Université de Savoie, UFR SFA — INFO523, L3 INFO
2017-2018 — 14 novembre 2017

Examen, INFO523

Durée : 1h30

Documents autorisés : tous documents du cours/td/tp, notes manuscrites (nb : pas de livres)

Les exercices sont indépendants. Le barème est indicatif. La durée est de 2h pour les étudiants bénéficiant d'un 1/3 temps.

1 Connaissance du C et pile d'exécution (/6)

On vous donne le programme C suivant :

```
void swap    (int* p, int* q)
{
  int t = *p;
  *p    = *q;
  *q    = t;
}
void mystere(int* f, int* m, int* l)
{
  int* n = m;
  // (2) situation initiale
  while ( f != n )
  {
    swap( f++, n++ );
    if      ( n == l ) n = m;
    else if ( f == m ) m = n;
  }
  // (3) situation en fin de boucle
}
```

```
int main()
{
  int t[ 5 ] = { 1, 0, 7, 4, 3 };
  // (1)
  mystere( t, t + 2, t + 5 );
  // (4)
  for ( int i = 0; i < 5; ++i )
    printf( " %d", t[ i ] );
  printf( "\n" );
  return 0;
}
```

1. (/2,5) Que contient le tableau `t` à la fin ? Plus généralement, que fait la fonction `mystere` ?
2. (/3,5) Afficher la pile d'exécution de ce programme aux lignes (1), (2), (3) et (4), comme précisé dans le programme.

2 Nombres d'inversion dans un tableau (/4)

Ecrire la fonction `int nbInversion( int* T, int n )` qui compte le nombre d'inversion de suites croissantes / décroissantes dans le tableau `T` de taille `n`. Par exemple, le tableau ci-dessous contient 2 inversions :

```
int tab[] = { 3, 1, -2, 5, 9, 11, 8, 4, -1 };
```

3 Polynômes (/12)

On va représenter les polynômes, e.g. $P(x) = x^5 - 3x^2 + 2$, sous forme de liste simplement chaînée de monômes triés en partant de celui de degré le plus petit, donc sur l'exemple précédent : 2, $-3x^2$, x^5.

Chaque monôme contiendra deux valeurs (le coefficient et le degré) et un pointeur vers le monôme de degré supérieur ou null si c'était déjà le monôme de degré maximal. Sur l'exemple précédent, notre polynôme P peut être modélisé ainsi :

```
P--> coef=2, deg=0, succ--> coef=-3, deg=2, succ--> coef=1, deg=5, succ=null
```

On en déduit les types C suivants :

```
typedef struct SMonome {
  double coef;
  int    deg;
  struct SMonome* succ;
} Monome;
typedef Monome* Polynome;
```

Par convention, le polynôme "null" est le polynôme $P(x) = 0$.

1. (/1) Ecrire un petit programme qui crée le polynôme de l'exemple ci-dessus sans allocation dynamique.
2. (/1.5) Ecrire la fonction qui affiche un polynôme (de manière simple).

   ```
   void      P_affiche      ( Polynome  P );
   ```

   Sur le polynôme précédent, cela afficherait `(2*x^0)+(-3*x^2)+(1*x^5)`.
3. (/2) On veut maintenant créer les polynômes en leur ajoutant (dynamiquement) des monômes. On veut donc écrire une fonction qui rajoute un monôme à un polynôme existant. Ce n'est malheureusement pas si simple car ajouter un monôme peut parfois l'enlever de la liste ! (Ex : On rajoute le monôme $3x^2$ à $P(x)$ au-dessus.)

   On va donc d'abord écrire une fonction qui cherche le monôme qui précède un monôme d'un certain degré d dans P, où qui retourne `null` s'il doit être en première place.

   ```
   Monome* P_cherche_pred( Polynome P, int d );
   ```

   Par exemple `P_cherche_pred( P, 2)` retourne l'adresse du monôme $2x^0$ sur l'exemple au-dessus.
4. (/3) Ecrire maintenant la fonction qui rajoute un monôme à un polynôme existant, en utilisant la fonction précédente.

   ```
   void P_ajout( Polynome* P, double c, int d );
   ```

   NB : Notez que P est passé par pointeur pour pouvoir être modifié le cas échéant. Cette fonction peut créer un monôme, supprimer un monôme ou modifier un monôme.
5. (/3) On veut maintenant évaluer le polynôme en un point x donné. Faites-le de la manière la plus efficace possible selon vous.

   ```
   double      P_eval         ( Polynome  P, double x );
   ```

   NB : vous pouvez utilisez la fonction `double pow( double x, int n)` qui calcule x^n, ... ou faire autrement.
6. (/1.5) Ecrire enfin la fonction qui réinitialise le polynôme au polynôme 0 (i.e. null).

   ```
   void P_termine( Polynome* P );
   ```