TD de test logiciel

Partie 1 questions QCM

Q1) Quelle affirmation est exacte :

a) L'activité de test nécessite d'être curieux c'est pour cela que c'est préférable de la confier à des personnes sans expérience.
b) L'activité de test nécessite d'être curieux mais demande de l'expérience.
c) L'activité de test ne nécessite qu'un bon sens de la communication.
d) L'activité de test nécessite de prendre position quitte à « grossir » ou extrapoler certaines données.

Q2) La technique des tests aux limites consiste à :

a) Pousser aux limites les équipes de tests en les mettant fortement sous pression.
b) Essayer d'atteindre la limite des fonctions de maturité du logiciel.
c) Faire fonctionner le logiciel aux limites de ses spécifications.
d) Faire fonctionner le logiciel le plus longtemps possible.

Q3) La technique de partitionnement en classes d'équivalence :

a) Consiste à trouver des domaines sur lesquels le logiciel se comporte de façon homogène.
b) Consiste à trouver des logiciels équivalant au logiciel à tester et à réutiliser les jeux de tests qui ont été faits pour ce logiciel.
c) Consiste à diviser l'équipe de test en groupe de tailles et d'expériences équivalentes

Partie 2 test boite noire

Exercice 1

Supposons que la donnée à l'entrée est un entier naturel qui doit contenir cinq chiffres, donc un nombre compris entre 10 000 et 99 999.

1. Donnez les classes d'équivalence
2. Proposer un jeu de données pour tester cette fonction (proposer également l'oracle)
3. Proposer un jeu de donnée pour tester cette fonction aux limites

Correction

1. Le partitionnement en classes d'équivalence identifierait alors les trois classes suivantes (trois conditions possibles pour l'entrée) :

a. N < 10000

b. 10000 <= N <= 99999

c. N >= 100000

2. On va alors choisir des cas de test représentatif de chaque classe, par exemple, au milieu et aux frontières (cas limites) de chacune des classes :

a. 0, 5000, 9999

b. 10000, 50000, 99999

c. 100000, 100001, 200000

Exercice 2

Soit un programme calculant la valeur absolue d'un entier relatif saisi sous forme d'une chaîne de caractères au clavier

1. Donnez les classes d'équivalence
2. Proposer un jeu de données pour tester cette fonction (proposer également l'oracle)
3. Proposer un jeu de donnée pour tester cette fonction aux limites

Correction

Classe 1
Entrée invalide, chaîne vide

Classe 2
Entrée invalide, chaîne avec plusieurs mots « 123 983 321 »

Classe 3
Entrée invalide, un seul mot mais pas un entier relatif « 123.az+23 »

Classe 4
Entrée valide, un mot représentant un entier relatif positif ou nul « 673.24»

Classe 5
Entrée valide, un mot représentant un entier relatif négatif « -2.4 »

Exercice *3*

Supposons que nous élaborions un compilateur pour le langage X. Un extrait des spécifications précis :

« L'instruction FOR n'accepte qu'un seul paramètre en tant que variable auxiliaire. Son nom ne doit pas dépasser deux caractères non blancs ; Après le signe = on doit préciser aussi une borne ssupérieure et une borne inférieure. Les bornes sont des entiers positifs et on place entre eux le mot-clé TO. »

1. Donnez les classes d'équivalence
2. Proposer un jeu de données pour tester cette fonction (proposer également l'oracle)

Correction

FOR A=1 TO 10	cas nominal
FOR A=10 TO 10	égalité des bornes
FOR AA=2 TO 7	deux caractères pour la variable
FOR A, B=1 TO 8	Erreur - deux variables
FOR ABC=1 TO 10	Erreur - trois caractères pour la variable
FOR I=10 TO 5	Erreur - Borne sup < Borne inf
FOR =1 TO 5	Erreur - variable manquante
FOR I=0.5 TO 2	Erreur - Borne inf décimale
FOR I=1 TO 10.5	Erreur - Borne sup décimale
FOR I=7 10	Erreur - TO manquant

Exercice 4 (Analyse partitionnelle).

1. Donner des classes d´équivalences pour les domaines suivants :

– nombre de stylos

– nom de planète

– tableau de 10 entiers

Donner des entrées pour le test aux limites, préciser les cas de base.

2. Proposer des jeux de données pour tester aux limites ces différentes classe

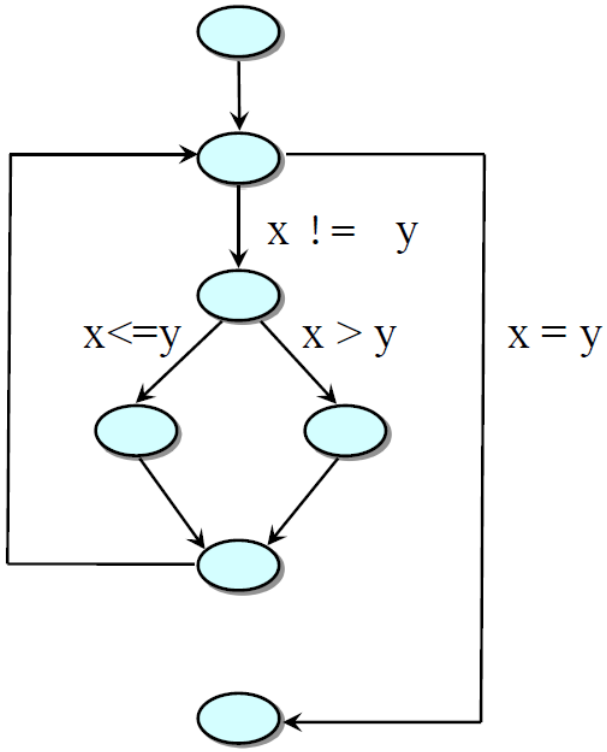
Partie 3 test boite blanche

Exercice 1

```
begin
read(x); read(y);
while (not(x = y)) loop
if x > y then
x := x - y;
else
y := y - x;
end if;
end loop;
pgcd := x;
end;
```

1. Établir le graphe de flot de contrôle de ce programme
2. Fournir l'expression des chemins

```
read(x);
read(y);
while x != y loop
   if x>y then
        x := x - y;
   else
        y := y - x;
   end if;
end loop;
gcd := x;
```

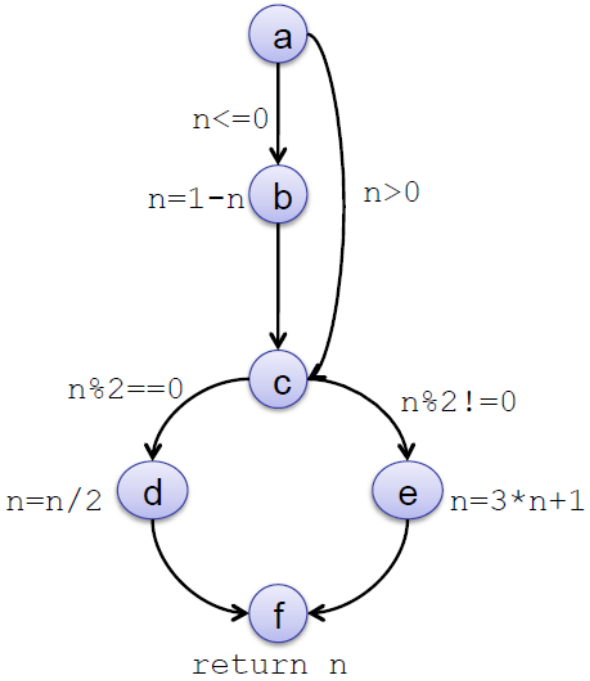


Exercice 2

```
int f(int n){
if (n<=0)
n = 1-n;
if (n%2==0)
n = n/2;
else
n = 3*n+1;
return n;
}
```

1. Établir le graphe de flot de contrôle de ce programme
2. Fournir l'expression des chemins

```
int f(int n){
  if (n<=0)
    n = 1-n;
  if (n%2==0)
    n = n/2;
  else
    n = 3*n+1;
  return n;
}
```



1. Etablir le graphe de flot de contrôle de ce programme
2. Fournir l' expression des chemins

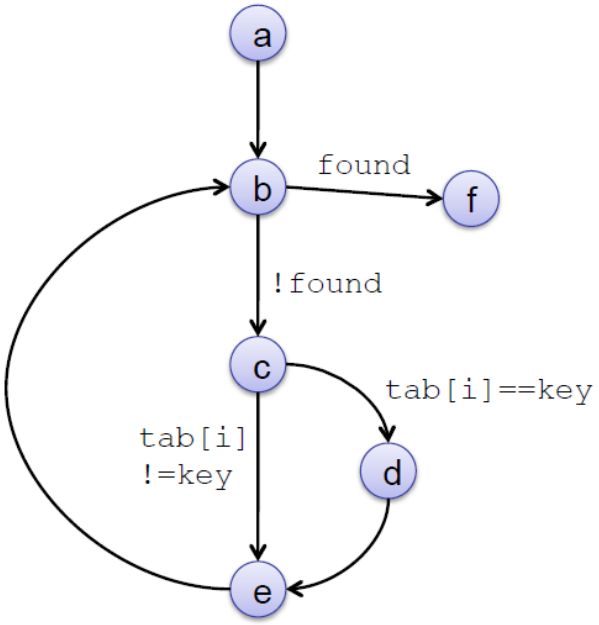
a (1 + b) c (e + d) f

Execice 3

```
int f(int* tab, int key){
int i = 0;
int res;
bool found = false;
while(!found){
if(tab[i]==key){
found=true;
res=i;
}
i = i+1;
}
return res;
}
```

1. Établir le graphe de flot de contrôle de ce programme
2. Fournir l'expression des chemins
3. Ecrire les chemins sous forme algébrique

```
int f(int* tab, int key){
  int i = 0;          a
  int res;
  bool found = false;
  while(!found){
    if(tab[i]==key){
      found=true;     d
      res=i;
    }
    i = i+1;          e
  }
  return res;         f
}
```



1. Etablir le graphe de flot de contrôle de ce programme
2. Fournir l' expression des chemins

ab (c (1 + d) eb)* f

Exercice 4

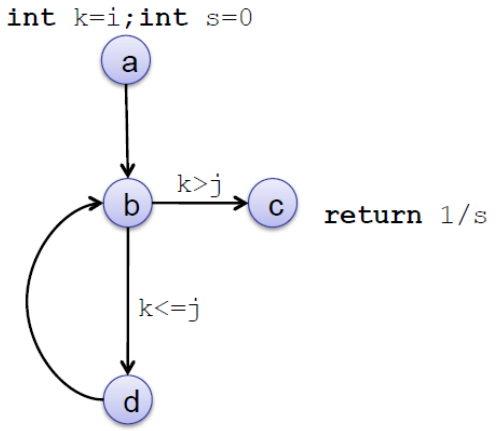
```
int somme(int* t, int i, int j)
{
int k = i;
int s=0;
while(k<=j){
s+=t[k];
k++;
}
return 1/s;
}
```

1. Construire le graphe de flot contrôle
2. Fournir l'expression de 2 chemins
3. Proposer qui permet de couvrir tous les nœuds

Calcul de l'inverse de la somme des éléments d'un tableau entre les indices **i** et **j**

```
int somme(int* t, int i, int j)
{
   int k = i;
   int s=0;
   while(k<=j){
        s+=t[k];
        k++;
   }
   return 1/s;
}
```

Taux de couverture =
nb d'arcs couverts/nb total d'arcs



La DT = {t = {1,2,3},i=0;j=2} permet de couvrir tous les arcs

Or, c'est la DT = {i=1,j=0} qui met le programme en erreur

Exercice 5

```
void f( int value[], int total_input, int total_valid,
int sum, double average)
int i=0;
total_input=0;
total_valid=0;
sum=0;
while(value[i]!=-999 && total_input<100)
{
total_input++;
```

```
if(value[i]>=min && value[i]<=max)
{
total_valid++;
sum+=value[i];
}
i++;
}
if (total_valid>0)
average=sum/total_valid;
else
average = -999;
```

Exercice 6

```
int f(int n){
if (n<=0)
n = 1-n;
if (n%2==0)
n = n/2;
else
n = 3*n+1;
return n;
}
```

1. Construire le graphe de flot contrôle
2. Donner des jeux de données pour tous les nœuds,
3. Donner des jeux de données pour tous les arcs,
4. Donner des jeux de données pour tous les chemins indépendants

Exercice 7

```
int a=0, b=0, p=0;
read(a, b)
if (a<0)
b = b-a;
if (b%2==0)
p = b*b;
else
p = 2*a;
println p;
```

1. Fournir le graphe de flot de contrôle

2. Donner des jeux de données pour tous les nœuds,
3. Donner des jeux de données pour tous les arcs,
4. Donner des jeux de données pour tous les chemins indépendants