Sujet de Travaux Pratiques Tests
Tests des logiciel 2007

1. INTRODUCTION

Ce TP consiste à réaliser les tests de 3 fonctions à l'aide de l'outil Alcotest.

1 - Tester les fonctions de conversion avec une couverture fonctionnelle et une couverture des données aux limites.

2 - Tests de Validation. Tester le module "`Segment`" avec une couverture fonctionnelle et une couverture des données aux limites.

3 - Tests Structurelle : Tester la fonction "`hysteresis`" avec une couverture des branches, et une couverture des données aux limites. Afin de faciliter la vérification de la couverture des branches, la fonction "hysteresis" effectue une trace des branches parcourues.

Résultats attendus :

- La trace d'exécution de l'outil Alcotest commentée des éventuelles anomalies détectées durant les tests.
- A rendre pour le jour de l'examen.
- Groupe de 2 personnes.

2. CONVERSION

2.1. RÔLE

Le composant "HexaToDecimal" effectue la conversion d'un nombre hexadécimal en base 10.

Le composant "OctalToDecimal" effectue la conversion d'un nombre octal en base 10.

2.2. PARAMÈTRES

La valeur Hexadécimal sur un maximum de 7 digits (par exemple: A14 retourne le résultat 2580)
La valeur Octal sur un maximum de 8 digits (par exemple: 147 retourne le résultat 103)

2.3. RETOUR

La valeur en décimal.

2.4. PROTOTYPE

```
void  HexaToDecimal(const char* Hexa, int *res)

void OctalToDecimal(const int octal, int *res)
```

3. MODULE SEGMENT

3.1. RÔLE

Le module "`Segment`" permet de manipuler les segments de droites. Un segment est représenté par les coordonnées sur la droite de ces deux extrémités. Le segment vide est représenté par le couple `[0,0]`. Ce module fournit le type d'un segment (tuple) ainsi qu'un certain nombre de primitives permettant de manipuler ces segments.

3.2. TRAITEMENTS

Les primitives du module segment sont :

- `Get_length`, qui calcule la longueur d'un segment
- `Is_include`, qui retourne un booléen indiquant si le segment_A est inclus dans le segment_B ou non
- `Are_Overlapped`, qui retourne un booléen indiquant si les segments A et B ont au moins un point commun
- `Segment_Equality`, qui réalise l'égalité structurelle entre deux segments.

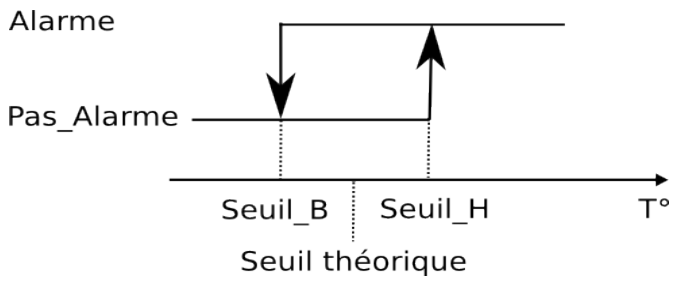
3.3. PROTOTYPES

```
typedef struct {
      int first;
      int second;
}

typedef enum {
      FALSE = 0,
      TRUE = 1
} Bool;

void Get_Length( tuple Segment, int * l);

void Is_include( tuple Segment_A, tuple Segment_B, Bool *res );

void Are_Overlapped( tuple Segment_A, tuple Segment_B, Bool *res);

void Segment_Equality( tuple Segment_A, tuple Segment_B, Bool *
res );
```

4. HYSTERESIS

4.1. RÔLE

Un système de sécurité doit réaliser la surveillance d'un capteur de température. Afin de ne pas avoir de changements intempestifs au niveau du seuil théorique, une hystéresis est appliquée suivant le schéma suivant :



4.2. PARAMÈTRES

La température issue du capteur

4.3. RETOUR

L'état de l'alarme. ALARME (1) ou PAS_ALARME (0).

4.4. PROTOTYPE

```
#define SEUIL_H 42
#define SEUIL_B 40

void hysteresis( int val, int *alarme);
```