

**Exercice 1** Indécidabilité de propriétés.

Utiliser le fait qu'il est indécidable de savoir si un programme P s'arrête sur la donnée D pour montrer que :

1. il est indécidable de montrer qu'une variable x d'un programme prend la valeur v à un point de programme donné.
2. il est indécidable qu'un programme P s'arrête pour toutes les données qu'il reçoit.

Vous supposerez qu'un programme est défini comme une fonction qui prend $n \geq 0$ arguments entiers et renvoie une valeur entière. Les instructions sont de la forme :

```
I ::= x ::= exp
      if b then I1 else I2
      while b do I
      return exp
```

avec b une expression booléenne et exp une expression arithmétique.

Exercice 2 Test d'une fonction simple.

1. Ce programme calcule le n^{ieme} nombre de fibonacci F_n (1,1 pour les F_0, F_1 , ensuite F_n est la somme des deux nombres précédents).

```
int fib(int n) {
  int f = 0;
  int f0 = 1;
  int f1 = 1;
  while (n > 1) {
    n--;
    f = f0 + f1;
    f0 = f1;
    f1 = f;
  }
  return f;
}
```

Donner une spécification précise de la fonction en utilisant la logique du premier ordre sur le domaine des entiers (avec les opérations $+$, $-$, $*$, $/$ pour décrire le lien entre l'entrée n et le résultat retourné $fib(n)$).

2. Donner un jeu de test pour tester la fonction précédente et dérouler à la main les résultats de la fonction.

Exercice 3 Test d'une méthode de calcul de pgcd.

1. Ecrire la spécification d'une fonction *gcd* qui reçoit deux entiers *a, b* en argument et calcule leur pgcd (plus grand commun diviseur). Le spécification sera donnée par une formule du premier ordre sur les entiers (en supposant l'existence des opérations +, -, *, /).
2. Définir un jeu de test pour cette fonction.
3. On donne la fonction suivante :

```
int gcd (int a, int b){
    int temp;
    if (a<0){
        a=-a;}
    if (b<0) {
        b=-b;
    }
    while (b!=0){
        temp=b;
        b=a % b;
        a=temp;
    }
    return a;
}
```

Respecte-t-elle la spécification du pgcd ? La tester sur votre jeu de test.

Exercice 4 Test d'une méthode sur les tableaux.

1. Ecrire une classe Java dont l'attribut privé est un tableau *tab* trié de $n=5$ éléments entiers et qui contient une méthode *cherche(int x)* utilisant la recherche dichotomique renvoyant vrai ssi *x* est un élément du tableau.
2. Définir un jeu de test pour cette classe et vérifier que votre classe passe les tests.
3. Donner le résultat de vos tests sur les 2 implémentations suivantes de *cherche* :

<pre>public boolean cherche(int x){ int i,j,m; boolean found; i=1; j=n; m=0; found=false; while (!(i==j && !found)) { m=(i+j)/2; if (tab[m]<x){ i=m+1; } else {</pre>	<pre>public boolean cherche(int x){ int i,j,m; i=1; j=n;m=0; while (i!=j) { m=(i+j)/2; if (tab[m]<=x) i=m; else j=m; } return (x==tab[m]); }</pre>
--	---

```
        if(tab[x]==m) {
            found=true;
        } else {
            j=m-1;
        }
    }
}
return found;
}
}
```