INFO0947 : Compléments de Programmation
Préfixe & Suffixe

B. Donnet, G. Brieven
Université de Liège

1 Problème

Soit T, un tableau à N valeurs entière ($N \geq 0$). On veut construire une fonction qui permet d'obtenir, pour T, le plus grand entier k ($k \in [0, \ldots, N-1]$) tel que le sous-tableau `T[0 ...k-1]` est à la fois préfixe et suffixe de `T`. Si un tel sous-tableau n'existe pas, la fonction doit renvoyer la valeur 0. Attention, on fait l'hypothèse que $k \neq N$ sinon le problème devient trivial.

Dans ce travail, on vous demande de construire la fonction suivante :

```
1 int prefixe_suffixe(int *T, const unsigned int N);
```

où `T` est un tableau de taille `N`, avec `N` ≥ 0. Cette fonction ne modifie pas `T`.

Pour votre solution, il est obligatoire de la construire autour d'une ou plusieurs boucles `while`. Vous ne pouvez pas passer par un tableau intermédiaire.

Exemple 1 :

	0								8
T1 =	1	4	2	4	5	1	4	2	4

Le bout de code suivant :

```
1 int lg = prefixe_suffixe(T1, 9);
2 printf("longueur: %d\n", lg);
```

doit afficher à l'écran la valeur entière 4 car le sous-tableau :

1	4	2	4

est le plus grand sous-tableau de `T1` qui est à la fois préfixe et suffixe de `T1`.

Exemple 2 :

	0									9
T2 =	1	2	3	2	1	1	2	3	2	1

Le bout de code suivant :

```
1 int lg = prefixe_suffixe(T2, 10);
2 printf("longueur: %d\n", lg);
```

doit afficher à l'écran la valeur entière 5 car le sous-tableau :

1	2	3	2	1

est le plus grand sous-tableau de `T2` qui est à la fois préfixe et suffixe de `T2`.

Exemple 3 :

Le bout de code suivant :

		0								8
T3	=	3	2	3	2	1	2	3	2	1

```
int lg = prefixe_suffixe(T3, 10);
printf("longueur: %d\n", lg);
```

doit afficher à l'écran la valeur entière 0 car `T3` ne possède pas de sous-tableau qui soit à la fois préfixe et suffixe, excepté `T3` lui-même.

2 Travail à Réaliser

On demande, pour le module `prefixe_suffixe()`, de résoudre le problème en suivant le cheminement suivant :

1. formaliser (= utiliser une notation mathématique concise et précise pour décrire) le problème. À cette fin, proposer de nouvelles notations et/ou de nouveaux concepts ;
2. spécifier le plus précisément et formellement possible le module `prefixe_suffixe()` ;
3. donner une découpe en SP respectant les contraintes du problème donné et indiquer comment ils s'emboitent (si applicable) ;
4. spécifier complètement et formellement chaque nouveau SP, si applicable ;
5. proposer un Invariant Graphique et un Invariant Formel pour chaque boucle de votre code ;
6. construire des morceaux de programme correspondant à chaque SP (en adoptant l'*approche constructive* vue au cours) ;
7. rédiger le module final ;
8. démontrer de manière rigoureuse et formelle [1] la complexité théorique, dans le pire des cas (i.e., en utilisant la notation de Landau), de votre solution.

Lors de la rédaction du code, veillez à respecter les principes de la **programmation défensive**.

3 Aspects Pratiques

Le travail à rendre se compose d'une archive `tar.gz` [2]. Votre archive portera le nom `pref_suf-groupeXX.tar.gz`, où `XX` fait référence à l'identifiant de votre groupe [3].

Une fois décompressée, votre archive devra donner naissance à deux répertoires : `rapport/` (cfr. Sec. 3.1) et `code/` (Sec. 3.2).

Tout non-respect des consignes se verra sanctionné par 2 points en moins dans la note finale de votre projet [4].

3.1 Rapport

Votre rapport devra être rédigé via l'outil LaTeX(nous vous renvoyons à la formation donnée le CSS, le 06/02/2025, pour la rédaction d'un document en LaTeX) en utilisant le template LaTeXdisponible sur eCampus (Sec. Projets).

Le template est organisé comme suit :

1. Toute justification impliquant un texte argumentatif en lieu et place d'équations ne sera pas lue. Expliquez tout de même votre raisonnement.
2. Nous vous renvoyons à la formation sur la ligne de commande, donnée au Q1, pour la compression des fichiers dans une archive `tar.gz`.
3. Pour rappel, il est interdit de changer de groupe entre les différents projets. Votre identifiant doit nécessairement être un nombre composé de deux chiffres (compris entre 01 et 50).
4. De plus, toute archive ne pouvant être décomposée (c'est-à-dire une archive corrompue) ou ne produisant pas les dossiers `rapport/` et `code/` lors de la décompression engendrera une note finale **nulle**.

— `main.tex`. C'est le fichier principal. Vous y trouverez une section « TITRE » délimitée par des commentaires. Il suffit de compléter les définitions des macros `\intitule`, `\GrNbr`, `\PrenomUN`, `\NomUN`, `\PrenomDEUX` et `\NomDEUX` par, respectivement, le titre du projet, votre numéro de groupe, vos prénoms et noms. **Attention !** Ne modifiez pas les dix lignes appelées « *Zone protégée* ».
— `introduction.tex`. Dans ce fichier, vous rédigez l'introduction de votre rapport (inutile de recopier l'énoncé du projet, essayez d'y mettre du vôtre).
— `formalisation.tex`. Dans ce fichier, introduisez toutes les notations mathématiques que vous jugez utiles à la réalisation du projet.
— `analyse.tex`. Dans ce fichier, vous devez définir (Input/Output/Objets Utilisés) proprement et clairement le problème. Si applicable, il est aussi demandé de réaliser une analyse complète (i.e., découpe en SP).
— `specifications.tex`. Dans ce fichier, pécifiez formellement chacun des SP.
— `invariants.tex`. Dans ce fichier, indiquez et décrivez tous les Invariants de Boucle nécessaires. Pour chaque SP nécessitant un Invariant de Boucle (une sous-section/SP), donnez l'Invariant Graphique et l'Invariant Formel correspondant. Pour l'Invariant Graphique, nous vous rappelons qu'il est possible d'exporter les Invariants Graphiques au format PDF depuis le GLI.
— `construction.tex`. Dans ce fichier, il est demandé d'appliquer l'approche constructive pour la construction de votre code ({PRÉCONDITION} INIT {INV}, ...).
— `code.tex`. Dans ce fichier, indiquez le code complet (sans assertions intermédiaires) de votre solution.
— `complexite.tex`. Dans ce fichier, vous devez étudier complètement la complexité de votre code. Soyez le plus formel (i.e., mathématique) possible.
— `conclusion.tex`. Dans ce fichier, vous indiquez la conclusion de votre rapport.

Soyez clairs et précis dans vos explications. N'hésitez pas à ajouter un schéma si vous le jugez nécessaire. Dans ce cas, expliquez-le : un schéma seul ne suffit pas !

Soignez votre orthographe : au delà de trois fautes, chaque faute vous fera perdre 0,25 points.

Pour compiler votre rapport, utiliser la commande suivante :

```
$>pdflatex main.tex
```

Une fois compilé, votre document LaTeXdevra produire un fichier PDF dont le nom sera `pref_suf-XX.pdf`, où `XX` représente toujours votre numéro de groupe. Ce rapport ne pourra pas compter plus de **12 pages**. [5]

Le dossier `rapport/` sera composé des éléments suivants :
— votre rapport au format `.tex` ainsi que toutes les sources utiles à la compilation ;
— votre rapport déjà compilé au format `.pdf`

Remarque : Pour inclure du code C dans un document LaTeX, voir la remarque publiée sur le forum eCampus (Projets – Informations Générales → Inclure du code dans le rapport + conseils latex).

3.2 Code Source C

Votre code source devra être rédigé en langage C. Votre code source devra être accompagné d'un `Makefile` [6] rédigé par vos soins, la commande `make` permettant la génération d'un fichier binaire exécutable appelé `prefixe_suffixe`. Ce fichier exécutable sera situé dans le même répertoire que celui où se trouve le code source (c'est-à-dire le répertoire `code/`). Son exécution doit nous permettre de tester la validité de votre code. La fonction `main` de cet exécutable devra être implémentée dans un fichier appelé `main-prefixe_suffixe.c`.

Votre code source sera adéquatement découpé en headers et modules. Les différents sous-problèmes identifiés dans votre rapport devront apparaitre clairement dans votre code et être déclaré dans le fichier `prefixe_suffixe.h`. Nous vous fournissons une première version du fichier `prefixe_suffixe.h` qui, à

5. Sans compter la page de garde, ni son verso.
6. cfr le cours INFO0030, "Partie 2 – Chapitre 1 : Compilation".

ce stade, ne contient que la déclaration du module `prefixe_suffixe()`. Il vous faut donc implémenter le contenu de `prefixe_suffixe.h` dans un fichier `prefixe_suffixe.c`.

Le module `prefixe_suffixe()` (et donc le fichier `prefixe_suffixe.c`) doit pouvoir être compilé et utilisé sans la présence du fichier `main-prefixe_suffixe.c`.

Il est **interdit**[7] d'utiliser des fonctions ou procédures de librairies tierces (y compris la librairie standard du C), à l'exception de `assert.h` et `stdlib.h`.

4 Agenda

4.1 Milestones

Pendant la durée du projet, nous vous proposons deux *milestones*. L'objectif de ces milestones est de rencontrer chaque groupe individuellement afin de discuter de l'avancement du projet et corriger le tir le cas échéant. Ces rencontres sont organisées durant resp. la 3e et la 2e semaines qui précèdent la remise du travail final. À l'issue de ces rencontres, aucune note ne sera affectée. L'objectif est donc de réaliser une évaluation formative de votre travail pour vous aider à réaliser le projet et à en tirer des enseignements.

Les deux milestones sont les suivants :

— **semaine du 10/03/2025**. Le milestone portera sur les notations/formalisations du problème. Afin de permettre à l'équipe pédagogique de préparer au mieux le feedback, il vous est demandé de nous (i.e., Benoit Donnet et Géraldine Brieven) envoyer par email un petit document[8] (pdf, rédigé en LaTeX) décrivant les différentes notations que vous comptez introduire. La deadline pour l'envoi de ce document est le samedi 08/03/2025, 12h00. Vous devez spécifier, dans l'objet de votre email, la chaîne de caractères suivantes : `[INFO0947] Projet 1 - Milestone 1`

— **semaine du 17/03/2025**. Le milestone portera sur les Invariants de Boucle nécessaires à la réalisation du problème. Afin de permettre à l'équipe pédagogique de préparer au mieux le feedback,il vous est demandé de nous (i.e., Benoit Donnet et Géraldine Brieven) envoyer par email un petit document (pdf, rédigé en LaTeX) décrivant vos Invariants de Boucle (Invariant Graphique et Invariant Formel) que vous comptez introduire. La deadline pour l'envoi de ce document est le samedi 15/03/2025, 12h00. Vous devez spécifier, dans l'objet de votre email, la chaîne de caractères suivantes : `[INFO0947] Projet 1 - Milestone 2`

Vous trouverez, sur eCampus, un template LaTeXpour formater correctement votre production pour chaque milestone.

La participation à ces milestones est obligatoire. Aucun groupe ne peut s'y soustraire. Pour définir le moment auquel votre groupe viendra rencontrer l'équipe pédagogique, deux sujets de discussion ont été créés dans le forum du cours, sur la plateforme eCampus. Il vous suffit de lire et de suivre les instructions détaillées dans le premier message de chaque sujet.

Veuillez faire débuter les sujets des mails envoyés à l'équipe pédagogique par la chaine "**[INFO0947]**".

4.2 Deadline

Les archives `tar.gz` finales doivent être soumises sur la Plateforme de Soumission avant le **lundi 07/04/2025** à **18h00** (l'heure du serveur faisant foi). Toute soumission postérieure à cette date ne sera pas prise en compte. Comme l'heure de l'horloge du serveur de soumission et celle de votre ordinateur peuvent être légèrement différentes, il est **fortement déconseillé** de soumettre votre projet à 17h59 et 59 secondes : ne prenez pas de risque inutile.

7. Cette restriction ne s'applique pas au fichier `main-prefixe_suffixe.c` où est implémenté l'exécutable de test `prefixe_suffixe`
8. Petit = 1 page.