# Test Structurel

Licence 3 – année 2017-2018

Travaux Dirigés

## Exercice 1

Soit le programme suivant :

```
int foo0(int i) {
    int j;
    j = 2;

    if (i <= 16) {
        j = j * i;
    }

    if (j > 8) {
        j = 0 ;
    }

    return j;
}
```

### Questions :

1. Distinguer les blocs d'instructions de ce programme,
2. Établir son graphe de contrôle,
3. Fournir l'expression des chemins, et donner le nombre de chemins,
4. Produire un ensemble de chemins d'exécution permettant de satisfaire un-à-un les critères suivants de couverture du graphe de contrôle, et fournir des données de test permettant de sensibiliser les chemins de manière à satisfaire les critères :
    a. Tous-les-nœuds,
    b. Tous-les-arcs,
    c. Tous-les-chemins.

## Exercice 2

Soit le programme suivant :

```
short foo1 (short i) {
    short j, r;
    j = 1 ;

    while (i > 0)  {
        j = j * i;
        i = i - 1;
    }

    if (j > 1) {
        r = j ;
    }
    else {
        if (i < 0) {
            r = 0 ;
        }
        else {
            r = 1 ;
        }
    }

    return r ;
}
```

### Questions :

1. Distinguer les blocs d'instructions de ce programme,
2. Établir son graphe de contrôle,
3. Fournir l'expression des chemins,
4. Produire un ensemble de chemins d'exécution permettant de satisfaire un-à-un les critères suivants de couverture du graphe de contrôle, et fournir des données de test permettant de sensibiliser ces chemins :
    a. Tous-les-nœuds,
    b. Tous-les-arcs,
    c. Tous-les-1-chemins,
    d. Tous-les-2-chemins.

## Exercice 3

Soit le programme suivant :

```
int foo2() {
    int x, y, z, signe;

    scanf(x) ;
    scanf(y) ;
    z = 0 ;
    signe = 1 ;

    if ( x < 0 ) {
        signe = -1 ;
        x = -x ;
    }

    if ( y < 0 ) {
        signe = -signe ;
        y = -y ;
    }

    while ( x >= y ) {
        x = x - y ;
        z = z + 1 ;
    }

    z = signe * z ;
    return z;
}
```

### Questions :

1. Distinguer les blocs d'instructions de ce programme,
2. Établir son graphe de contrôle,
3. Fournir l'expression des chemins,
4. Produire un ensemble de chemins d'exécution permettant de satisfaire un-à-un les critères suivants de couverture du graphe de contrôle, et fournir des données de test permettant de sensibiliser ces chemins :
    a. Tous-les-nœuds,
    b. Tous-les-arcs,
    c. Tous-les-1-chemins,
    d. Tous-les-2-chemins,
    e. Tous-les-chemins-indépendants (avec le calcul de $V(G)$)

## Exercice 4

Soit le programme simple suivant :

```
int foo4(int x, int y, int z) {
    int r;
    if (x > 0 & (y > 0 | z < 0)) {
        r = 1;
    }
    else {
        r = -1;
    }
    return r;
}
```

### Questions :

Établir le graphe de contrôle et produire des données de test permettant de satisfaire les critères de couverture suivants sur le graphe :

1. tous-les-nœuds
2. tous-les-arcs
3. toutes-les-conditions
4. toutes-les-conditions/décisions
5. toutes-les-conditions/décisions modifiées
6. toutes-les-conditions multiples.

A chaque fois, essayez de couvrir le critère demandé sans couvrir de critère supérieur dans la hiérarchie (si c'est possible).

Conseil : établir la table de vérité du prédicat constituant la décision.

## Exercice 5

```
#define EOF -1
#define YES 1
#define NO 0
main() {
    int c, nl, nw, nc, inword;
    inword = NO;
    nl = nw = nc = 0;
    c = getchar();
    while (c != EOF) {
        nc = nc + 1;
        if (c == '\n')
            nl = nl + 1;
        if ((c == ' ' || c == '\n' || c == '\t'))
            inword = NO;
        else
            if (inword == NO) {
                inword = YES;
                nw = nw + 1;
            }
        c = getchar();
    }
    printf("%d\n", nl);
    printf("%d\n", nw);
    printf("%d\n", nc);
}
```

### Questions :

1. Établir le graphe de contrôle de ce programme
2. Calculer les chemins et les données de test pour satisfaire :
    a. TER1
    b. TER2
    c. TER3

## Exercice 6

Reprendre le programme `foo2` de l'exercice 3.

### Questions :

1. Annoter le graphe de contrôle précédent par les indications du flot de données,
2. Calculer les chemins et les données de test pour couvrir les critères
    a. Toutes-les-définitions
    b. Toutes-les-utilisations
    c. Tous-les-DU-chemins

## Exercice 7 - Sujet d'examen 2012-2013

```
public boolean reconocer(String s) {
    boolean p = true;
    int i;
    char c1, c2;

    if (s.length() > 1) {
        i = 1;
        while (i < s.length()/2 && p) {
            c1 = s.charAt(s.length()-i);
            c2 = s.charAt(i-1);
            if (c1 != c2) {
                p = false;
            }
            i++;
        }
    }
    return p;
}
```

### Questions :

1. Donner le graphe de flot de contrôle associé au code de la fonction `reconocer(String)`, en décomposant les conditions multiples, et donner l'expression des chemins.

2. Générer les données de test pour vérifier le critère *tous-les-nœuds* (chemins + données de test).
   Avez-vous trouvé une erreur dans le code ? Si oui, la corriger en donnant les modifications apportées à l'algorithme et au graphe de contrôle, puis générer les données de test pour vérifier *tous-les-nœuds* sur le graphe corrigé.

3. Générer les données de test pour vérifier le critère *tous-les-arcs* (chemins + données de test).
   Avez-vous trouvé une erreur dans le code ? Si oui, la corriger en donnant les modifications apportées à l'algorithme et au graphe de contrôle, puis générer les données de test pour vérifier *tous-les-arcs* sur le graphe corrigé.

4. Générer les données de test pour vérifier le critère *tous-les-chemins-indépendants* (chemins + données de test).
   Avez-vous trouvé une erreur dans le code ? Si oui, la corriger en donnant les modifications apportées à l'algorithme et au graphe de contrôle, puis générer les données de test pour vérifier *tous-les-chemins-indépendants* sur le graphe corrigé.

5. Que calcule la fonction `reconocer` ?