Université Bordeaux 1 Licence Informatique 2008/2009

Algorithmique 1

Interrogation 1 groupe CSB3A4

NOM :	Prénom

Exercice 1

Ecrire une fonction récursive terminale qui calcule a^n pour a réel et n entier positif.

Exercice 2

Ecrire une fonction qui retourne le maximum d'une liste d'entier. On utilisera une liste simplement chaînée d'entiers et les primitives données en annexe.

Exercice 3

On considère ici les listes doublement chaînées d'entiers implémentées par allocation dynamique.

3.1 Ecrire les primitives `insererAvant` et `supprimerAvant` sans utiliser `insererApres` et `supprimerApres` (voir annexe)

`insererAvant`

`supprimerAvant`

3.2 Ecrire une fonction `concatene(ref L1, L2 liste_DC_ent) : vide`
qui modifie L1 et L2 de sorte que les éléments de L2 sont ajoutés à la fin de la liste L1, et L2 devient une liste vide. On utilisera l'implémentation donnée en annexe et on ne créera aucune nouvelle cellule. Exemple :
L1 contenant 12, 6, 8, 17, 3 et L2 contenant 5, 9, 11 devient L1 contenant 12, 6, 8, 17, 3, 5, 9, 11 et L2 vide.

3.3 On suppose la liste L triée dans l'ordre croissant. Ecrire une fonction `insereListeTriee` qui insère l'entier `x` dans la liste `L` de sorte que la liste reste triée. On utilisera uniquement 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
```

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
```