INFO0946 : Introduction à la Programmation
Challenge 4 (Modules & Invariants)

Benoit Donnet, Simon Liénardy

1 Énoncé du Problème

Simon souhaite disposer d'une énumération des couples de nombres premiers dits *jumeaux*. Deux nombres premiers [1], avec $n > p$, sont dits jumeaux si $n - p = 2$.

Par exemple

```
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
101 103
107 109
```

sont des couples d'entiers premiers jumeaux. Le premier élément de chaque couple donne le nombre p, le deuxième le nombre n.

Dans ce challenge, nous vous demandons de compléter les **trois** fonctions/procédures suivantes :

1. La fonction `est_premier(unsigned int x)` permet de déterminer si le nombre naturel `x` est premier ou pas. Cette fonction retourne 1 si, effectivement, `x` est premier. 0 sinon :

```c
/*
 * @pre: /
 * @post: est_premier vaut 1 si x est un nombre premier. 0 sinon.
 */
int est_premier(unsigned int x){
  //votre code ici
}
```

2. La fonction `sont_jumeaux(unsigned int p, unsigned int n)` permet de déterminer si le couple (p, n) correspond à des nombres premiers jumeaux. Cette fonction retourne 1 si, effectivement, les nombres p et n sont des nombres premiers jumeaux. 0 sinon :

```c
/*
 * @pre: n > p
 * @post: sont_jumeaux vaut 1 si p et non sont premiers jumeaux.
 *                          0 sinon.
 */
int sont_jumeaux(unsigned int p, unsigned int n){
  //votre code ici
}//fin est_jumeaux()
```

3. Enfin, la procédure `affiche_jumeaux(unsigned int x)` affiche la liste des nombres premiers jumeaux entre 1 et x non compris :

1. Pour rappel, l'entier 1 n'est pas considéré comme un nombre premier.

```c
/*
 * @pre: /
 * @post: Tous les nombres premiers jumeaux entre 1 & x sont
 *        affichés à l'écran.
 */
void affiche_jumeaux(unsigned int x){
  //votre code ici
}//fin affiche_jumeaux()
```

L'exemple d'output donné plus haut a été généré avec le code suivant :

```c
#include <stdio.h>

int main(){
  affiche_jumeaux(110);
}//fin programme
```

Dans le corps de chacune des fonctions/procédures, vous pouvez définir toutes les variables que vous jugez nécessaires. Soyez néanmoins précis sur les types de ces variables et n'utilisez pas le caractère underscore (« _ ») dans vos noms de variable.

Lors de votre soumission, vous devrez fournir votre code complétant les fonctions/procédures indiquées ci-dessus ainsi que les invariants de boucle et fonctions de terminaisons nécessaires. La façon de formuler votre invariant est décrite dans la Sec. 3, tandis que la manière de préciser les fonctions de terminaison est décrite dans la Sec. 4. La façon d'écrire votre code est indiquée dans le fichier servant de canevas à votre soumission, fichier disponible sur la page web du cours [2].

1.1 Contrainte supplémentaire

L'instruction « `return` » ne peut être que la dernière instruction de vos fonctions/procédures. Elle ne peut pas apparaitre dans le corps de vos boucles. De plus, vous n'avez pas le droit d'utiliser des instructions « exotiques » vous permettant de court-circuiter une boucle, comme « `break` » ou encore « `continue` ». Votre code sera inspecté en conséquence ! La moindre infraction à cette contrainte entrainera une note nulle à l'ensemble du challenge !

2 Agenda

Votre challenge doit être soumis pour le **vendredi 29/11, 18h00**, au plus tard. Pour rappel, vous disposez de maximum trois essais.

2. https://www.ecampus.ulg.ac.be, Sec. Challenges.

3 Spécifier un Invariant

3.1 Fonction `est_premier`

Complétez l'invariant informel suivant :

Tous les nombres nb, tels que $\boxed{1} \leq nb < \boxed{2}$ *ont été testés et sont tels qu'ils* $\boxed{3}$ $\boxed{4}$ *et* $\boxed{5} \leq \boxed{2} \leq \boxed{6}$.

Mathématiquement, « Tous les nombres nb » s'écrirait $\forall nb$. Les boites **1**, **2** et **4** sont des noms de variables, de constantes ou de valeurs numériques. Les boites **5** et **6** servent à donner des bornes à la boite **2**.

Les différents choix possibles pour $\boxed{3}$ sont les suivants :

1.	sont multiples de	3.	ne divisent pas	5.	multiplient
2.	divisent	4.	sont égaux à	6.	métabolisent

La boite **4** correspond donc au complément du verbe de la boite **3**.

Les différents choix possibles pour $\boxed{6}$ sont les suivants :

1.	x	3.	$log(x)$	5.	$\lfloor\sqrt{x}\rfloor$
2.	$x+1$	4.	$log(x)+1$	6.	$\lfloor\sqrt{x}\rfloor+1$

Veuillez indiquer, dans le fichier de soumission, à la réponse sur l'Invariant de la fonction `est_premier`, le numéro de la boite, suivi d'un point, suivi de la valeur numérique ou du nom de variable ou constante de votre choix (pour les boites de 1, 2, 4 et 5) ou d'un nombre entre 1 et 6 correspondant à votre choix pour les boites 3 et 6.

3.1.1 Exemple

Si vous pensez que l'invariant devrait être :

Tous les nombres nb, tels que $\texttt{0} \leq nb < \texttt{MAX} + 1$ *ont été testés et sont tels qu'ils métabolisent*

`i` *et* $\texttt{0} \leq \texttt{MAX} + 1 \leq \lfloor\sqrt{x}\rfloor + 1$.

Il vous suffit d'encoder la réponse suivante :

```
1. 0
2. MAX+1
3. 6
4. i
5. 0
6. 6
```

3.2 Fonction `sont_jumeaux`

Une brève analyse en sous-problèmes et une bonne compréhension de l'énoncé devraient vous convaincre que l'implémentation de cette fonction ne nécessite pas de boucle (et donc d'Invariant). Si vous n'êtes pas convaincu, relisez attentivement cet énoncé.

3.3 Procédure `affiche_jumeaux`

Complétez l'Invariant informel suivant :

Les nombres $\boxed{1}$ $\boxed{2}$ *entre* $\boxed{3}$ *et* $\boxed{4}$ *(inclus) ont déjà été* $\boxed{5}$.

Les boite **1** et **2** doit être remplacée par des mots prenant la fonction d'épithète [3] du mot « nombres ». N'utilisez pas de diacritiques [4] pour écrire ces mots. Accordez correctement vos épithètes avec le mot masculin pluriel auquel ils se rapportent.

Les boites **3** et **4** doivent être remplacés par des noms de variables, de constantes ou des valeurs numériques. La boite **3** doit désigner une valeur plus petite ou égale à celle de la boite **4**.

Les différents choix possibles pour 5 sont les suivants :

1.	exclus	3.	parcourus	5.	disposés
2.	calculés	4.	affichés	6.	effectués

Veuillez indiquer, dans le fichier de soumission, à la réponse sur l'Invariant de la fonction `affiche_jumeaux`, le numéro de la boite, suivi d'un point, suivi d'un mot (pour les boites 1 et 2), de la valeur numérique ou du nom de variable ou constante de votre choix (pour les boites 3 et 4) et d'un nombre entre 1 et 6 correspondant à votre choix pour la boite 5.

3.3.1 Exemple

Si vous pensez que l'invariant devrait être :

> *Les nombres rebelles entre delta-1 et 666 (inclus) ont déjà été exclus.*

Il vous suffit d'encoder la réponse suivante :

```
1. rebelles
2. _
3. delta-1
4. 666
5. 1
```

Pour n'utiliser qu'un seul mot comme épithète, la boite 2 a été laissée vide.

3.4 Pour ces deux fonctions/procédures

Il est possible d'affecter une variable ou une constante d'un modificateur +1 ou -1. Il suffit dans ce cas d'écrire ce +1 ou -1 comme montré ci-dessus. N'introduisez pas de parenthèses.

Dans tous les cas, si vous pensez que la bonne réponse consiste à ne rien écrire à l'emplacement d'une boite, n'inscrivez rien comme réponse ou un « _ ».

Pour votre facilité, (et pour les distraits qui en oublieraient un), les numéros des boites sont déjà inscrits dans le fichier de réponse.

4 Fonctions de terminaison

La fonction de terminaison [5] permet de fournir la preuve que la boucle se termine. Dans ce Challenge, nous vous demandons de fournir la fonction de terminaison de chacune de vos boucles.

Dans le fichier de réponses, à la question sur la fonction t, nous vous demandons de nous la fournir sous la forme d'une **expression C** valide. Elle ne doit contenir que des noms de variables, constantes, des valeurs numériques et les opérateurs arithmétiques suivants : `+`, `-`, `*`, `/`, `%`.

La correction s'assurera entre autres :
— Que cette expression utilise des variables de votre code ;
— Que son domaine est bien l'ensemble des Entiers (positifs si le gardien est vrai) ;
— Que sa valeur décroit strictement entre deux itérations.

3. https://fr.wikipedia.org/wiki/%C3%89pith%C3%A8te
4. Ouvrez un dictionnaire si vous ne savez pas ce que c'est.
5. À ne confondre ni avec la condition d'arrêt, ni avec le gardien

4.1 Exemple

Si vous pensez que la fonction de terminaison de votre code, qui manipule la variable `toto` et la constante `G` devrait être :

$$t = \texttt{G} + 17 - \texttt{toto}$$

Encodez :

```
G + 17 - toto
```

N'indiquez pas « `F =` », « `t =` » ni « > 0 » mais seulement l'expression qui sert à calculer la valeur de votre fonction de terminaison.

5 Soumettre une Solution

Ces consignes ne sont pas différentes de celles des challenges précédents, hormis le nom du challenge qui change (`challenge4.txt`). Elles sont toutefois rappelées pour mémoire.

Pour tous les challenges, un fichier servant de canevas pour la soumission du challenge est disponible sur la page web du cours [6]. Le nom du fichier est `challengeX.txt` où `X` est remplacé par le numéro du challenge. Le squelette pour ce challenge 4 est donc contenu dans le fichier `challenge4.txt`. Par la suite, libre à vous de modifier le nom du fichier que vous soumettez, cela n'a pas d'importance.

Tous les challenges doivent être compressés en une archive « `.zip` ». Voici comment procéder sur les systèmes d'exploitation les plus courants.

Sous Windows Il suffit de cliquer sur le fichier à l'aide du bouton droit de la souris, sélectionner « Envoyer vers... » et sélectionner ensuite « Dossier compressé ».

Sous Linux (Ubuntu, Fedora, Linux Mint, ...) Il suffit de cliquer sur le fichier à l'aide du bouton droit de la souris, sélectionner « Compresser... ». Veillez bien à sélectionner « `.zip` » dans la liste des extensions possibles pour le fichier.

Sous OS X Cliquez sur le fichier en maintenant la touche Contrôle enfoncée (ou cliquez avec 2 doigts), sélectionnez « Compresser ».

Dans tous les cas Ne soumettez pas de fichier `.tar.gz`, `.7z`, `.rar` ou autre ! C'est bien un fichier `.zip` qui est attendu. Le nom de l'archive importe peu, tant que c'est une archive `zip` valide, dont le nom se termine bien par « `.zip` » **et ne comporte pas de caractères spéciaux comme des espaces, des parenthèses, etc.**

Si vous commettez un erreur dans la soumission, comme par exemple :

- Donner un mauvais nom à l'archive ;
- Soumettre deux fois d'affilée en cliquant trop rapidement ;
- Mal placer les réponses dans le fichier `challenge4.txt`
- ...

C'est dommage pour vous [7]. Redoublez d'attention la prochaine fois ! Pour autant, la plateforme de soumission et le soucis d'équité entre tous les étudiants ne permettent pas de vous octroyer une nouvelle soumission.

6. http://www.ecampus.ulg.ac.be, Sec. Challenges.
7. Nous partageons votre peine.