Correction des exercices du chapitre 2

1. (a) $x = 1$, $y = -1$, $z = -1$
   Explication : La variable x n'est incrémentée qu'après l'expression alors que la variable y est décrémentée avant. L'expression devient donc : $z = 0 + -1$.

   (b) $x = 1$, $y = 0$, $z = 2$
   Explication : La variable x n'est incrémentée qu'après l'expression. Comme nous lisons de gauche à droite l'opération logique &&, nous regardons d'abord si $x \neq 0$. Ici ce n'est pas le cas. Comme l'opération retournera d'office faux, peu importe ce que donne l'élément à droite de l'opération, $y++$ ne s'effectuera pas. La condition est fausse donc nous obtenons que $z = z - 2$.

   (c) $x = 1$, $y = 1$, $z = 1$
   Explication : Nous incrémentons d'abord la variable x. Comme $x \neq 0$, la partie gauche de l'opération && est vraie et nous regardons maintenant si c'est également le cas pour la partie droite. Nous incrémentons la variable y et donc $y \neq 0$. La condition est vraie. Nous obtenons donc que $z = z + 1$.

   (d) $x = 1$, $y = 1$, $z = 1$
   Explication : L'opérateur || renvoie vrai si au moins une de ses opérandes est vraie. $x > 0$ n'est pas vrai et comme la variable y est incrémentée avant l'expression, nous avons bien que la négation de y est faux. Nous obtenons donc que $z =$ la valeur de la variable x après qu'elle soit incrémentée.

   (e) $x = 1$, $y = 0$, $z = 0$
   Explication : Quand nous avons un opérateur virgule, nous pouvons diviser l'expression en 2 sous-expression. La variable x est incrémentée après la première sous-expression. Nous obtenons donc dans la deuxième sous-expression que $x > 0$. Nous avons alors que $z = 1$.

2. (a)
```
i = 0;
while(i < 1000){
    j += i;
    i += 10;
}
```

   (b)
```
i = 2, j = 0;
while(j < k){
    j += i;
    i += 2;
}
```

(c)
```
i = j = 0;
while(i < 1000){
    if (f(i) > f(j))
        i++;
        continue;
    j += i;
    i++;
}
```

(d)
```
while();
```

3. (a)
```
for (i = 2; i < 100000; i *= 2, j += i);
```

(b)
```
for (p = premier(); !est_dernier(p);
                            p = suivant(p))
    traiter(p);
```

(c)
```
for (; attendre(););
```

(d)
```
for (i = 0, j = 0; j < k; j += i++);
```

4.1.
```
#include <stdio.h>
#include <math.h>
int main() {
    float a, b, c;
    scanf("%f %f %f", &a, &b, &c);
    if(a == 0) {
        printf("%f\n", -c/b);
    }
    else {
        float delta, x_1, x_2, x;
        delta = b*b - 4*a*c;
        if(delta == 0) {
            x = -b/(2*a);
            printf("%f\n", x);
        }
        else if(delta < 0) {
            printf("L'equation n'a pas de
                solution dans les reels\n");
        }
        else {
            x_1 = (-b + sqrt(delta))/(2*a);
            x_2 = (-b - sqrt(delta))/(2*a);
            printf("1e racine: %f\n", x_1);
```

```
            printf("2e racine: %f\n", x_2);
        }
    }
}
```

4.2.
```
#include <stdio.h>
int main(){
    int n, k, multifactorielle = 1;
    scanf("%d %d", &n, &k);
    for (int i = n; i > 0; i -= k)
        multifactorielle *= i;
    printf("%d\n", multifactorielle);
}
```

4.3.
```
#include <stdio.h>
int main(){
    float nb;
    scanf("%f", &nb);
    float petit = nb, grand = nb, somme = nb,
                                  produit = nb;
    int nb_nb = 1;
    while(scanf("%f", &nb)){
        if (nb < petit)
            petit = nb;
        if (nb > grand)
            grand = nb;
        somme += nb;
        produit *= nb;
        nb_nb += 1;
    }
    float moyenne = somme/nb_nb;
    printf("%f %f %f %f %f\n", somme, produit,
                    moyenne, grand, petit);
}
```

4.4.
```
#include <stdio.h>
int main(){
    int nb, premier = 1;
    scanf("%d", &nb);
    for (int i = 2; i*i <= nb && premier; i++){
        if (!(nb%i))
            premier = 0;
    }
    if (premier)
        printf("Nombre premier\n");
    else
```

```
            printf("Nombre pas premier\n");
    }
```

4.5.
```
#include <stdio.h>
int main(){
    int nb, log = 0;
    scanf("%d", &nb);
    for(int puissance2 = 2; puissance2 <= nb;
                                            log++)
        puissance2 *= 2;
    printf("%d\n", log);
}
```

4.6.
```
#include <stdio.h>
int main(){
    int resultat = 1, base, exposant, modulo;
    scanf("%d %d %d", &base, &exposant, &modulo);
    for(int i = 0; i < exposant; i++)
        resultat = (resultat * base)% modulo;
    printf("%d\n", resultat);
}
```

5.
```
#include <stdio.h>
int main(){
    int f = 1, f_1 = 1, f_2 = 1, n;
    scanf("%d", &n);
    for (int i = 2; i < n; i++){
        f = f_1 + f_2;
        f_2 = f_1;
        f_1 = f;
    }
    printf("%d\n", f);
}
```

6.
```
#include <stdio.h>
int main(){
    int nb_etapes, z;
    scanf("%d", &z);
    for (nb_etapes = 0; z != 1; nb_etapes++){
        if (!(z%2))
            z = z/2;
        else
            z = 3*z + 1;
    }
    printf("%d\n", nb_etapes);
}
```

7.
```c
#include <stdio.h>
int main(){
    int n, m;
    scanf("%d %d", &n, &m);
    int fact = 1;
    if (m > (n-m)){
        for (int i = n; i > m; i--){
            fact *= i;
            fact /= (i - m);
        }
    }
    else {
        for (int i = n; i > n-m; i--){
            fact *= i;
            fact /= (i - (n - m));
        }
    }
    printf("%d\n", fact);
}
```

8.
```c
#include <stdio.h>
int main(){
    int nb;
    scanf("%d", &nb);
    for (int i = 2; nb > 1; i++){
        int premier = 1;
        for (int j = 2; (j*j) <= i && premier; j++){
            if (!(i%j))
                premier = 0;
        }
        if (premier){
            int puissance = 0, k = 1;
            while (!(nb%(k*i))){
                k *= i;
                puissance++;
            }
            if (puissance > 0){
                if (puissance > 1)
                    printf("%d^%d ", i, puissance);
                else
                    printf("%d ", i);
                nb /= k;
            }
        }
    }
    printf("\n");
```

```
}
```