INFO0946 : Introduction à la Programmation
Challenge 3 (Tableaux & Invariants de Boucle)

Benoit Donnet, Simon Liénardy

1 Énoncé du Problème

Dans ce Challenge, on considère des entiers positifs (i.e., $\in \mathbb{N}$) en base binaire et l'addition binaire de ces entiers.

La forme binaire d'un entier positif, appelée *nombre*, est stockée dans un tableau de N bits. Ainsi, l'entier 244 est représenté en base binaire par le tableau (si on suppose une représentation sur 8 bits – i.e., $N = 8$) `t` suivant :

	0							7
`t` :	1	1	1	1	0	1	0	0

Il est important de remarquer que, dans le tableau `t`, les premiers indices permettent de stocker les bits de poids fort de la représentation binaire de *nombre*, tandis que les derniers indices concernent les bits de poids faible.

Dans ce Challenge, on souhaite écrire un programme permettant de placer, dans un tableau `C`, le résultat de l'addition (binaire) entre deux entiers représentés sous forme binaire par les tableaux `A` et `B`.

Les nombres binaires s'additionnent de la même manière que les nombres décimaux (en travaillant bit par bit au lieu de chiffre par chiffre). Bien évidemment, il y a une différence dans la *retenue*. En effet, en base binaire, il y a une retenue à partir de 2 alors qu'en base décimale, la retenue se fait à partir de 10 [1]. Par exemple, si on désire additionner $(10001101)_2$ (i.e., 141) avec $(00001001)_2$ (i.e., 9), cela donne (la première ligne avec les deux bits en gras correspond à la retenue) :

				1			**1**	
	1	0	0	0	1	1	0	1
+	0	0	0	0	1	0	0	1
	1	0	0	1	0	1	1	0

soit 150.

Pour rappel, les règles de l'addition binaire sont les suivantes :

— $0 + 0 = 0$
— $0 + 1 = 1$
— $1 + 0 = 1$
— $1 + 1 = 10$: j'écris 0 et je retiens 1.
— $1 + 1 + 1 = 11$: j'écris 1 et je retiens 1.

1.1 Contraintes à respecter

Attention, pour résoudre ce problème, vous devez respecter ces contraintes. Ne pas le faire peut entrainer jusqu'à une note nulle pour ce challenge :

— Vous ne pouvez utiliser qu'une et une seule boucle de type **`while`** ;

1. Et en base hexadécimale, la retenue se fait à partir de 16, c'est logique.

— Vous devez viser une complexité théorique $\mathcal{O}(N)$. De plus, vous ne pouvez entrer dans votre boucle, qu'au plus N fois. Attention, le nombre d'itérations sera compté !
— Il existe plusieurs solutions pour résoudre ce problème, notamment une solution (avancée [2]) qui n'utilise ni la moindre structure de contrôle conditionnelle, ni l'opérateur d'addition. Dès lors, **2 points** bonus seront attribués si votre solution (correcte) ne contient ni structure conditionnelle, ni opération d'addition. Cette solution avancée ne doit pas non plus sortir du cadre du cours théorique.
— Vous devez implémenter l'addition binaire comme décrite ci-dessus. La solution consistant à convertir les tableaux en nombres entiers, à les additionner et à reconvertir le résultat en tableau ne sera pas acceptée.
— Vous ne pouvez utiliser que les tableaux fournis `A`, `B` et `C` et vous ne pouvez accéder à leurs éléments qu'avec l'opérateur `[]`. Le contenu de `A` et `B` ne devrait pas changer.
— **Le code que vous soumettez ne peut faire appel à aucune librairie.**

1.2 Squelette du code

Le squelette de votre code est le suivant :

```
int main(void){
  const int N : ...;
  /* Les tableaux suivants contiennent uniquement les valeurs entières 1 et 0 */
  unsigned char A[N];
  unsigned char B[N];
  unsigned char C[N];

  /*
   * Code de remplissage des tableaux A et B
   * Ce code ne vous est pas donné.
   */

  // Votre code viendra ici (variables + instructions)

  return 0;
}//fin programme
```

Vous devez considérer, lors de la soumission, que votre extrait de code est copié et collé [3] à l'endroit indiqué par le commentaire « `Votre code viendra ici (variables + instructions)` ». Dès lors, redéclarer la constante `N`, ou encore les tableaux `A`, `B` ou `C` fera échouer la compilation.

Lors de votre soumission, vous devrez fournir votre code ainsi que l'Invariant Graphique et sa Fonction de Terminaison. La façon de formuler votre Invariant Graphique est indiquée dans la Sec. 3 de cet énoncé tandis que la façon de formuler la Fonction de Terminaison est décrite dans la Sec. 4. Le fichier servant de canevas à votre soumission est disponible sur eCampus (Sec. Challenges).

2 Agenda

Votre challenge doit être soumis pour le **vendredi 20/11, 20h00**, au plus tard. Pour rappel, vous disposez de maximum trois essais.

3 Spécifier un Invariant Graphique

Ce problème peut être résolu sans découpe en sous-problème.

3.1 Conseils

Il est recommandé de d'abord rechercher les Invariants Graphiques **sans** s'inspirer des canevas imposés. Par exemple, vous pouvez faire votre propre dessin sur une feuille de papier ou en utilisant le Gli

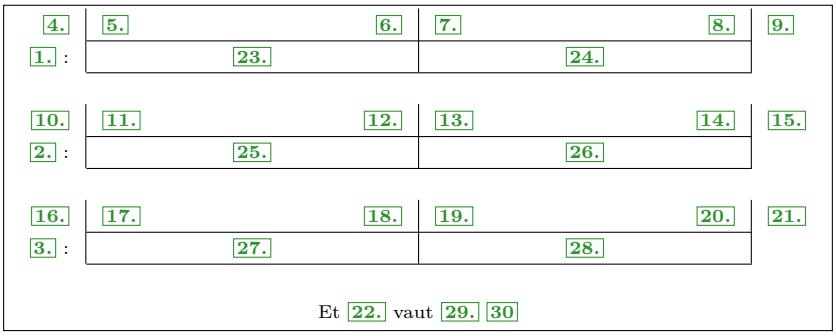
2. Conseil : n'envisagez une telle solution que si vous avez encore 2 soumissions disponibles, au cas où...
3. En fait, c'est à quelques détails près le cas.

et par la suite, comparez ce dessin aux Invariants Graphiques à compléter, détaillés dans les sections suivantes. De toute manière, si votre Invariant Graphique est correct, il devrait contenir la *même information* que celui que vous soumettrez selon notre modèle et vous n'aurez qu'à adapter légèrement votre solution pour la rendre compréhensible par la machine.

Si vous êtes bloqué·e, un coup d'œil au modèle devrait vous mettre sur la piste.

3.2 Invariant Graphique Muet

Voici un Invariant Graphique muet [4] :



Selon l'équation fondamentale des Invariants de Boucle des codes qui manipulent des tableaux, i.e., :

$$1 \text{ Tableau} = 1 \text{ Dessin},$$

cet Invariant de Boucle doit représenter les tableaux manipulés par votre code.

Les boites 1. à 22. doivent être remplacées par des constantes ou des noms de variables.

Toutes les boites ne doivent pas forcément être précisées : si vous indiquez que la valeur de la boite 8. est « k », il est implicite que la valeur de la boite 9. est est « $k+1$ » : inutile alors de le préciser.

Pour les autres boites (23. à 30.), les choix disponibles sont résumés dans le tableau 1.

boites 23. à 28	**boites 29. et 30**
1. À Calculer	1. un routier [5]
2. Déjà Calculé	2. sauce samouraï
3. À parcourir	3. le reste
4. Déjà parcouru	4. de la multiplication
5. À Afficher	5. le bit de poids faible
6. Déjà Affiché	6. du calcul
7. À Additionner	7. de l'addition
8. Déjà Additionné	8. le report
9. À Remplir	9. le bit de poids fort
10. Déjà rempli	10. de la division

TABLE 1 – Invariant Graphique : Possibilités de choix pour les boites 23. à 30.

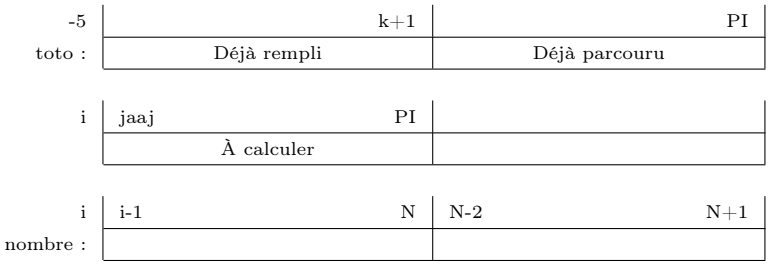
Veuillez indiquer, dans le fichier de soumission, à la réponse sur l'Invariant Graphique, le numéro de la boite, suivi d'un point, suivi de la valeur numérique ou du nom de variable ou constante de votre choix (pour les boites de 1. à 22.) ou d'un nombre entre 1 et 10 correspondant à votre choix (pour les boites 23. à 30.).

4. Tout ce qui est dans le cadre fait partie de l'Invariant Graphique

5. Spécialité culinaire composée d'une baguette, remplie d'une viande, de sauce et recouverte de frites. En fonction de la région, on parle aussi de « mitraillette ». Quant à la sauce samouraï, sans aucun rapport avec le Japon, c'est un mélange de ketchup, mayonnaise et piment.

3.2.1 Encodage dans le Fichier de Soumission

Si vous pensez qu'un bon Invariant Graphique serait :



Et Simon vaut un routier sauce samouraï

Alors[6], complétez le fichier de réponse comme suit (Pour gagner de la place, j'ai mis ça sur trois colonnes. Dans le squelette, il faut les mettre les unes en dessous des autres. Les numéros sont mis pour vous. Il est très difficile de se tromper.) :

```
1. toto          11. jaaj         21. _
2. _             12. PI           22. Simon
3. nombre        13. _            23. 10
4. -5            14. _            24. 4
5. _             15. _            25. 1
6. k+1           16. i            26. _
7. _             17. i-1          27. _
8. PI            18. N            28. _
9. _             19. N-2          29. 1
10. i            20. N+1          30. 2
```

Il est toujours 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 distrait·e·s qui en oublieraient un), les numéros des 30 boites sont déjà inscrits dans le fichier de réponse.

4 Fonction de Terminaison

La Fonction de Terminaison[7] 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 (puisque 1 Invariant de Boucle == 1 boucle).

Dans le fichier de réponses, aux questions sur la Fonction de Terminaison, nous vous demandons de nous la fournir sous la forme d'une **expression C** valide.

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

6. Vous êtes conscient que ce n'est pas la bonne réponse ?
7. À ne confondre ni avec le Critère d'Arrêt, ni avec le Gardien de Boucle

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 =` » ou « `t =` » mais seulement l'expression qui sert à calculer la valeur de votre Fonction de Terminaison. De plus, n'ajouter pas « > 0 » à votre soumission. en effet, $G + 17 - toto > 0$ n'est pas une fonction à valeur entière mais Booléenne [8].

5 Soumettre une Archive `.zip`

Pour tous les challenges, un fichier servant de canevas pour la soumission du challenge est disponible sur la page web du cours [9]. Le nom du fichier est `challengeX.txt` où `X` est remplacé par le numéro du challenge. Le squelette pour ce challenge 3 est donc contenu dans le fichier `challenge3.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 ou utiliser des caractères inhabituels (e.g., des parenthèses) ;
- Soumettre deux fois d'affilée en cliquant trop rapidement ;
- Mal placer les réponses dans le fichier `challenge3.txt`
- ...

C'est dommage pour vous [10]. 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.

8. Et donc c'est incorrect...
9. http://www.ecampus.ulg.ac.be, Sec. Challenges.
10. Nous partageons votre peine.