IF122 : Correction du contrôle intermédiaire

Exercice 1 : Voici deux solutions possibles :

```
But: Retourne "true" si a est divisible par b
Préconditions: a>=0 et b>0
bool est_divisible_par(int a, int b) {
    if(a==0) {
        return true;
    } else if (a < b) {
        return false;
    } else {
        return est_divisible_par(a-b, b);
    }
}
```

La deuxième solution consiste à considérer que la fonction précédente est une fonction auxilliaire et tester les préconditions avant d'appeler cette fonction :

```
But: Retourne "true" si a est divisible par b
bool est_divisible(int a, int b) {
    if (b==0) {
        return false;
    } else if (a<0) {
        return est_divisible(-a,b);
    } else if (b<0) {
       return est_divisible_par(a,-b);
    } else {
       return est_divisible_par(a,b);
    }
}
```

Exercice 2 :

```
static int v = 0;       : globale, X, statique
int w = 1;              : globale, X, non statique
int f(int v) {          : argument, valeur, X
  v=v+1;
  return v;
}
int h(int &v) {         : argument, référence, X
  v = v - w;
  return v;
}
int i(int w) {          : argument, valeur, X
  static int v = 10;    : locale, X, statique
  w = v+1;
  return v;
}
int j(int v) {          : argument, valeur, X
  return f(v)+1;
}
```

Exercice 3 : L'affichage nous donne : 0, 5, 1, 0, 10, 7, 0, 10

Exercice 4 :

- $\#include < iostream >$: inclusion de la bibliothèque définissant cin et cout ;
- $int\ main()\{$: déclaration de la fonction principale du programme ;
- $int\ *pe1...$: déclaration d'une variable locale pe1 en tant que pointeur vers un entier ;
- $... = new\ int[5]$: réservation dynamique de 5 cases mémoires consécutives stockant des entiers. pe1 pointe vers la première de ces 5 cases.
- $int\ **pe2...$: déclaration d'une variable locale pe2 en tant que pointeur vers un pointeur d'entier ;
- $... = new\ (int*)[6]$: réservation dynamique de 6 cases mémoires consécutives stockant des "pointeurs d'entier". pe2 pointe vers la première de ces 6 cases.
- $pe2[0] = pe1$: affectation de pe1 à la première case de pe2. pe2 est un pointeur vers un tableau de 6 cases de "pointeur d'entier". La case 0 est donc de type "pointeur d'entier" et pe1 est un pointeur d'entier. L'affectation est donc autorisée.
- $for(int\ i = 0; i < 6; i++)$: boucle faisant varier l'indice i de 0 inclus jusqu'à 6 exclus. L'indice i est défini uniquement à l'intérieur de cette boucle ;
- $cin >> pe1[i]$: demande une valeur à l'utilisateur et la stocke dans la case i de pe1. pe1 est un tableau d'entier donc la valeur attendue est un entier (pas de problème pour cin). Par contre pe1 est associé à un tableau de 5 cases donc la case d'indice 5 n'existe pas. La boucle doit être corrigée pour faire varier i de 0 à 5 exclus.
- $for(int\ i = 0; i < 6; i++)$: boucle faisant varier l'indice i de 0 inclus jusqu'à 6 exclus.
- $cout << pe2[i][0]$: pe2 est un pointeur vers un pointeur d'entier. Il pointe vers un tableau de 6 cases. $pe2[i]$ accède à la i-ème case de ce tableau. Or si la première case de ce tableau est pe1, rien n'a été associé aux autres cases. Seul $pe2[0][0]$ a un sens : il accède à la case 0 de pe1.
- Le programme principal se termine mais les deux zones mémoires n'ont pas été libérées par un delete ;
- La fonction main est déclarée comme retournant un entier mais il n'y a aucun return dans la fonction.

Exercice 5 :

```
t_list cons_sorted(t_element e, t_list l) {
    if (isEmpty(l)) {
        return cons(e,l);
    } else if (!lower_element(head(l), e)) {
        return cons(e,l);
    } else {
        return cons(head(l), cons_sorted(e, queue(l)));
    }
}
```

Exercice 6 :

```
t_list search_sorted(t_element e, t_list l) {
    if (isEmpty(l)) {
        return EMPTY_LIST;
    } else if (lower_element(e, head(l))) {
        return EMPTY_LIST;
    } else if (equal_element(e, head(l))) {
        return l;
    } else {
        return search_sorted(e, queue(l)));
    }
}
```