Nom Prénom : Groupe :

Université de Bordeaux DS du 8 mars 2016
Département Licence, **J1MI2013** 14h–15h30

Le sujet comporte 4 exercices

Aucun document n'est autorisé.

Exercice 1 Questions de Cours

1. On suppose disposer d'une fonction capable de calculer la moyenne de plusieurs entiers. Le code de cette fonction est écrit dans un fichier appelé `moyenne.c`.
   L'exécutable est obtenu suite à la compilation suivante :
   ```
   gcc -std=99 -Wall moyenne.c -o moyenne
   ```
   L'utilisateur exécute la commande suivante :
   ```
   ./moyenne 11 8 13 12 7 10
   ```
   Compléter le tableau suivant en indiquant pour chacune des expressions son type et sa valeur.

	Type	Valeur
`argc`	int	7
`argv[2]`	char *	"8"
`atoi(argv[5])`	int	7
`argv[7]`	void *	Null

2. Soit `b` une variable de type `bool`. Compléter le tableau suivant en indiquant les différentes valeurs de `b`. Justifiez brièvement votre réponse.

`bool b`	Valeur de `b`	Justification
`b=(!8);`	false	Tout entier non nul correspond à `true` et `!true = false`, donc `b` vaut `false`.
`b= (!0) && (3||0);`	`true`	L'entier 0 correspond à `false`, donc `(!0) && (3||0) =!false && (true||false)`. Ainsi `b` vaut `true && true`, c'est à dire `true`.

`int d; int c;`
`int a=1;`
`b=(a || (d==c));`
	`true`	D'après "l'évaluation paresseuse", comme `a` vaut `true`, `b` vaut `true`.

`int d; int c;`
`int a=0;`
`b=(a && (d==c));`
	`false`	D'après "l'évaluation paresseuse", comme `a` vaut `false`, `b` vaut `false`.

Exercice 2

On dira qu'un tableau `t` à `n` éléments possède un "pic" à la position `i` si la valeur `t[i]` est strictement plus grande que `t[i-1]` et `t[i+1]`.
Notez bien que `t` ne peut pas avoir de pic en position `0`, ni en position `n-1`.
De même, un tableau à moins de trois éléments ne peut pas posséder de pics.

1. Donner le nombre de pics du tableau `t=[0,1,0,1,2,-1,3,3,0,0,1]` à `n=11` éléments ainsi que leurs positions éventuelles.

   Le nombre de pics du tableau `t` est 2. Ils sont en position 1 et 4.

2. Écrire une fonction `nombrePics` qui calcule et renvoie le nombre de pics d'un tableau `t` de `n` entiers.

```
1  int nombrePics(int t[], int n){
2      int nb_pics = 0;
3      for(int i=1; i<n-1: i++){
4          if(t[i-1]<t[i] && t[i]>t[i+1]){
5              nb_pics ++;
6          }
7      }
8      return nb_pics;
9  }
```

Exercice 3 La fonction `mystere` donnée ci-dessous prend en paramètre un entier naturel :

```
1  int mystere(int n){
2    if ( n==0 ){
3      return 1;
4    }
5    int k = 0;
6    while ( n>0 ){
7      n = n/10;
8      k = k+1;
9    }
10   return k;
11 }
```

1. Quelle est la valeur renvoyée par l'appel `mystere(57312)` ?

   La valeur renvoyée par l'appel `mystere(57312)` est `5`.

2. Que renvoie la fonction en général si on lui passe en paramètre un entier naturel `n` ?

   En général, la fonction renvoie le nombre de chiffre de `n`.

3. Écrire une fonction `int chiffreMax(int n)` qui calcule et renvoie le plus grand chiffre de l'entier naturel `n`. Par exemple si `n` vaut 57321, la fonction doit renvoyer 7.

```
1  int chiffreMax(int n){
2      int max = 0;
3      int r;
4      while( n>0 ){
5          r = n%10;
6          if( r>max ){
7              max = r;
8          }
9          n = n/10;
10     }
11     return max;
12 }
```

Exercice 4

On dira qu'un tableau `t` de `n` entiers est *strictement croissant* si les valeurs qu'il contient sont toutes différentes et rangées dans l'ordre croissant lorsqu'on les lit de gauche à droite.
Par exemple `t=[3,6,7,8]` est strictement croissant, mais `t=[3,6,6,7,8]` ne l'est pas.
Par convention, un tableau contenant 0 ou 1 élément est considéré comme strictement croissant.

Dans les questions 1 et 2 suivantes on considère des **tableaux d'entiers dont toutes les valeurs sont distinctes**.

1. Écrire une fonction `bool estStrictementCroissant(int t[], int n)` qui retourne `true` si le tableau `t` de `n` entiers est strictement croissant, et `false` sinon.

```
bool estStrictementCroissant(int t[], int n){
    for(int i=1; i<n; i++){
        if( t[i-1]>t[i] ){
            return false;
        }
    }
    return true;
}
```

   On dira qu'un tableau est *strictement monotone* si les valeurs qu'il contient sont rangées soit dans l'ordre strictement croissant, soit dans l'ordre strictement décroissant.
   Par exemple, `t=[2,5,6,8]` est strictement monotone, ainsi que `t=[7,6,3,1]`, mais `t=[2,3,2,1]` ne l'est pas.
   Par convention, un tableau contenant 0 ou 1 élément est considéré comme strictement monotone.

2. Écrire une fonction `bool estMonotoneStrict(int t[], int n)` qui retourne `true` si le tableau `t` de `n` entiers est strictement monotone, et `false` sinon. **Votre fonction ne devra parcourir le tableau qu'une seule fois.**

```
bool estMonotoneStrict(int t[], int n){
    if( n==0 || n==1 ){
        return true;
    }
    if( t[0]<t[1] ){
        for(int i=2; i<n; i++){
            if( t[i-1]>t[i] ){
                return false;
            }
        }
    }
    else{
        for(int i=2; i<n; i++){
            if( t[i-1]>t[i] ){
                return false;
            }
        }
    }
    return true;
}
```

On considère maintenant des tableaux d'entiers dont les valeurs **ne sont pas** nécessairement toutes distinctes. On dira qu'un tableau `t` est *croissant* (respectivement *décroissant*) si les valeurs qu'il contient sont croissantes (respectivement *décroissantes*) lorsqu'on les lit de gauche à droite. On dira qu'un tableau est *monotone* si les valeurs qu'il contient sont rangées soit dans l'ordre croissant soit dans l'ordre décroissant. Par convention, un tableau contenant 0 ou 1 élément est considéré comme monotone.

3. Écrire une fonction `bool estMonotone(int t[], int n)` qui retourne `true` si le tableau `t` de `n` entiers est monotone, et `false` sinon.
Votre fonction ne doit utiliser qu'une seule structure de boucle et n'appeler aucune autre fonction.

Solution 1 :

```
1   bool estMonotone(int t[], int n){
2       int i=1;
3       bool c=true;
4       bool d=true;
5       while( i<n && (c||d) ){
6           if( t[i-1]<t[i] )
7               d=false;
8           if( t[i-1]>t[i] )
9               c=false;
10          i++;
11      }
12      return (c||d);
13  }
```

Solution 2 :

```
1   bool estMonotone(int t[], int n){
2       int i=1;
3       bool c=t[0]<=t[n-1];
4
5       for(int i=1; i<n; i++){
6           if(c){
7               if( t[i-1]>t[i] ){
8                   return false;
9               }
10          }
11          else{
12              if( t[i-1]<t[i] ){
13                  return false;
14              }
15          }
16      }
17      return true;
18  }
```

4. En réutilisant les fonctions demandées dans les questions précédentes, écrire une fonction `bool estCroissant(int t[], int n)` qui retourne `true` si le tableau `t` de `n` entiers est croissant, et `false` sinon. **Votre fonction ne doit pas contenir de structures de boucle.**

```
1    bool estCroissant(int t[], int n){
2        if( n==0 ){
3            return true;
4        }
5        if( t[0]<=t[n-1] && estMonotone(t,n)){
6            return true;
7        }
8        return false;
9    }
```