant ← null
         fin
    sinon
         début
             tant que (p^.suivant <> null) faire
                  p← p^.suivant
             fin tant que
             p1 ← allouer (Noeud)
             p1^.info ← x
             p1^.suivant ← null
             p^.suivant ← p1
         fin
    finsi
Fin Procédure
```

Question 1 *:*

Réalisez un schéma de la structure déclarée avant la procédure.



Question 2 *:*

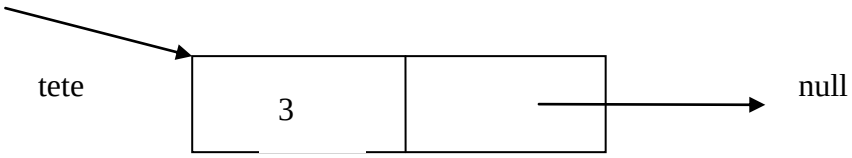
Réaliser un schéma de l'application de la procédure après chaque appel de procédure selon les instructions qui suivent :

Vous justifierez votre schéma (déroulement du code de la procédure « à la main »)

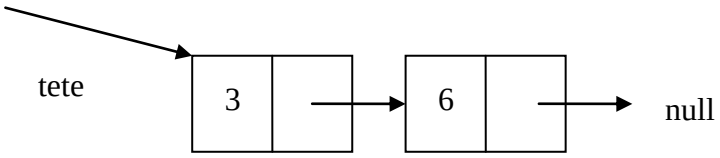
1. *q ← null*
2. *XXXX (q, 3)*

3. *XXXX (q, 6)*
4. *XXXX (q, 1)*
5. *XXXX (q, 2)*

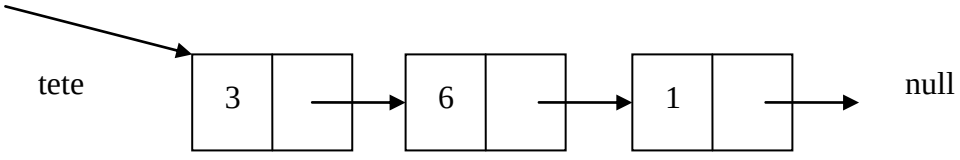
1. q est affecté à null ;
2. L'appel de la procédure XXXX se fait avec q (soit tete) qui vaut null et x qui vaut 3 => on affecte à p la valeur null et on entre donc dans le premier bloc du si (on crée la structure et on y entre le nombre 3 et on pointe le suivant sur nil, tete qui était à nil pointe désormais sur la tete de liste):



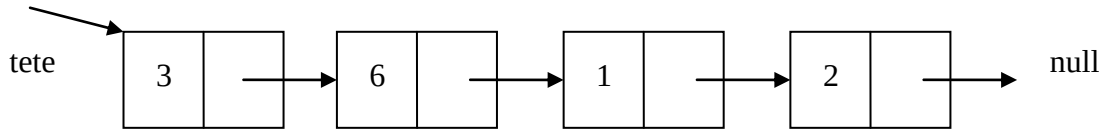
3. L'appel de la procédure XXXX se fait avec la tete de liste et x qui vaut 6 => on affecte à p la tete de liste et on entre donc dans le deuxième bloc du si, le tant que s'arrête immédiatement (suivant pointe sur null) => on crée la structure p1, on y entre le nombre 6 et on pointe le suivant sur null, puis p^.suivant pointe désormais sur p1 qu'on vient d'ajouter :



4. L'appel de la procédure XXXX se fait avec la tete de liste et x qui vaut 1 => on affecte à p la tete de liste et on entre donc dans le deuxième bloc du si, le tant que s'arrête après avoir avancé sur la deuxième cellule (dont suivant pointe sur null) => on crée la structure p1 et on y entre le nombre 1 et on pointe le suivant sur null, puis p^.suivant pointe désormais sur p1 qu'on vient d'ajouter:



5. L'appel de la procédure XXXX se fait avec la tete de liste et x qui vaut 2 => on affecte à p la tete de liste et on entre donc dans le deuxième bloc du si, le tant que s'arrête après avoir avancé sur la troisième cellule (dont suivant pointe sur null) => on crée la structure p1 et on y entre le nombre 2 et on pointe le suivant sur null, puis p^.suivant pointe désormais sur p1 qu'on vient d'ajouter:



Question 3 *:*
Que fait cette procédure ? Comment pourrait-on la renommer pour qu'elle soit plus explicite ?
Cette procédure ajoute un élément à la fin d'une liste d'entiers.
On pourrait la renommer Procédure ajouter (E/S tete : ptNoeud, E x : entier)).

Question 4 *:*
Documenter cette procédure :

```
Type ptNoeud = ^ Nœud       // déclaration du pointeur
Type Noeud = Structure      // déclaration du type de nœud contenu dans la liste
          Info : Entier ;
          Suivant : ptNoeud
        finstructure

Procédure XXXX ( E/S tete : ptNoeud, E x : Entier)
// procédure qui ajoute un élément en fin de la liste pointée par tete
// un paramètre en Entrée/Sortie tete qui est un pointeur
// un paramètre en entrée x de type entier qui sera la valur placée dans le nouveau noeud
    Déclaration p, p1 : ptNoeud
    Début
    p ← tete // p sera notre pointeur d'avancement dans la liste initialisé sur tête
    si (p=null) alors
        début
            tete ← allouer (Noeud)
            tete^.info ← x              // on crée l'élément de tête, on y met la valeur x et on
            tete^.suivant ← null        // pointe le  suivant  sur null
        fin
    sinon  // la liste n'est pas vide
        début
            tant que (p^.suivant <> null) faire
                 p← p^.suivant //on avance jusqu'au dernier élément (qui pointe sur null)
            fin tant que
            p1 ← allouer (Noeud)       // on crée le nœud p1,
            p1^.info ← x               // on y met la valeur x
            p1^.suivant ← null         // et on pointe son suivant sur null
            p^.suivant ← p1            // le dernier nœud parcouru pointe sur ce nouvel élément
        fin
    finsi
Fin Procédure
```