2 - Eléments de C++

1. Valeurs par défaut des paramètres

Quel texte sera affiché très exactement par le programme ci-dessous ?

```
#include <iostream>

using namespace std;

void afficher(int i)
{
    cout << "i: " << i << endl;
}

void afficher(int i, int j, int k = 0)
{
    cout << "i: " << i << "; j: " << j << "; k: " <<
       k << endl;
}

int main(void)
{
    afficher(1);
    afficher(1, 2);
    afficher(1, 2, 3);
    return 0;
}
```

2. Les références

2.1. Utilisation des références

Le programme ci-dessous utilise des affectations de variables et de références.

Il permet de tester l'utilisation des références dans différentes situations.

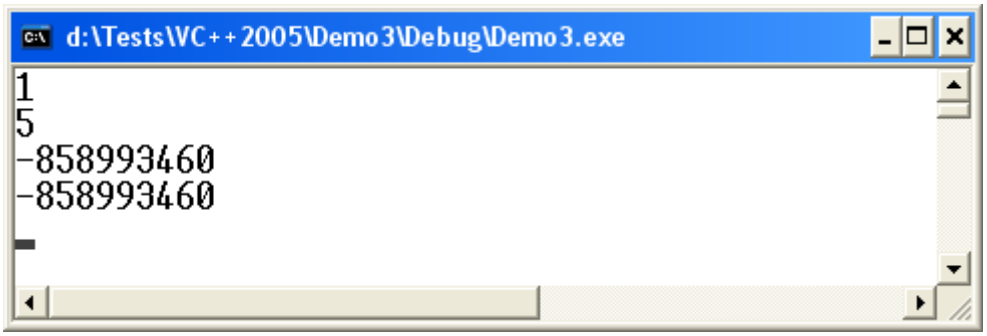
Indiquer dans chaque cas quel est l'affichage obtenu.

```
#include <iostream>

using namespace std;

int main(void)
{
    int i = 1;
    int j = 10;
    int & ri = i;
    int & rj = j;

    cout << "a) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    i = 2;
    cout << "b) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    ri = 3;
    cout << "c) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    rj = ri;
    cout << "d) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    i = 4;
    cout << "e) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    rj = 11;
    cout << "f) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    j = 12;
    cout << "g) " << "i:" << i << " ; ri:" << ri  <<
        " ; j:" << j  << " ; rj:" << rj << endl;
    return 0;
}
```

cas	i	ri	j	rj
a)				
b)				
c)				
d)				
e)				
f)				
g)				

2.2. *Problèmes avec les références*

Le programme ci-dessous fournit l'affichage suivant :



Comment expliquer les valeurs affichées ?

Quelle incorrection grave comporte ce programme ?

```
#include <iostream>

using namespace std;

int globale = 1;

int & fonction(bool b)
{
    int locale;
    if (b)
        return locale;
    else
        return globale;
}

int main(void)
{
    cout << fonction(false) << endl;
    fonction(false) = 5;
    cout << fonction(false) << endl;
    cout << fonction(true) << endl;
    fonction(true) = 6;
    cout << fonction(true) << endl;

   return 0;
}
```

3. Allocation dynamique de mémoire en C++

On considère la déclaration suivante pour une liste chaînée :

```
typedef struct element
{
   float valeur;
   element *suivant;
} T_ELEMENT;


T_ELEMENT * e;
```

Ecrire les instructions C++ permettant :

1. De faire pointer la variable « e » sur une structure T_ELEMENT allouée dynamiquement en mémoire.

2. De libérer la mémoire précédemment allouée.

3. De faire pointer la variable « e » sur un tableau de 100 structures T_ELEMENTS allouées dynamiquement en mémoire

4. De libérer le tableau précédemment alloué.