

TD 4 - Spécification et tests fonctionnels

Semaine du 24 octobre 2020

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. Donnez une spécification formelle de cette opération.
2. Donnez, par construction DNF, un ensemble de tests pour l'opération `middle`.

Exercice 2 (Test fonctionnel : Révision)

On reprend l'exercice 2 du TD3.

Une société vend deux produits A et B au prix unitaire de 5 EUR pour A et de 10 EUR pour B. Une commande comprend une certaine quantité du produit A et une certaine quantité du produit B. Le coût d'une commande est la somme totale des prix unitaires des produits commandés, à laquelle on applique une réduction selon les règles suivantes :

- Si la somme totale est supérieure ou égale à 200 EUR, on applique une réduction de 5%, si elle est supérieure ou égale à 1000 EUR, la réduction est de 20%. Ces deux réductions ne sont pas cumulables et portent sur la somme totale.
- La société souhaitant encourager la vente de A, on applique, sur le prix obtenu grâce à la règle précédente, une réduction supplémentaire de 10% si la commande comprend au moins 45 produits A.

1. Donnez une spécification formelle de la fonction `calc_prix(A,B:int):Rat`.
2. Donnez, par construction DNF, un ensemble de tests pour le calcul du coût d'une commande.
3. Complétez votre jeu de tests par une analyse aux limites.

Exercice 3 (Test boîte noire)

On veut tester l'implémentation d'une fonction `magic_sort` qui manipule des listes. Nous n'avons pas à disposition le code source de cette fonction `magic_sort`.

On spécifie d'abord la notion de tri comme suivant :

```

definition insert :: int => int list => int
  where insert a [] = [a]
        — insert a (b#S) = (if a <= b then a#(b#S) else b#(insert a S))

```

Cela nous permet de définir le contrat `magic_sort(S)`, pour un k prédéfini :

```

definition pre_magic_sort(S) ≡ length(S) = k
definition post_magic_sort(S, res) ≡ (sort S = res)

```

On note que `#` et le constructeur d'une liste, et que la notation `[a, b, c]` est donc une abbréviation de `a#(b#(c#[]))`.

Questions

1. Dépliez, puis simplifiez jusqu'à la DNF l'application de *magic_sort* sur une liste contenant deux entiers : $pre_{magic_sort}([a, b]) \wedge post_{magic_sort}([a, b], res)$.
2. Construisez un jeu de tests concret couvrant la DNF. Ajouter les cas limites.
3. Combien y a-t-il de cas à couvrir dans le cas $k = 3$? (Cas concrets et cas limites)