Examen – Session principale

Matière : Atelier programmation II Enseignants : Majdi JRIBI et Dorsaf SEBAI Filière : MPI Nombre de pages : 7 pages	Semestre: Second semestre Date: 17 Mai 2018 Durée: 1h30 Documents : ***non autorisés***

Les réponses doivent être rédigées obligatoirement sur les feuilles de réponse (pages 6 et 7)

L'examen contient 7 pages. Seulement les pages 6 et 7 sont à rendre.

Exercice 1 : Arbre binaire de recherche

Considérons la déclaration suivante relativement à un nœud ***d'un arbre binaire de recherche***:

```
typedef struct noeud
{
char info[32] ;
struct noeud* gauche;
struct noeud * droit;
} NOEUD ;
```

Le champ de données est une chaine de caractères.
Déterminer le rôle de chacune des fonctions suivantes (Répondre sur les feuilles de réponses).

Question 1

```
NOEUD* traitement_1(NOEUD *R,
NOEUD *N)
{ if (R == NULL) return N;
if (strcmp(N→info,R→info)<0)
R→gauche = traitement_1(R→gauche,N);
else
if (strcmp(N→info,R→info)>0)
R→droit = traitement_1(R→droit, N);
else
{
free(N);
}
return R;
}
```

Question 2

```
void traitement_2(NOEUD *R)
{
if (R != NULL)
{
traitement_2(R→gauche);
traitement_2(R→droit);
free(R);
}}
```

Question 3

```
int traitement_3(NOEUD *A, char*
mot)
{int icmp=0;
if (A==NULL)
{ return -1; }
icmp= strcmp(mot,A→info);
if (icmp==0)
{ return 0; }
if (icmp<0)
return traitement_3(A→gauche,mot);
if (icmp>0)
return traitement_3(A→droit, mot);
}
```


CamScanner

Question 4

```
NOEUD *traitement_4(NOEUD *A,
NOEUD *Ins)
{
if(A== NULL)
return Ins;
A→gauche=traitement_4(A→gauche, Ins);
return A;
}
```

Question 5

```
NOEUD *traitement_5(NOEUD *A,
char * motS)
{ int icmp=0; NOEUD *S=NULL;
if(A==NULL)
{ return NULL;}
icmp= strcmp(motS,A→info);
if (icmp==0)
{ S=A;
A=traitement_4(A→droit,A→gauche);
free(S); return A; }
if (icmp<0)
{ A→gauche=traitement_5(A→gauche,
motS );
return A; }
if (icmp>0)
{
A→droit=traitement_5(A→droit,motS);
return A; }
}
```

CamScanner

Exercice 2

On se propose de développer un ensemble de modules permettant de manipuler des polynômes creux. Un polynôme creux est un polynôme contenant très peu de monômes non nuls.

Exemple : $\mathbf{P(x) = 5.6\ x^{1280} + 0.8\ x - 9}$ contient **1281** termes dont **3** seulement sont non nuls.

Un monôme est defini par la structure suivante:

```
typedef struct sMonome {
  int deg;
  double coef;
  struct sMonome *suiv;
} TMonome;
```

Chaque monôme est décrit par une structure de données de type ***TMonome*** comportant les 3 champs suivants :

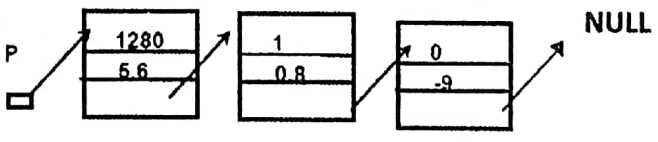
- *deg* : entier représentant le degré du monôme ;
- *coef* : réel représentant le coefficient du monôme ;
- *suiv* : pointeur sur le monôme suivant.

Un polynome est défini par la structure de données suivante :

```
typedef struct sPolynome {
  TMonome *prem;
} TPolynome;
```

Un polynôme sera décrit par une structure ***TPolynome*** contenant un seul champ ***prem*** :

- Si *prem* est NULL, le polynôme correspondant sera le polynôme zéro ;
- Si *prem* est non NULL, il pointera sur le monôme de plus haut degré dont le coefficient est non nul. Les monômes de coefficient non nuls sont chaînés par ordre de degré **décroissant.**



Une illustration du polynôme P

Donner le rôle de chacune des fonctions suivantes (sur la feuille de réponse).

CamScanner

Question 1

```
void fonction_1(TPolynome *pp) {
  pp→prem = NULL;
}
```

Question 2

```
void fonction_2(TPolynome *pp) {
  TMonome *cur = pp→prem , *next;
  while (cur != NULL) {
    next = cur→suiv;
    free(cur);
    cur = next;
  }
  pp→prem = NULL;
}
```

Question 3

```
void fonction_3(TPolynome p) {
  TMonome *cur;
  cur = p.prem;
  while (cur != NULL) {
    if (cur→deg == 0)
      printf("%f", cur→coef);
    else if (cur→deg == 1)
      printf("%f X", cur→coef);
    else printf("%-.2f X ** %d", cur→coef,
cur→deg);
    cur = cur->suiv;
    if (cur != NULL) printf(" + ");
  }
  printf("\n");
}
```

Question 4

```
void fonction_4(TPolynome *p, TPolynome
polynome) {
  TMonome *cur, *d, *pred;
  int prem = 1;
  cur = polynome.prem;
  while (cur != NULL) {
    if (cur→deg > 0) {
  d = (TMonome *) malloc(sizeof(TMonome));
      if (d == NULL) {
        printf("Erreur allocation memoire\n");
            exit(1);
      }
      d→coef = cur→coef * cur→deg;
      d→deg = cur→deg - 1;

      if (prem) {
            prem = 0;
            p→prem = d;
      }
      else pred→suiv = d;
      pred = d;
    }
    cur = cur→suiv;
  }
}
```

Question 5

```
void fonction_5(TPolynome *p, TPolynome
polynome) {
  TMonome *cur, *d, *pred;
  int prem = 1;
  cur = polynome.prem;
  while (cur != NULL) {
    d = (TMonome *)
malloc(sizeof(TMonome));
    if (d == NULL) {
      printf("Erreur allocation memoire\n");
      exit(1);
    }
    d→deg = cur→deg + 1;
    d→coef = cur→coef / cur→deg;
    if (prem) {
      prem = 0;
      p→prem = d;
    }
    else pred→suiv = d;
    pred = d;
    cur = cur→suiv;
  }
  d = (TMonome *)
malloc(sizeof(TMonome));
  if (d == NULL) {
    printf( "Erreur allocation memoire\n");
    exit(1);
  }
  d→deg = 0;
  d→coef = 1;
  if (prem)  p→prem = d;
  else pred→suiv = d;
}
```

CamScanner