TD15 : Tests

1 Tests fonctionnels

Question 1 : On dispose d'une fonction

```
/** entrée : 3 entiers
 * sortie : le plus grand des trois entiers */
int max(int a, int b, int c);
```

Proposer des tests fonctionnels pour cette fonction en utilisant la méthode du partitionnement du domaine d'entrée.

Question 2 : On dispose d'une fonction

```
/** entrée : un tableau d'entier et sa longueur
 * sortie : la somme des éléments du tableau */
int somme(int *tab, int n);
```

Proposer des tests fonctionnels pour cette fonction en utilisant la méthode du partitionnement du domaine d'entrée.

Question 3 : On dispose d'une fonction

```
/** entrée : tableau tab d'entiers de longueur strictement positive dont la
    première
 * case contient le nombre de cases -1 (ainsi les cases qui suivent ont des indices
 * allant de 1 à tab[0]).
 * sortie : nouveau tableau avec les éléments dans l'ordre inverse (sauf celui de
    la
 * case 0 qui doit toujours donner l'information sur le
 * nombre de cases). */
int *retourner(int *tab);
```

Proposer des tests fonctionnels pour cette fonction en utilisant la méthode du partitionnement du domaine d'entrée.

Question 4 : On dispose de deux fonctions de traduction d'un tableau d'entiers en une liste d'entiers et vice-versa. Proposer des tests fonctionnels pour ces fonctions en utilisant la méthode du partitionnement du domaine d'entrée.

Question 5 : Proposer un jeu de tests fonctionnels pour une fonction qui prend en argument une liste et renvoie une nouvelle liste avec les valeurs dans l'autre sens.

2 Tests structurels

Question 6 : On a le code suivant pour la fonction `max` de la question 1 :

```
1  int max(int a, int b, int c){
2    if (a > b) {
3      if (b > c){
4        return a;
5      } else if (a > c) {
6        return a;
7      } else {
8        return c;
9      }
10   } else {
11     if (b > c) {
12       return b;
13     }
14   }
15 }
```

1. Dessiner le graphe de flot de contrôle de la fonction `max`.
2. Proposer un jeu de tests qui possède une couverture de sommets totale.
3. Proposer un jeu de tests qui possède une couverture d'arcs totale. En profiter pour identifier une erreur dans la fonction et la corriger.

Question 7 : On a maintenant le code suivant pour la fonction `max` de la question 1 :

```
1  int max(int a, int b, int c){
2    int n;
3    if (a > b){
4      n = a;
5    } else {
6      n = b;
7    }
8    if (n > c) {
9      return n;
10   } else {
11     return c;
12   }
13 }
```

1. Dessiner le graphe de flot de contrôle de la fonction `max`.
2. Proposer un jeu de tests qui possède une couverture de sommets totale.
3. Proposer un jeu de tests qui possède une couverture d'arcs totale.

Tableaux et listes d'entiers

On considère un tableau d'entiers dans lequel par convention la première case indique le nombre d'élements qui suivent, et le type

```
struct maillon {
  int val;
  struct maillon *suiv;
};
```

qui permet d'obtenir une liste (en tant que pointeur sur un maillon) dont le premier maillon serait un maillon factice permettant de connaître le nombre d'éléments de la liste (ainsi une liste vide contient un unique maillon dont le champ `val` vaut 0 et le champ `suiv` vaut `NULL`).

Question 8 : On a la fonction

```
1   int *list_to_tab(struct maillon *liste){
2     int *tab;
3     int i = 1;
4
5     assert(liste != NULL);
6     tab  = (int *)malloc(liste->val*sizeof(int));
7     for(; liste != NULL; liste = liste->suiv){
8       tab[i] = liste->val;
9       i = i+1;
10    }
11
12    return tab;
13  }
```

1. Le jeu de tests fonctionnels trouvé question 4 permet-il de trouver des erreurs dans le fonctionnement de cette fonction?
2. Proposer un jeu de tests qui possède une couverture de sommets totale.

3 Un peu de programmation

Question 9 : Écrivez une fonction `tab_to_list` qui passe le jeu de tests fonctionnels proposé en question 4.

Question 10 : Écrire une fonction `reverse_list` qui prend en argument une liste et renvoie une nouvelle liste avec les valeurs dans l'autre sens, qui passe votre jeu de tests fonctionnels de la question 5.