La Rochelle Université
Licence Informatique
UE Génie logiciel 2

TD 3 - Test boite blanche et test boite noire

Version enseignant

1 Graphe de contrôle et analyse des chemins : Test boite blanche

Vous disposez, en annexe, d'un exemple permettant de faire cet exercice.

- A partir du code ci-dessous, donnez le graphe de contrôle de la fonction *prime*;
- Donnez le nombre de McCabe pour ce graphe de contrôle;
- Donnez la liste des chemins du graphe de contrôle;
- Donnez un jeu de test pour couvrir tous les chemins possibles;
- Donnez un jeu de test pour couvrir toutes les instructions.

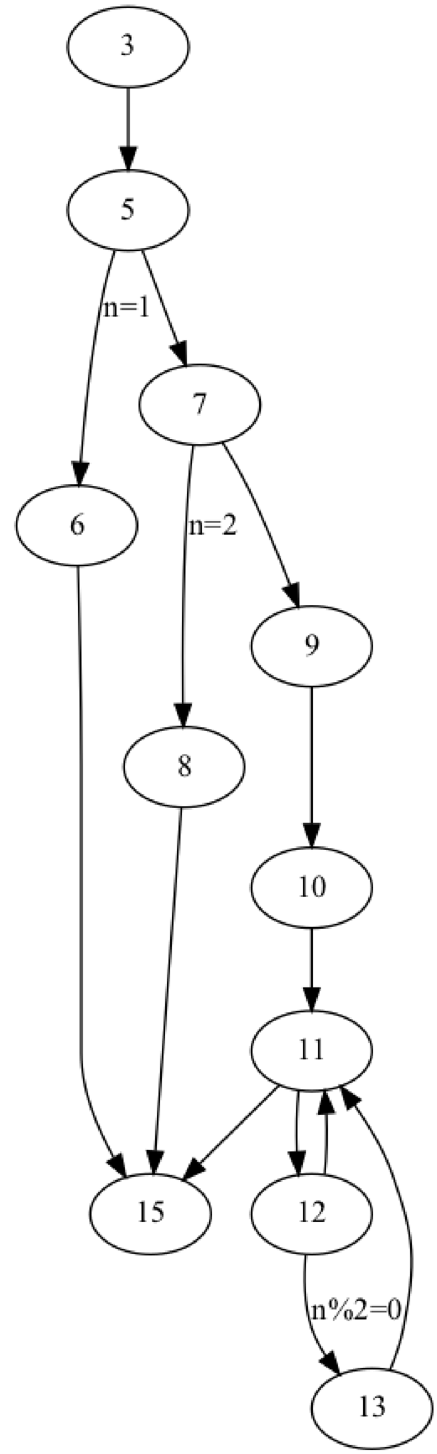
```
public boolean prime(int n)
{
   boolean prime;

   if(n==1)
      prime = false;
   else if(n==2)
      prime = true;
   else {
      prime = true;
      for(int i = 2; i < n; i++)
         if(n % i == 0)
            prime = false;
   }
   return prime;
}
```

Nombre de MacCabe = nb arcs – nb nœuds + 2 = 14 - 11 + 2 = 5

Liste des chemins :
I. 3 - 5 - 6 - 15
II. 3 - 5 - 7 - 8 - 15
III. 3 - 5 - 7 - 9 - 10 - 11 - 14 - 15
IV. 3 - 5 - 7 - 9 - 10 - 11 - 12 - 11 - 14 - 15
V. 3 - 5 - 7 - 9 - 10 - 11 - 12 - 13 - 11 - 14 - 15

Jeu de test pour couvrir tous les chemins :
I. n=1
II. n=2
III. n=0
IV. n=3
V. n=4

Jeu de test pour couvrir toutes les instructions :
Avec le chemin V, on couvre presque tout (n=4).
Les nœuds 6 et 8 restent à couvrir, on ajoute le chemin I (n=1) et II (n=2).



2 Test combinatoire : Test boite noire

Un site internet gère les inscriptions des participants à une course à pieds amateurs. Le montant de l'inscription réglé par chaque coureur dépend d'un certain nombre de facteurs présentés sur le prototype d'interface de la figure 1.

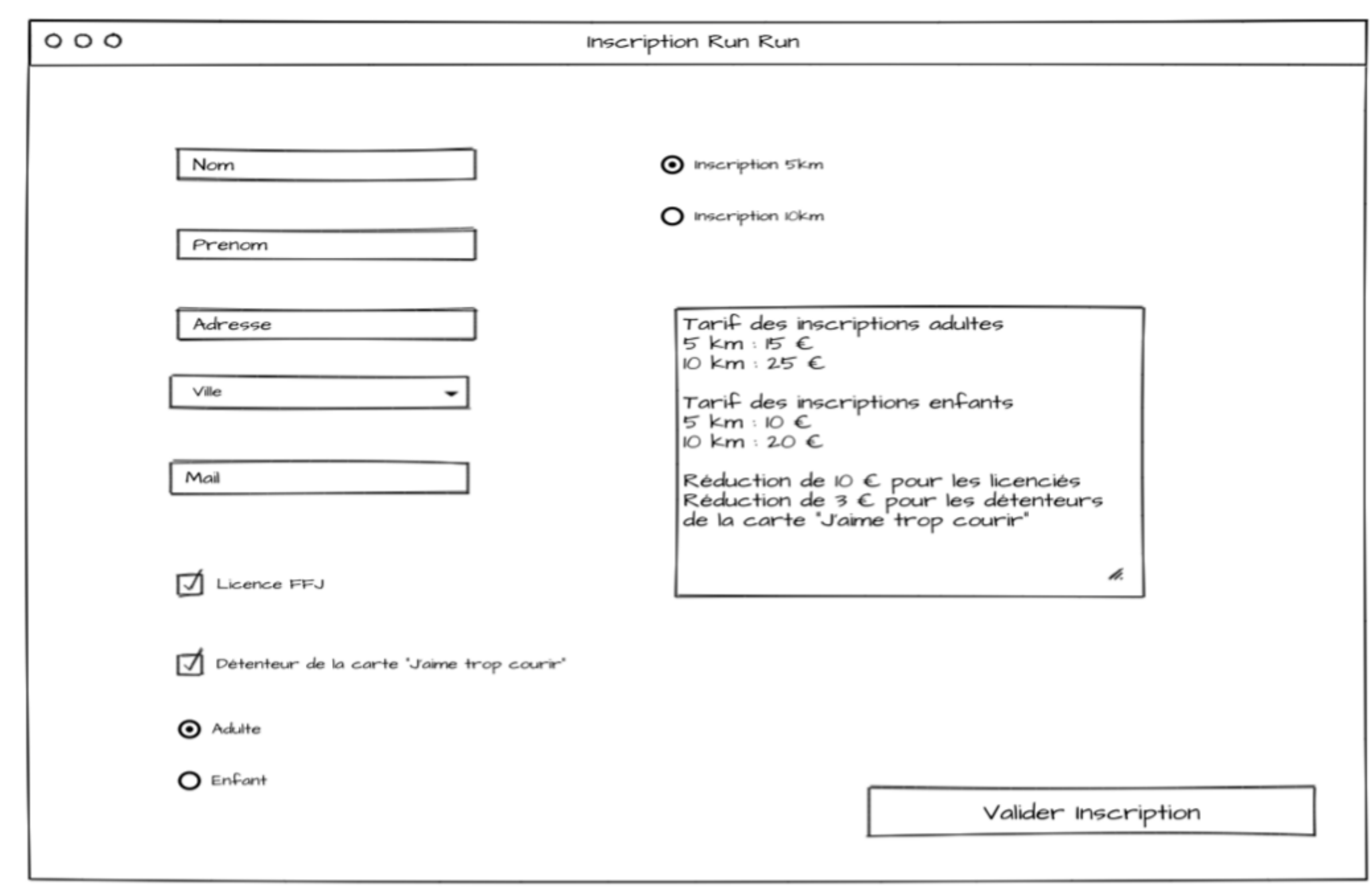


FIGURE 1 – Interface du site internet de la course

Vous disposez, en annexe, d'un exemple permettant de faire cet exercice.

— En utilisant une approche de test combinatoire, proposez sous forme tabulaire l'ensemble des données de test permettant de calculer le montant de l'inscription lorsque l'utilisateur a rempli son formulaire;
— Combien de tests doivent être exécutés avec une approche purement combinatoire?
— Utilisez une approche Pairwise pour réduire cet ensemble de tests et donnez le sous-ensemble de tests obtenus.

	A Licence FFJ	B Carte	C Adulte	D 10km
DT1	0	0	0	0
DT2	0	0	0	1
DT3	0	0	1	0
DT4	0	0	1	1
DT5	0	1	0	0
DT6	0	1	0	1
DT7	0	1	1	0
DT8	0	1	1	1
DT9	1	0	0	0
DT10	1	0	0	1
DT11	1	0	1	0
DT12	1	0	1	1
DT13	1	1	0	0
DT14	1	1	0	1
DT15	1	1	1	0
DT16	1	1	1	1

FIGURE 2 – Table de vérité de l'exercice 2

Approche Pairwise

Le nombre de paires de variables est 6 : (A, B), (A, C), (A, D), (B, C), (B, D) et (C, D).

Le nombre de paires de valeurs est 24 (nombre de paires de variables x 4). Par exemple, (A=0, B=0) indique que le coureur n'est pas licencié et qu'il n'a pas de carte.

Nous obtenons donc l'ensemble de paires de valeurs suivant :

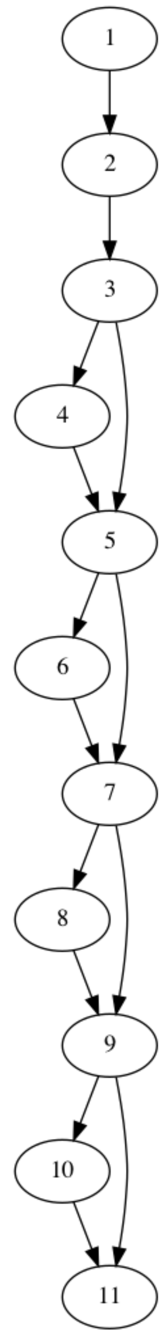
- (A=0, B=0), (A=0, B=1), (A=1, B=0), (A=1, B=1)
- (A=0, C=0), (A=0, C=1), (A=1, C=0), (A=1, C=1)
- (A=0, D=0), (A=0, D=1), (A=1, D=0), (A=1, D=1)
- (B=0, C=0), (B=0, C=1), (B=1, C=0), (B=1, C=1)
- (B=0, D=0), (B=0, D=1), (B=1, D=0), (B=1, D=1)
- (C=0, D=0), (C=0, D=1), (C=1, D=0), (C=1, D=1)

Ici, 5 tests pour tout couvrir :

- (0,0,1,1), (0,1,0,1), (0,0,0,0), (1,1,1,0) et (1,0,0,1)

Annexe 1 : Exercice 1

```
1  read a,b,c
2  type = "scalene"
3  if(a == b || b == c || a == c)
4     type = "isocele"
5  if(a == b && b == c)
6     type = "equilateral"
7  if(a >= b+c || b > a+c || c >= a+b)
8     type = "non triangle"
9  if(a <= 0 || b <= 0 || c <=0)
10    type = "entrees invalides"
11 print type
```



Nombre de MacCabe (également appelé nombre cyclomatique) :
nb arcs – nb nœuds + 2
14 – 11 + 2 = 5

Nombre de chemins = nombre de MacCabe = 5

Liste des chemins :
I. 1 – 2 – 3 – 5 – 7 – 9 – 11
II. 1 – 2 – 3 – 4 – 5 – 7 – 9 – 11
III. 1 – 2 – 3 – 5 – 6 – 7 – 9 – 11
IV. 1 – 2 – 3 – 5 – 7 – 8 – 9 – 11
V. 1 – 2 – 3 – 5 – 7 – 9 – 10 – 11

Liste des tests (a, b, c) :
I. (5, 3, 4)
II. (5, 5, 6)
III. (5, 5, 5)
IV. (10, 2, 2)
V. (-1, 2, 2)

Annexe 2 : Exercice 2

La technique des tests combinatoires consiste à associer des booléens aux entrées du système et à procéder au test exhaustif de toutes les combinaisons de valeurs. Les entrées numériques du système peuvent être traduites en booléens selon les critères de décision.

Prenons en exemple un magasin de jouets qui vend des petits trains. Les clients peuvent personnaliser leur petit train via une application disponible sur une borne à l'entrée du magasin. Le client pourra :

- choisir le matériau du train : fer ou bois ;
- choisir la couleur du train : gris ou vert ;
- choisir d'ajouter un logo sur le dessus du train : logo de la SNCF, logo TGV.

En analysant cette spécification, nous pouvons en déduire 3 paramètres avec chacun 2 valeurs :

- Matériau M : fer (0), bois (1) ;
- Couleur C : gris (0), vert (1) ;
- Logo L : SNCF (0), TGV (1).

Nous pouvons construire la table de vérité suivante :

	M	C	L	Résultat attendu	Résultat observé
DT1	0	0	0		
DT2	0	0	1		
DT3	0	1	0		
DT4	0	1	1		
DT5	1	0	0		
DT6	1	0	1		
DT7	1	1	0		
DT8	1	1	1		

FIGURE 3 – Table de vérité de l'exemple du magasin de jouets

Approche PAIRWISE : Réduction du nombre de combinaisons à tester
Le nombre de paires de variables est 3 : (M, C), (M, L) et (C, L).
Le nombre de paires de valeurs est 12 (nombre de paires de variables x 4).
Par exemple, (M=0, C=0) indique un choix du fer pour le matériau M et gris pour la couleur C.

Nous obtenons donc l'ensemble de paires de valeurs suivant :

- (M=0, C=0), (M=0, C=1), (M=1, C=0), (M=1, C=1)
- (M=0, L=0), (M=0, L=1), (M=1, L=0), (M=1, L=1)
- (C=0, L=0), (C=0, L=1), (C=1, L=0), (C=1, L=1)

Ici, 4 tests pour presque tout couvrir :

- (0,0,1), (0,1,0), (1,0,1), (1,1,0)

Avec l'ajout de ces deux tests, tout est couvert :

- (1,0,0) et (1,1,1)