CSC_3IN01_TA - Examen
21/01/2025
François Pessaux

Énoncé

- **Lisez attentivement les consignes et tout le sujet avant de commencer.**

- Les documents (polys, transparents, TDs, livres ...) sont autorisés.
- Sont **absolument interdits** : le WEB, le courrier électronique, les messageries diverses et variées, le répertoire des camarades, le téléphone (même pour avoir l'heure puisque vous l'avez sur votre ordinateur).

- Votre travail sera (en partie) évalué par un mécanisme automatique. Vous **devez** respecter les règles de **nommage** des fichiers et autres **consignes** qui vous sont données.
- La connaissance de C est un pré-requis du cours. Ainsi, la présence d'avertissement(s) ou d'erreur(s) à la compilation réduira mécaniquement la note de l'exercice considéré.
- **Sauf indications contraires**, vos programmes doivent **gérer** les cas **d'erreur** pouvant survenir.
- Lorsqu'il vous est demandé que votre programme réponde en affichant « **Yes** » ou « **No** », il ne doit **rien** afficher d'autre, et **pas** « Oui » ou « Yes. » ou « no » ou « La réponse est : no ». Donc, pensez à retirer vos affichages de test / debug.
- La **lisibilité** et l'efficacité / simplicité / complexité de vos programmes seront **prises en compte** dans l'évaluation.

- **À la fin de l'examen**, vous soumettrez vos fichiers sources dans l'espace **Moodle** dédié à l'examen. Pas de retour par mail, à l'exception des **photos de brouillon** (c.f haut de la page suivante).
- **N'oubliez pas** d'effectuer cet envoi sinon nous devrons considérer que vous n'avez rien rendu !

- Le sujet comporte **4** pages et l'examen dure **3** heures.

Attention : Pour les questions de réflexion, de pseudo-code, vous pouvez les rédiger sur papier. Dans ce cas, vous mettrez votre nom sur chaque page et la numéroterez (en ordre strictement croissant) puis vous ferez une photo desdites pages que vous m'enverrez par **mail** (et uniquement ça par mail). **Vous devrez quand même remettre ces feuilles/brouillons aux surveillants avant de sortir de l'examen.**

Sinon, si vous préférez, vous pouvez rédiger les réponses dans votre fichier source, en commentaire en début de fichier.

1 Cryptographie

Des fichiers texte vous sont parvenus, mais chiffrés par votre correspondant. Le contenu de chaque fichier est sous la forme d'une séquence arbitrairement longue de paires constituées d'un mot et du numéro de son successeur dans le fichier chiffré. Le successeur du dernier mot du texte est indiqué par la valeur -1. En tout début de fichier se trouve indiqué le nombre total de mots du texte.

Pour savoir quel est l'indice du premier mot, une clef secrète doit être connue et transmise au programme de déchiffrement.

Dans le cas où le déchiffrement échouerait pour quelle que raison que ce soit, que des choses aient commencé à être affichées ou non, l'affichage devra faire figurer **sur une ligne à part** le texte `"Error while decoding."` (sans guillemets) suivi d'un retour à la ligne.

Le but macroscopique est d'écrire le programme qui prend en argument le nom du fichier chiffré et la clef secrète (dans cet ordre) et affiche le texte déchiffré.

Exemple :

```
5 deux 3 quatre 2 cinq -1 trois 1 un 0
```

Si l'on sait que la clef de déchiffrement est 4, on peut retrouver le texte initial qui est « *un deux trois quatre cinq* ». En effet, avec la clef qui vaut 4, on trouve le premier mot « *un* », le suivant se trouve à la position 0 et est donc « *deux* » et ainsi de suite jusqu'à « *cinq* » dont le successeur est inexistant puisqu'il est indiqué la valeur -1.

Nommage : Le fichier source de ce programme devra s'appeler `decipher.c`.

Format de sortie : Le texte décodé, chaque mot étant séparé du suivant par un espace. Un retour chariot final devra terminer l'affichage.

Si un message d'erreur doit être affiché (dont celui imposé ci-dessus), il devra l'être sur une ligne **séparée** du texte éventuellement déjà déchiffré et affiché.

Exemples et jeux de données : Plusieurs exemples de fichiers vous sont fournis pour tester votre programme (`text1_key3.cip`, `text1_key8.cip`, `text2_key16.cip`, etc.). La partie `key`xyz spécifie quelle est la clef secrète qui doit être utilisée pour le déchiffrement. Par exemple, pour `text2_key16.cip`, la clef est 16.

Vous avez également les fichiers d'origine (avant chiffrement) à disposition (`text1.txt`, `text2.txt`, etc.). Le déchiffrement devra donc redonner ces texte, aux sauts de ligne près.

Question 1.1

Quelles sont les grandes lignes du programme (la structure du futur `main` en quelque sorte) ?

Question 1.2

Par quelle structure de données pouvez-vous représenter le texte chiffré ?

Question 1.3

On s'intéresse maintenant à la fonction qui va permettre d'acquérir le contenu du texte à déchiffrer. Quels sont ses domaines d'entrée et de sortie ?

Question 1.4

Quel est l'algorithme de la fonction de la question précédente ? **On ne vous demande pas** de gérer les cas de malformation du contenu du fichier à lire.

Question 1.5

On s'intéresse maintenant à la fonction qui va permettre de déchiffer un texte préalablement chargé. Quels sont ses domaines d'entrée et de sortie ?

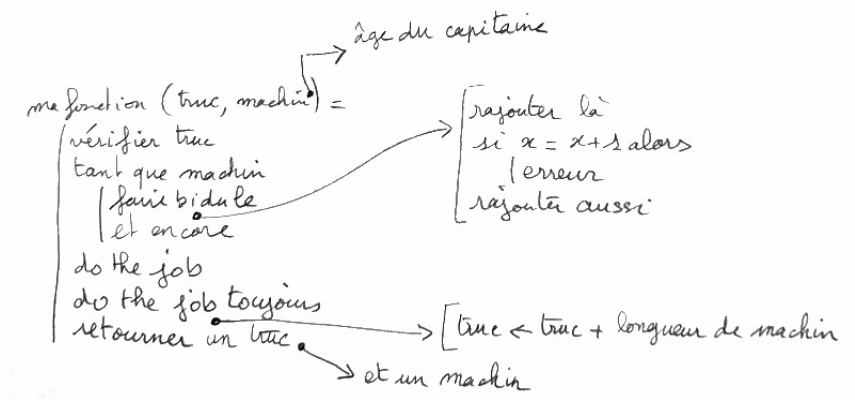
Question 1.6

Décrivez l'algorithme de cette fonction de déchiffrement.

Question 1.7

On souhaite maintenant ajouter une vérification disant si le déchiffrement a été un succès, autrement dit, si la clef était la bonne. Comment peut-on vérifier cette propriété, quels changements sont à apporter à votre algorithme ?

Si vous avez décrit votre algorithme sur papier, vous pouvez insérer des flèches vers des encarts sur votre papier où figurent les quelques lignes à rajouter à droite à gauche. Un « brouillon » tout à fait acceptable pourrait avoir la forme suivante :



Question 1.8

Des petits malins pourraient vouloir provoquer un déni de service en fournissant un texte chiffré incorrect dont le successeur d'un mot est un mot déjà affiché, afin de faire boucler indéfiniment notre programme.

Proposez une solution et expliquez ce qu'elle change à votre algorithme déjà amélioré à la question précédente (1.7).

Question 1.9

Proposez un principe (pas besoin de pseudo-code ni de l'implanter en C) qui permettrait de casser le chiffrement d'un fichier (i.e. de le déchiffrer même si l'on n'a pas la clef secrète).

Question 1.10

Implantez le programme de déchiffrement (avec toutes les vérifications rajoutées aux questions 1.7 et 1.8) en C et avec un `main` qui orchestre ces traitements.

Question 1.11

Afficher le texte déchiffré au vol, comme demandé jusqu'à présent, n'est pas très satisfaisant car en cas d'erreur, on aura une version tronquée affichée qui viendra polluer la seule information importante, à savoir le message d'erreur `"Error while decoding."` imposé par l'énoncé.

On souhaite avoir une **seconde fonction** (par exemple, `decode_message_better`) de décodage qui **retourne** l'unique chaîne représentant tout le texte une fois décodé (et non qui se contente, comme la première, de l'afficher).

Expliquez quelles modifications devront être apportées à votre algorithme suffisamment clairement ou bien donnez une nouvelle version de celui-ci.

Remarque : Il sera sans doute bien utile que votre fonction d'acquisition du texte à déchiffrer retourne une valeur en plus. Elle ne sera simplement pas utile pour la première version du déchiffrement.

—— Fin du sujet ——