Université Bordeaux 1 Licence Informatique 2009/2010

Algorithmique 1

Interrogation 1 groupe CSB3A4

NOM :	Prénom

Exercice 1

Ecrire 3 versions d'une fonction qui calcule x^n pour x réel et n entier positif : une version itérative, une version récursive non terminale et une version récursive terminale.

Itérative	*Récursive non terminale*
	Récursive terminale

Exercice 2

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

2.1 Ecrire une fonction qui retourne la moyenne d'une liste d'entier `L`.
On utilisera les primitives données en annexe.

2.2 Ecrire une fonction `insereRang` qui insère l'entier `x` dans la liste `L` de sorte qu'il occupe le rang `k` dans la liste. On supposera la valeur de `k` cohérente avec la liste ($1 \leq k \leq$ longueur(`L`)+1).

2.3 On suppose la liste `L` sans doublon et triée dans l'ordre croissant.
Ecrire une fonction `placeDansListeTriee` qui retourne le rang qu'occuperait l'entier `x` si on l'insérait dans la liste `L` de sorte qu'elle reste triée et retourne 0 si `x` est déjà dans la liste.
Exemple :
Si `L` est la liste 5, 9, 12, 25, 27 la fonction renvoie 1 si `x` = 3, 4 si `x` = 20, 6 si `x` = 30 et 0 si `x` = 12
On utilisera uniquement les primitives données en annexe.

Exercice 3

On considère dans cet exercice les listes doublement chaînées d'entiers implémentées par allocation dynamique.

3.1 Ecrire la primitive `insererEnFin` sans utiliser `insererApres` (voir annexe)

3.2 Ecrire une fonction `supprimerToutOccurence` 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
```