ICC MT & EL Test C++ /White vendredi 5 novembre 2021

Ne PAS retourner ces feuilles avant d'en être autorisé!

Merci de poser votre carte CAMIPRO en évidence sur la table.

Vous pouvez déjà compléter et lire les informations ci-dessous:

NOM _____

Prénom _____

Numéro SCIPER _____

Signature _____

BROUILLON : Ecrivez aussi votre NOM-Prénom sur la feuille de brouillon fournie. Toutes vos réponses doivent être sur cette copie d'examen. Les feuilles de brouillon sont ramassées pour être immédiatement détruites.

Le test écrit commence à: **14h15**

Nous recommandons de passer à l'autre examen à : **15h25**

Les deux copies d'examens sont ramassées à : **16h45**

Le contrôle de ICC reste SANS appareil électronique

Vous avez le droit d'avoir tous vos documents **personnels** sous forme papier: dictionnaire, livres, cours, exercices, code, projet, notes manuscrites, etc...

Vous pouvez utiliser un crayon à papier et une gomme

Ce contrôle écrit de C++ permet d'obtenir **19 points** sur un total de 100 points pour le cours complet.

1) (4 pts) : Compléter une fonction vérifiant si la valeur absolue d'un entier est un palindrome

Un entier positif est un **palindrome** s'il est symétrique. Par exemple, **414** et **123321** sont des palindromes et **412** et **1231** n'en sont pas. Un entier ayant un seul chiffre est aussi un palindrome. On demande de compléter la fonction **`palindrome`** qui accepte un entier en entrée et qui doit déterminer si *sa valeur absolue* est un **palindrome** ou pas.

1.1) Remplacer les 4 expressions numérotées __(numéro)__ par un opérateur du C++ :

__(1)__ doit être remplacé par l'opérateur :

__(2)__ doit être remplacé par l'opérateur :

__(3)__ doit être remplacé par l'opérateur :

__(4)__ doit être remplacé par l'opérateur :

```
1  void palindrome(int num)
2  {
3       int n(0), digit(0), rev(0);
4
5       n = num = (num>0)? num : -num ; // valeur absolue
6
7       do
8       {
9           digit = num __(1)__ 10;
10          rev   =(rev __(2)__ 10) __(3)__ digit;
11          num   = num __(4)__ 10;
12      } while (num != 0);
13
14      cout << " The reverse of n is: " << rev << endl;
15
16      if (n == rev)
17          cout << " The number is a palindrome.";
18      else
19          cout << " The number is not a palindrome.";
20 }
```

1.2) on effectue l'appel **`palindrome(718)`** : utiliser le nombre de lignes nécessaires pour indiquer les valeurs intermédiaires des variables **`digit`**, **`rev`** et **`num`** au moment de chaque exécution de la ligne 12 :

passage	digit	rev	num
1			
2			
3			
4			
5			
6			

2) (4 pts) Surcharge et valeur par défaut

Ce programme compile en C++11 et s'exécute correctement

```
1   #include <iostream>
2   using namespace std;
3
4   char f ( int n )
5   {
6         return (n + 3) % 26 + 'a';
7   }
8
9   int f ( char c )
10  {
11       return 10 * ( c - 'a' );
12  }
13
14  int f ( double d = 1.5 )
15  {
16       int e(d) ;
17       return f(2*e) - 'a' ;
18  }
19
20  int main ()
21  {
22       // Question 2.1
23       cout << f( 'd' ) << endl;
24       cout << f( 5 )   << endl;
25
26       // Question 2.2
27       cout << f ( 10  / 2  )  << endl;
28       cout << f ( 10. / 2  )  << endl;
29       cout << f ( 10  / 2. )  << endl;
30       cout << f ( 10. / 2. )  << endl;
31
32       // Question 2.3
33       cout << "f () :" << f () << endl;
34       return 0;
35  }
```

2.1) justifier l'affichage obtenu pour l'exécution des lignes suivantes :
Ligne 23 :

Ligne 24 :

2.2) y a-t-il des différences d'affichage pour l'exécution des lignes 27 à 30 ?
Justifier les éventuelles différences d'affichage.

2.3) L'exécution de f() à la ligne 33 produit-elle un affichage ? Justifier la réponse.

3) (4 pts) Entrées-sorties standards

Le code suivant compile en C++11 et s'exécute correctement.

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int number(0);
7       int sum(0);
8
9       cout << "Enter a number: ";
10      cin  >> number;
11
12      while (number >= 0)
13      {
14          sum += number;
15          cin >> number;
16      }
17      cout << sum  << endl;
18
19      return 0;
20  }
```

Expliquer l'exécution du programme. Indiquer s'il y a *un affichage produit par le programme ou pas*. Si *oui lequel*. **Attention l'exécution est relancée pour chacune des questions de 3.1 à 3.4.**
Remarque : pour chaque cas, on indique la suite des touches du clavier qui ont été frappées. L'indication **Enter** signifie qu'on a frappé une fois la touche de validation (passage à la ligne).

3.1) frappe des touches :

2	0	Enter

3.2) frappe des touches:

-	1	0	Enter

3.3) frappe des touches :

1	5	Enter	1	5	Enter	-	1	5	Enter

3.4) frappe des touches (une case vide signifie qu'on a frappé une fois la barre d'espace) :

4	0		4	0		-	2	0	Enter

4) (3 pts) Récursivité

Le code suivant compile en C++11 et s'exécute correctement.

```
1   #include <iostream>
2   using namespace std;
3
4   void f(char c);
5   int main()
6   {
7       char c('A');
8       cin >> c;
9       f(c);
10      cout << c;
11      return 0;
12  }
13
14  void f(char c)
15  {
16      if (c != '.')
17      {
18          char new_char('A');
19          cin >> new_char;
20          f(new_char);
21          cout << new_char;
22      }
23  }
```

Voici la suite des touches frappées au clavier pour l'exécution de ce programme (l'indication **Enter** signifie qu'on a frappé une fois la touche de validation (passage à la ligne)) :

L	Y	N	X	.	Enter

4.1) Utiliser le nombre de lignes nécessaires pour indiquer, pour chaque appel récursif, la valeur reçue à la **ligne 14** pour le paramètre formel **c** de la fonction **f**, la valeur de la variable **new_char** après exécution de la **ligne 19** et s'il produit un autre appel récursif (si oui avec quel argument en **ligne 20**).

appel	c Ligne 14	new_char Ligne 19	Autre appel récursif ? si oui, avec quel argument ? Ligne 20
1			
2			
3			
4			
5			
6			

4.2) S'il y en a un, quel est l'affichage produit dans le terminal ?

4.3) En bref, que se passe-t-il pour le même input si on inverse les **lignes 20** et **21** du code ? Y a-t-il un affichage ? Si oui, lequel ?

5) (4 pts) Priorité des opérateurs et évaluation d'expressions

Le code suivant compile en C++11 et s'exécute correctement.

```
1   #include <iostream>
2   using namespace std;
3
4   int main ()
5   {
6         int a(1);
7         int j(2 + 2 * --a);
8         cout << " j = " << j << endl;
9         cout << " a = " << a << endl;
10
11        int k(2 + 1 / 2 + 1);
12        cout << " k = " << k << endl;
13
14        bool b(k % 2 == 0);
15        cout << " b = " << b << endl;
16
17        for( int i = 0; i >= 0; i = ( i > 3 ) ? i - 5 : i + 2 )
18        {
19              cout <<  " i = " << i << endl;
20        }
21        return 0;
22  }
```

Compléter la table en précisant l'affichage obtenu à l'exécution de ce programme. Indiquer ce qui est affiché (2ième colonne) et justifier comment cette valeur a été obtenue (colonne de droite), c'est-à-dire ***avec quelle expression et comment cette expression a été évaluée ; indiquer la valeur des sous-expressions s'il y en a***. Ajouter des lignes si nécessaire.

N° Ligne	Affichage	Justification
8		
9		
12		
15		
19		