# Algorithmique 1

## Interrogation 2 groupe CSB3A2

| NOM : | Prénom |
|---|---|

## Exercice 1

On considère les fonctions :

```
fonction f(val n, m :
entier) :entier ;
  debut
  si n == 0 alors
    retourner m ;
  sinon
    retourner f(n-1, m)+1 ;
  finsi
  fin
```

```
fonction g(val n, m :
entier) :entier ;
  debut
  si n == 0 alors
    retourner m ;
  sinon
    retourner g(n-1, m+1) ;
  finsi
  fin
```

Que calculent f et g ? Faire tourner les deux fonctions pour n = 4, m = 5.
Peut-on parler de récursivité terminale pour ces fonctions ?

## Exercice 2

On considère dans cet exercice des listes simplement chaînées d'entiers.
On considère une liste simplement chaînée non vide `L` ; on appelle `x` la valeur du premier élément de la liste. Ecrire une fonction qui crée une liste simplement chaînée `LS` en mettant en début de liste les éléments de `L` inférieurs ou égaux à `x` dans le même ordre que dans `L`, puis l'élément `x`, puis les éléments strictement supérieurs à `x` dans le même ordre que dans `L`.
Par exemple si `L={5, 7, 2, 3, 10, 4, 1, 8}`
alors `LS={2, 3, 4, 1, 5, 7, 10, 8}`

On utilisera les primitives du type abstrait Liste Simplement Chaînée d'entiers et, au choix, une pile ou une file d'entiers.

## Exercice 3

On implémente une file d'entiers avec une liste doublement chaînée d'entiers.
Ecrire les primitives `valfile`, `enfiler` et `défiler` à l'aide des primitives du type abstrait liste doublement chaînées d'entiers.

## Exercice 4

On considère dans cet exercice les listes doublement chaînées d'entiers implémentées par allocation dynamique : `listeDC-ent`

Ecrire une fonction `supprimerToutesOccurence` qui supprimer toutes les occurrences de l'entier `x` dans la liste `L`. Exemple :
Si la liste `L` est : 5, 5, 5, 5, 9, 4, 5, 13, 4, 9, 5 et si `x` = 5 la liste `L` devient 9, 4, 13, 4, 9.
On utilisera les primitives données en annexe.

## Annexe

## Liste simplement chaînée d'entiers

```
cellule = structure
          valElem : entier ;
          pointSuiv : ^cellule ;
          finstructure

curseur = ^cellule

listeSC_ent : structure
              premier : curseur ;
              cle : curseur ;
              finstructure

fonction creerListe(ref L : liste_SC_ent) : vide
fonction debutListe(ref L : liste_SC_ent) : vide
fonction valeur(val L : liste_SC_ent) : entier
fonction suivant(ref L : liste_SC_ent) : vide
fonction listeVide(val L : liste_SC_ent) : booleen
fonction insererEnTete(ref L : liste_SC_ent ; val x :entier) : vide
fonction insererApres(ref L : liste_SC_ent ; val x :entier) : vide
fonction supprimerEnTete(ref L : liste_SC_ent) : vide
fonction supprimerApres(ref L : liste_SC_ent) : vide
fonction estFinListe(val L : liste_SC_ent) : booleen
fonction estDernier(val L :listeSC_ent) :booleen
```

## Liste doublement chaînée d'entiers

```
cellule = structure
          valElem : entier ;
          pointPrec : ^cellule ;
          pointSuiv : ^cellule ;
          finstructure

curseur = ^cellule

listeDC_ent : structure
              premier : curseur ;
              dernier : curseur ;
              cle : curseur ;
              finstructure

fonction creerListe(ref L : liste_DC_ent) : vide
fonction debutListe(ref L : liste_DC_ent) : vide
fonction finListe(ref L : liste_DC_ent) : vide
fonction valeur(val L : liste_DC_ent) : entier
fonction suivant(ref L : liste_DC_ent) : vide
fonction precedent(ref L : liste_DC_ent) : vide
fonction listeVide(val L : liste_DC_ent) : booleen
fonction insererEnTete(ref L : liste_DC_ent ; val x :entier) : vide
fonction insererApres(ref L : liste_DC_ent ; val x :entier) : vide
fonction supprimerEnTete(ref L : liste_DC_ent) : vide
fonction supprimerApres(ref L : liste_DC_ent) : vide
fonction estFinListe(val L : liste_DC_ent) : booleen
```