INFO0947 : Compléments de Programmation
Préfixe & Suffixe

B. Donnet, S. Liénardy
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 :

```
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
T =	1	4	2	4	5	1	4	2	4

Le bout de code suivant :

```
int lg = prefixe_suffixe(T, 9);
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 `T` qui est à la fois préfixe et suffixe de `T`.

Exemple 2 :

	0									9
T =	1	2	3	2	1	1	2	3	2	1

Le bout de code suivant :

```
int lg = prefixe_suffixe(T, 10);
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 `T` qui est à la fois préfixe et suffixe de `T`.

	0								8
T =	3	2	3	2	1	2	3	2	1

Exemple 3 :

Le bout de code suivant :

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

doit afficher à l'écran la valeur entière 0 car `T` ne possède pas de sous-tableau qui soit à la fois préfixe et suffixe, excepté `T` lui-même.

Dans la suite de l'énoncé, vous trouverez à la Sec. 2 une énumération qui vous liste, avec précision le travail que vous devez réaliser pour mener à bien ce projet. L'évaluation du projet comprend `3 étapes` : deux *Milestones*[1] qui ne comptent pas dans la note finale mais qui sont néanmoins obligatoires (voir Sec. 4.2) ainsi que la remise du rapport final (Sec. 3.1) et de votre code source (Sec. 3.2) sur la plateforme de soumission. Les sections suivantes vous détaillent quelles parties du travail sont à produire pour chaque étape de l'évaluation.

2 Travail à Réaliser

On demande, pour la procédure `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. Cette étape sera évaluée à la **Milestone 1** ;
2. spécifier le plus précisément et formellement possible la procédure demandée ;
3. donner une découpe en sous-problèmes respectant les contraintes du problème donné et indiquer comment ils s'emboitent ;
4. spécifier complètement et formellement chaque nouveau sous-problème ; Ce point ainsi que le précédent sont l'objet de la **Milestone 2** ;
5. construire des morceaux de programme correspondant à chaque sous-problème (en adoptant l'*approche constructive* vue au cours) ;
6. rédiger la fonction finale ;
7. démontrer de manière rigoureuse et formelle[2] que la complexité théorique, dans le pire des cas (i.e., en utilisant la notation de Landau) de votre solution est bien en $O(N^2)$.

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`[3]. Votre archive portera le nom `pref_suf-groupeXX.tar.gz`, où `XX` fait référence à l'identifiant de votre groupe[4].

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

1. « Étapes », en français
2. Toute justification impliquant un texte argumentatif en lieu et place d'équations ne sera pas lue. Expliquez tout de même votre raisonnement.
3. Nous vous renvoyons à la formation sur la ligne de commande, donnée au Q1, pour la compression des fichiers dans une archive `tar.gz`.
4. 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).

Tout non-respect des consignes se verra sanctionné par 2 points en moins dans la note finale de votre projet [5].

3.1 Rapport

Votre rapport devra être rédigé via l'outil LaTeX(nous vous renvoyons à la formation donnée le 04/02/2020 pour la structuration de votre document et celle du 13/02/2020 pour la rédaction d'un document en LaTeX) en utilisant l'en-tête du template LaTeXdisponible sur la page web du cours.

Dans le template, vous 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. Vous pouvez redéfinir la macro `\tablemat` pour insérer une table des matières.

Attention ! Ne modifiez pas les dix lignes appelées « *Zone protégée* ».

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**. [6]

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`

Votre rapport devra contenir tous les éléments nécessaires à la compréhension de votre code source, présentés clairement et dans un ordre logique. Soit, entre autres :
— la spécification formelle de votre procédure `prefixe_suffixe()` ;
— votre découpe en sous-problèmes, la description de chaque sous-problème et comment ils s'emboitent ;
— la spécification formelle de chaque sous-problème ;
— si un problème nécessite l'introduction d'une boucle, il faudra fournir un dessin de la situation générale et en dériver un invariant formel ;
— les différentes étapes ({Pré} *INIT* {Inv}, ...) de la construction de chaque sous-problème et la fonction de terminaison de chaque sous-problème (quand cela s'avère pertinent). Nous vous demandons donc d'appliquer **explicitement** la démarche constructive telle que vue au cours.

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.

Nous vous renvoyons au séminaire donné par Patricia Tossings, le 04/02/2020, sur la rédaction de rapport pour vous guider au mieux dans vos efforts d'écriture (n'hésitez pas à consulter les slides de la formation).

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 [7] 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.c`.

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

Votre code source sera adéquatement découpé en headers et modules. Les différents sous-problèmes identifiés dans votre rapport devront apparaitre clairement dans votre code. La fonction `prefixe_suffice()` doit être déclarée dans un header nommé `prefixe_suffixe.h` et doit pouvoir être compilée et utilisée sans la présence du fichier `main.c`.

Il est **interdit** [8] d'utiliser des fonctions ou procédures de librairies tierces (y compris la librairie standard du C), à l'exception de `assert.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 09/03/2020**. Le milestone portera sur les notations/formalisations du problème. Afin de permettre à l'équipe pédagogique de préparer au mieux cette rencontre, il vous est demandé de nous (i.e., Benoit Donnet **et** Simon Liénardy) envoyer par email un petit document [9] (pdf, rédigé en LaTeX) décrivant les différentes notations que vous comptez introduire. La deadline pour l'envoi de ce document est le samedi 07/03/2020, 12h00.

— **semaine du 16/03/2020**. Le milestone portera sur les invariants nécessaires à la réalisation du problème. Afin de permettre à l'équipe pédagogique de préparer au mieux cette rencontre, il vous est demandé de nous (i.e., Benoit Donnet **et** Simon Liénardy) envoyer par email un petit document (pdf, rédigé en LaTeX) décrivant vos invariants (situation générale, dessin, et invariant formel) que vous comptez introduire. La deadline pour l'envoi de ce document est le samedi 14/03/2020, 12h00.

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 `http://submit.montefiore.ulg.ac.be` avant le **30/03/2020** 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.

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