TD 4 - Test fonctionnel et structurel

Exercice 1

L'opération middle prend en entrée trois entiers différents deux à deux et renvoie l'entier parmi les trois qui n'est ni le plus grand ni le plus petit.

1)

Pré-conditions :

- $A \neq B$
- $C \neq B$
- $A \neq C$

Les trois entiers doivent être distincts.

Post-conditions :

L'entier retourné m est celui qui satisfait :

- $A \leq m \leq B$ ou $B \leq m \leq A$
- $A \leq m \leq C$ ou $C \leq m \leq A$
- $C \leq m \leq B$ ou $B \leq m \leq C$

Autrement dit, m est celui qui n'est ni le plus grand, ni le plus petit parmi A, B, et C.

2)

La forme normale disjonctive consiste à lister les conditions sous lesquelles chacun des trois nombre est le milieu en couvrant les cas où chaque nombre est le plus grand, le plus petit, ou entre les deux :

- A est le milieu si : $(A > B \land A < C) \lor (A > C \land A < B)$
- B est le milieu si : $(B > A \land B < C) \lor (B > C \land B < A)$
- C est le milieu si : $(C > A \land C < B) \lor (C > B \land C < A)$

3)

Pour chaque cas, nous allons choisir des valeurs d'entiers qui couvrent chaque disjonction :

Cas 1 : A est le milieu

- Test 1.1 : a=2, b=1, c=3 (milieu : 2)
- Test 1.2 : a=4, b=6, c=5 (milieu : 4)

Cas 2 : B est le milieu

- Test 2.1 : a=1, b=2, c=3 (milieu : 2)
- Test 2.2 : a=5, b=6, c=4 (milieu : 6)

Cas 3 : C est le milieu

- Test 3.1 : a=3, b=1, c=2 (milieu : 2)
- Test 3.2 : a=6, b=5, c=4 (milieu : 4)

Ces cas de test couvrent tous les scénarios possibles.

Exercice 2

On considère le programme suivant, qui calcule XN pour N ≥ 0.

```
int puissance(int X, int N) {
    int S = 1;
    int P = N;
    while(P >= 1) {
        if(P mod 2 != 0) {
            P = P - 1;
            S = S * X;
        }
        S = S * S;
        P = P / 2;
    }
    return S;
}
```

1)

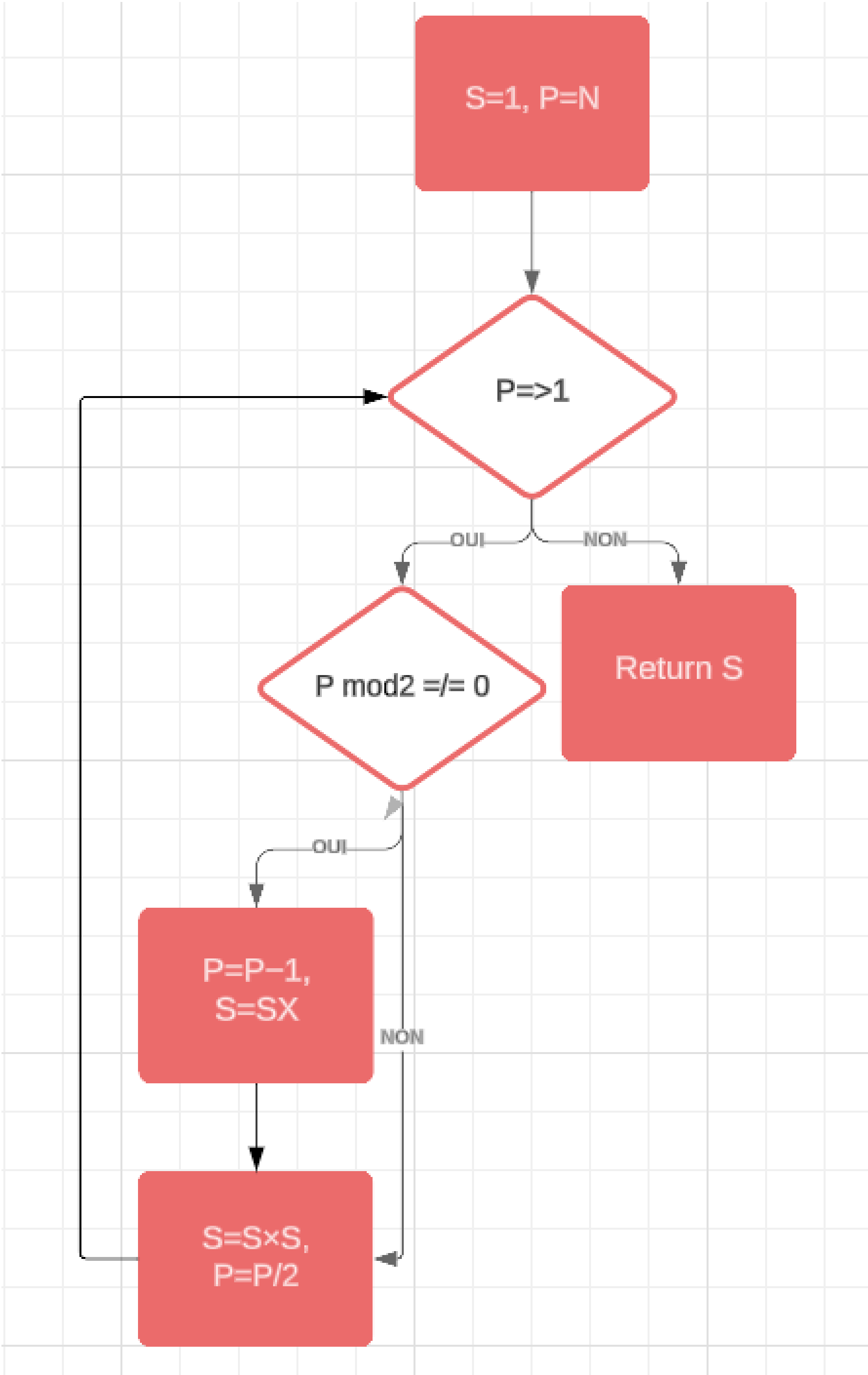
Pré-conditions :

- N≥0 : N doit être un entier positif ou nul.
- X est un entier quelconque.

Post-conditions :

- S= X^N : Le résultat est la puissance X élevée à N.

2)



3)

Pour satisfaire ce critère, il faut que chaque instruction du programme soit exécutée au moins une fois. Voici les chemins minimaux et les conditions associées pour chaque chemin :

- **Chemin 1** : 1→2→6 (exécution sans entrer dans la boucle)
    - Condition : N = 0
    - Test concret : X = 5, N = 0 (retourne 1)
- **Chemin 2** : 1→2→3→5→2→6 (exécution de la boucle sans passer par l'instruction P=P−1)
    - Condition : N=2 (pair)
    - Test concret : X=3, N=2 (retourne 9)
- **Chemin 3** : 1→2→3→4→5→2→6 (exécution de la boucle avec P mod 2≠0
    - Condition : N=3 (impair)
    - Test concret : X=2,N=3 (retourne 8)

4)

Pour satisfaire le critère "toutes les décisions", il faut que toutes les branches des instructions conditionnelles soient testé. Nous avons déjà couvert plusieurs cas dans la question précédente, mais nous devons ajouter les chemins où les branches non exécutées précédemment le sont.

Les tests sont similaires à ceux du critère précédent, mais doivent couvrir tout les possibilité de chaque décision.

5)

Pour satisfaire ce critère, il faut inclure les chemins qui exécutent la boucle une ou deux fois. Certains chemins sont cependant infaisables. Par exemple, pour des valeurs de N, la boucle sera exécutée un nombre limité de fois, donc un chemin pourrait être théoriquement possible, mais ne correspondrait à aucune condition réalisable en pratique.

- **Chemin 4** : 1→2→3→4→5→2→3→5→2→6