

# TD – Tests d’applications

## Rendu

Le but est de rendre un rapport présentant vos réponses aux questions de ce TD. Vous devez me rendre le rapport et **uniquement** le rapport. Il doit donc présenter **tous les éléments nécessaires** pour démontrer que vous avez bien compris la question et bien répondu à la question (et donc être illustré par des screenshots des éléments importants, par des copier/coller des morceaux de code importants, etc). **Mettez l’accent sur vos explications**, ce sont elles qui importent le plus. Le résultat n’a de valeur qu’en lien avec son côté explicable (comme lors d’un audit réel en fait).

Le rapport doit être dans l’un de ces formats :

- PDF (préféablement)
- DOCX (et surtout PAS DOC)
- ODT

Format du nom de fichier : `td-tests-Nom1-Nom2.pdf`

Envoyez votre fichier à l’adresse [allan.blanchard@cea.fr](mailto:allan.blanchard@cea.fr) au plus tard le **07/03** avec comme objet :

[INSA CVL] Test

## 1 Prise en main de `utest.h` et `gcov(r)`

Le dossier contenant les codes du TD se décompose comme suit :

- un Makefile permettant de lancer (entre autres) les cibles :
  - `tests` qui lance les tests de la base,
  - `coverage` qui construit le HTML de couverture,
- `utest.h` une bibliothèque *header-only* qui fournit de l’outillage pour le test,
- `tests-base.c` le fichier qui va accueillir nos tests et qui contient déjà quelques tests de la fonction fournie dans
- `example.c/h` qui contient une petite fonction d’exemple.

Les autres fichiers C et H sont ceux que vous devrez tester. Vous n’avez **jamais** besoin de lire ou modifier `utest.h`.

Faites déjà quelques essais.

```
$ make tests
$ make coverage
$ firefox coverage.html
```

Un test dans `utest.h` est de cette forme :

```
UTEST(nom_du_groupe_de_tests, nom_du_test) {
    // code du test
}
```

et contient au moins une opération de la forme ASSERT/EXPECT. Pour ce TD, nous préférons la forme EXPECT qui n'arrête pas l'exécution des tests. La documentation de `utest.h` est disponible à cette adresse : <https://github.com/sheredom/utest.h>.

*Pendant la compilation, le Makefile instrumente également la présence de certains comportements indéterminés pour détecter ce type de problèmes.*

Changez l'un des tests de la suite de test pour qu'il échoue, quels changements constatez-vous sur les sorties obtenues ? Supprimez ensuite le test, quels nouveaux changements constatez-vous (et pourquoi) ? Supprimez en un autre, y a-t-il d'autres différences (et pourquoi) ?

## 2 Valeur absolue

```
int my_abs(int x) {  
    if (x >= 0)  
        return x;  
    return -x;  
}
```

*Pour exécuter une sous partie des tests, l'option `--filter` peut être utilisée. Par exemple, pour exécuter uniquement la suite `x`, nous pouvons lancer :*

```
| $ ./tests-base --filter="x.*"
```

*Après avoir lancé cette sous-suite, on peut lancer l'analyse de couverture à la main :*

```
| $ gcovr --decisions --html-details coverage.html
```

*Attention cependant ! Il faut faire un nettoyage des fichiers d'analyse de couverture avant d'exécuter la base de tests avec :*

```
| $ make clean-coverage
```

1. Donnez une spécification informelle à la fonction valeur absolue.
2. Produisez une suite de test en vous basant sur la spécification que vous avez fourni pour la fonction, si la spécification n'est pas assez précise n'hésitez pas à la retravailler. Justifiez que votre suite couvre tous les cas de spécification identifiés.
3. Que peut-on reprocher à ces suites minimales ? Ajoutez les tests permettant d'éviter cette faiblesse.
4. Le critère utilisé par `gcov` est une couverture par instruction. La couverture de code a-t-elle changé entre la question 2 et la question 3 ? Cette couverture était-elle donc un bon indice sur la qualité de votre suite de test ? Concluez sur la qualité présumée d'une suite de tests qui satisfait le critère "toutes les instructions".

## 3 Indice du minimum

```
/* Finds the minimum of a sequence of len elements */  
size_t index_min(int* seq, size_t len);
```

1. Quels problèmes relevez-vous dans la spécification de la fonction ci-dessus ?
2. Écrivez les tests permettant de mettre en lumière ces problèmes. En particulier, pour chaque test relevant une imprécision de la spécification, indiquez en quoi la spécification d'origine ne permettait pas de créer l'oracle de test. Si vos tests révèlent des problèmes dans la fonction, corrigez-la.

- Proposez une spécification plus précise pour la fonction et montrez en en quoi vos tests recouvrent bien les comportements présentés par la spécification.
- Dans le corps de la fonction, la variable `min` ne sert à rien modifier le code pour vous en débarrasser. Qu'est-ce qui vous donne confiance dans votre modification ?

*Dans la fonction précédente, nous aurions pu introduire des erreurs plus subtiles et beaucoup plus difficile à détecter par du test, nous verrons cela plus tard dans le cours.*

## 4 Recherche de valeur

```
/* Search for the first occurrence of v in a sequence of len
   elements, and returns its index. If v does not belong to the
   sequence, returns len.
*/
size_t search(int v, int* seq, size_t len);
```

- Produisez une suite de tests *functionality-based* nommée `search_functionality` pour cette fonction, expliquez en quoi la suite couvre bien la spécification. Corrigez la fonction si vous rencontrez des bugs.
- Produisez une suite de tests *interface-based* nommée `search_interface`. Y a-t-il des restrictions sur les combinaisons que vous pouvez former ? Pourquoi ?
- Que déduisez-vous à propos de la capacité à automatiser la génération de tests *interface-based* ?

## 5 Ordonner 3 valeurs

```
void order_3_incr(int* a, int* b, int* c);
```

- Produire le graphe de contrôle de cette fonction. Ignorer les cas d'erreur.
- Produire un jeu de tests pour le critère toutes les instructions.
- Votre jeu de test respecte-t-il le critère toutes les conditions ? Justifier votre réponse à l'aide des résultats de couverture.
- Enrichir le jeu de tests pour le critère toutes les conditions.

## 6 Somme des N premiers entiers

```
unsigned sum(unsigned n) {
    unsigned r = 0;
    for (unsigned i = 1; i <= n; i++) {
        r += i;
    }
    return r;
}
```

- Cette fonction a-t-elle une condition particulière pour pouvoir être utilisée ?
- Produire le graphe de contrôle de cette fonction. Ignorer les cas d'erreur.
- Donner un ensemble de chemins permettant de satisfaire le critère toutes les instructions. Produisez la suite de tests correspondante nommée `sum_all_insts`.
- Donner un ensemble de chemins permettant de satisfaire le critère toutes les conditions. Produisez la suite de tests correspondante nommée `sum_all_conds`.
- Donner un ensemble de chemins permettant de satisfaire le critère tous les 2-chemins. Produisez la suite de tests correspondante nommée `sum_all_2_paths`.