MASTER 2 MPM **Université Paris XIII**
Année 2011-2012 **Institut Galilée**

Langage C avancé pour la simulation numérique

Examen du 20 mars 2012

Durée : 3h

Aucun document autorisé
Aucun appareil electronique autorisé

Exercice 1 : (3 points)

Listing 1: exo1.c

```
1  #include <stdio.h>
2
3  void f(double *i, double j){
4    *i=j+3/2;
5    j=*i-1.;
6  }
7
8  void g(double x,double *y){
9    *y=x+1.;
10   x=*y-2.;
11 }
12
13 int main(){
14   double s=1.,t=-1;
15   f(&s,t);
16   printf("s=%lf, t=%lf\n",s,t);
17   g(s,&t);
18   printf("s=%lf, t=%lf\n",s,t);
19   return 0;
20 }
```

Q. 1 *Donner la ou les commandes Linux permettant de créer le programme executable* `exo1` *associé au fichier* `exo1.c`. ▪

Q. 2 *A l'aide de diagrammes de représentation mémoire, expliquer le déroulement de ce programme et donner l'affichage.* ▪

Exercice 2 : (7 points)

De manière générique, un polynôme P de degré N peut s'écrire sous la forme

$$\forall x \in \mathbb{R},\ P(x) = \sum_{i=0}^{N} a_i x^i$$

avec $a_i \in \mathbb{R},\ \forall i \in \{0, \ldots, N-1\}$, $a_N \in \mathbb{R}^*$

Q. 1 *Définir le type* `Poly` *correspondant à une structure de champs :*

- *N, un entier qui correspondra au degré,*
- *A un pointeur de* `double` *qui contiendra l'ensemble des a_i, $\forall i \in \{0, \ldots, N\}$.* ▪

Q. 2 *1. Ecrire la fonction* `AllocPoly` *permettant d'«initialiser» correctement les deux champs N et A d'une variable de type* `Poly` *correspondant à un polynôme quelconque de degré k donné.*

2. Ecrire la fonction `FreePoly` *permettant de désallouer «proprement» une variable de type* `Poly`. ▪

Q. 3 *Ecrire la fonction* `freadPoly` *permettant de créer une variable de type* `Poly` *à partir d'un fichier. Ce fichier est de la forme*

```
N
a0
a1
...
aN
```

Par exemple le fichier `exemple.poly` *contient*

```
3
1.0000
-2.0000
3.0000
-4.0000
```

et correspond au polynôme de degré 3

$$P(x) = 1 - 2x + 3x^2 - 4x^3.$$

Q. 4 *Ecrire la fonction* `printPoly` *permettant d'afficher le contenu d'une variable de type* `Poly`. *Avec l'exemple precedant, cette fonction affiche :*

```
1-2*x+3*x^2-4*x^3
```

Q. 5 *Soit P un polynôme et $x \in \mathbb{R}$ donnés, écrire la fonction* `evalPoly` *permettant de calculer $y = P(x)$.* ▪

Q. 6 *Soit P un polynôme et $r \in \mathbb{N}^*$ donnés, écrire la fonction* `diffPoly` *permettant de calculer le polynôme Q défini par $Q = \frac{d^r P}{dx^r}$.* ▪

On suppose l'ensemble des fonctions `AllocPoly`, `freadPoly`, `printPoly`, `evalPoly` et `diffPoly` écrites dans le fichier `Poly.c`.

Q. 7 *Ecrire le fichier header* `Poly.h` *associé au fichier* `Poly.c`. ▪

Q. 8 *Ecrire le fichier* `TestPoly.c` *contenant le programme principal et effectuant les opérations suivantes :*

```
Lecture du polynome P contenu dans le fichier <exemple.poly>
Calculer x <- P(1)
Afficher x
Determiner Q polynome tel que Q est la derivee seconde de P
Afficher Q
Calculer x <- Q(1)
Afficher x
Quitter proprement!
```

Q. 9 *Donner les commandes permettant de créer l'executable* `TestPoly` *avec la possibilité de debuggage.* ▪

Exercice 3 : (10 points)

Dans cet exercice, nous allons implémenter et utiliser quelques outils d'algèbre linéaire.

Q. 1 *Définir le type* `vector` *correspondant à une structure ayant pour champs :*

- `dim` *: dimension du vecteur (type* `int`*),*

- *pval : pointeur sur les composantes du vecteur (type* `double *`*).* ▪

Q. 2 *Nous voulons écrire une fonction* `AllocVector` *permettant d'allouer un objet de type* `vector` *à la dimension N.*

1. *Donner deux prototypes possibles (non trivialement similaire) pour la fonction* `AllocVector`*.*
2. *Choisir un prototype et écrire la fonction associée.* ▪

Q. 3 1. *Ecrire une fonction* `freeVector` *permettant de désallouer le champ* `val` *d'un objet de type* `vector`*.*

2. *Justifier le choix de ses arguments!* ▪

Q. 4 1. *Ecrire une fonction* `FREEVector` *permettant de désallouer le champ* `val` *d'un objet de type* `vector`*, de mettre à zéro le champ* `dim` *et à* `NULL` *le champ* `val`*.*

2. *Justifier le choix de ses arguments!* ▪

Q. 5 *Nous voulons écrire une fonction* `RandVector` *permettant d'initaliser chacune des composantes d'un vecteur par un nombre aléatoire (utiliser la fonction drand48() du langage C qui retourne un* `double` *entre 0 et 1 :* `double drand48 (void);`*). Le vecteur est supposé déjà alloué.*

1. *Donner deux prototypes possibles (non trivialement similaire) pour la fonction* `RandVector`*.*
2. *Choisir un prototype et écrire la fonction associée.* ▪

Q. 6 *Soient* $\boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \in \mathbb{R}^n$, $\boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n$ *et* α, β *deux réels. Les fonctions* `ScalarProd`, `aUpbV` *et* `Norm` *correspondent à :*

$$\begin{array}{lll} \texttt{ScalarProd} & : & \langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} u_i v_i, \\ \texttt{aUpbV} & : & \alpha \boldsymbol{u} + \beta \boldsymbol{v}, \\ \texttt{Norm} & : & \|\boldsymbol{u}\| = \left(\sum_{i=1}^{n} u_i^2 \right)^{1/2}. \end{array}$$

1. *Pour chacune des trois fonctions, donner deux prototypes possibles (non trivialement similaire).*
2. *Pour chacune des trois fonctions, choisir un prototype et écrire la fonction associée.* ▪

Q. 7 *Ecrire un* `main` *simple et fonctionnel utilisant au moins les fonctions* `RandVector`, `ScalarProd`, `aUpbV` *et* `Norm`*.* ▪