TD
Notion de graphe de contrôle

Exercice 1
Soit le fragment de code suivant :

```
1: a := a * a;
2: x := a + b;
3: y := b;
4: if x = 0 then
5:    z:=0 ;
6: else
7:    z:=1 ;
8: end if
9: if y > 0 then
10:   x:=x+y ;
11: else
12:   x:=y-x;
13: end if
14: print z,x ;
```

Réaliser une exécution symbolique de ce fragment de code.

Exercice 2
Soit le fragment de code suivant :

```
1: Procedure mystere(A,B,X:INTEGER)
2: if  (A > 1 and B = 0)  then
3:    X := X / A;
4: end if
5: if (A = 2 or X > 1) then
6:    X := X + 1;
7: end if
8: return X;
```

- Donner le graphe de contrôle de la procédure mystere.
- Quel est le nombre de chemins ?
- Donner un ensemble de données d'entrée permettant de couvrir toutes les instructions.
- Donner un ensemble de données d'entrée permettant de couvrir toutes les conditions.
- Donner un ensemble de données d'entrée permettant de couvrir tous les chemins exécutables.

Exercice 3
Soit le fragment de code suivant :

```
1: PROGRAM P
2: VAR A, B, C, X, Y, Z : INTEGER
3: BEGIN
4: READ(C) ;
5: X := 7 ;
6: Y := X+A ;
7: B := X + Y +A ;
8: C := Y + 2*X + Z;
9: RETURN (C);
10: END
```

- Construire le graphe de contrôle associé à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.
- Introduire le flot de donnée et étudier la validité du programme.

Exercice 4

On considère le programme suivant, qui calcule X^N pour $N \geq 0$.

On veut générer des tests pour ce programme en utilisant un critère de couverture sur le graphe de flot de contrôle.

```
int puissance(int X, int N)
int S = 1;
int P = N;
while P ≥ 1 do
  if P mod 2 !=0 then
    P = P - 1;
    S = S * X;
  end if
  S = S * S;
  P = P/2;
end while
return S;
```

- Construire le graphe de flot de contrôle de ce programme.
- Sélectionner un ensemble de chemins pour satisfaire le critère tous les arcs .
- Sélectionner un ensemble de chemins pour satisfaire le critère toutes les chemins de longueur au plus k ,où la longueur d'un chemin est comptée en nombre de nœuds. Choisir k de façon sèlectionner les chemins passant au plus deux fois dans la boucle.
- Pour trois des chemins trouvés à la question précédente, calculer par exécution symbolique les conditions de chemin associées.
- Donner des tests concrets pour chacun des cas de test obtenus.

Exercice 5

Soit la procédure P suivante :

```
1: ProcedureP(C1, C2 : BOOL)
2: BEGIN
3: if C1 then
4:    WRITE(C1) ;
5: end if
6: if C2 then
7:    WRITE(C2) ;
8: end if
9: END
```

- Construire le graphe de contrôle associée à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.
- Donner les jeux minimaux de valeurs permettant de couvrir :
    1. les instructions
    2. les conditions
    3. les branches
    4. les chemins

Exercice 6
Soit la procédure suivante :

```
1: PROCEDURE P (C1, C2: BOOL)
2: BEGIN
3: if C1 OR C2 then
4:    WRITE("C1 C2")
5: end if
6: END
```

- Construire le graphe de contrôle associé à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.
- Donner les jeux minimaux de valeurs permettant de couvrir :
    1. les instructions.
    2. les conditions (analyser les jeux de tests (T, F) et (F, T)).
    3. les branches.
    4. les chemins.

Exercice 7
Soit l'algorithme d'Euclide qui calcule le pgcd (plus grand commun diviseur) de deux nombres :

```
1: BEGIN
2: READ(x) ;
3: READ(y) ;
4: while not(x=y) do
5:    if x>y then
6:       x:=x-y ;
7:    else
8:       y:=y-x ;
9:    end if
10: end while
11: pgcd:=x ;
12: END
```

- Construire le graphe de contrôle associé à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.
- Donner les jeux minimaux de valeurs permettant de couvrir :
  1. les instructions.
  2. les conditions.
  3. les branches.
  4. les chemins.
  5. les 2-chemins.

Exercice 8
Soit le programme suivant :

```
1: BEGIN
2: READ(x) ;
3: READ(y) ;
4: z:=0 ;
5: signe:=1 ;
6: if x<0 then
7:    signe:=-1 ;
8:    x:=-x ;
9: end if
10: if y<0 then
11:    signe:=-1 ;
12:    y:=-y ;
13: end if
14: while x≥y do
15:    x:=x-y ;
16:    z:=z+1 ;
17: end while
18: z:=signe*z;
19: END
```

- Déterminer les entrées et les sorties du programme.
- Construire le graphe de contrôle associé à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.

- Donner les jeux minimaux de test permettant de couvrir :
    1. les instructions.
    2. les conditions.
    3. les branches.
    4. les 2-chemins.

Exercice 9
Soit le programme suivant :

```
1: Procedure search(desiredelement : ELEMENT, table: ELEMENT[], numberofitems: INTEGER)
2: VAR
3: found: boolean;
4: counter: integer ;
5: BEGIN
6: found:=false ;
7: if numberofitems >0 then
8:    counter:=1 ;
9:    while (not found) and (counter< numberofitems) do
10:      if table(counter)=desiredelement then
11:         found:=true ;
12:         counter:=counter+1 ;
13:      end if
14:   end while
15: end if
16: if found then
17:   write("the desired element exists") ;
18: else
19:   write("the desired element does not exist") ;
20: end if
21: END
```

- Déterminer les entrées et les sorties du programme.
- Construire le graphe de contrôle associé à cette procédure.
- Calculer le nombre cyclomatique de cette procédure.
- Donner les jeux minimaux de test permettant de couvrir :
    1. les instructions.
    2. les conditions.
    3. les branches.
    4. les chemins.