INF1600

Contrôle périodique 2

Mon nom de famille est : zerhouni

Mon prénom est : rayan

Mon matricule est : 2019519

J'affirme sur mon honneur avoir fait ce devoir sans l'aide de personne.

Réécrivez la phrase ci-après :

J'affirme sur mon honneur avoir fait ce devoir sans l'aide de personne.

Pour ce travail, il vous est demandé de programmer en assembleur des fonctions données en C. Le contrôle périodique est personnalisé suivant votre matricule. Pour ce faire, il vous est demandé de compléter le calcul suivant pour déterminer les valeurs de A, B, C et D.

Votre matricule = 2019519 ;

Votre matricule % 2 = \A = 1 ;

Votre matricule % 3 = B = 0 ;

Votre matricule % 5 = C = 4 ;

Votre matricule % 7 = D = 5 ;.

Le contrôle périodique est accompagné de fichiers .h, .c, .s et d’un makefile. Pour générer l’exécutable, il vous suffit d’exécuter make à la console. Le programme s’exécute en entrant à la console :

$> ./exec <votre matricule>

Pour chaque fonction \*\_asm.s que vous devez implémenter ci-après, une version \*\_c.c vous est fournie à titre de référence. Il n’est pas nécessaire de produire un code assembleur équivalent à celui que le compilateur gcc vous donnerait à partir du code c. Par exemple, si vous estimez qu’il n’est pas utile de sauvegarder l’ancienne valeur de %ebp sur la pile, il sera inutile de le faire. Indiquez et justifiez ce choix dans vos réponses aux questions ci-après.

Il est possible de tester votre implémentation \*\_asm.s en modifiant le fichier main.c de sorte que la constante associée retourne 1. Par exemple, pour tester la question 3, on mettra :

#define TEST\_Q3 **1**

Dans votre remise, fournissez un fichier zip avec l’ensemble des fichiers \*.s que vous aurez complétés ainsi qu’un fichier pdf comportant vos réponses aux questions.

1) **(1 point)** Reproduisez l’affichage obtenu en exécutant le programme avec votre matricule en paramètre :

A = 1, B = 0, C = 4, D = 5, key = 11

keyword = 2019519

- Q2

Mon matricule est 2019519

- Q3

is\_lower\_case\_c a vu que 'a' est une minuscule.

is\_lower\_case\_c a vu que 'A' n'est pas une minuscule.

- Q4

is\_upper\_case\_c a vu que 'A' est une majuscule.

is\_upper\_case\_c a vu que 'a' n'est pas une majuscule.

- Q5

Encryption de Cesar:

Ecfxa l-e-tw rlryp wpd pwpnetzyd?

- Q6

Decryption de Cesar:

Trump a-t-il gagne les elections?

- Q7

Encryption de Vigenere:

Vrvvu j-v-ju hjipe ujt gnedcnpwu?

- Q8

Decryption de Vigenere:

Trump a-t-il gagne les elections?

2) **(1 point)** Implémentez la fonction append\_asm qui joint à une chaîne de caractères dst une seconde chaîne de caractère src. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

Non, il n’est pas utilisé dans la méthode

3) **(1 point)** Implémentez la fonction is\_lower\_case\_asm qui renvoie 1 si le caractère reçu en paramètre est une minuscule et 0 autrement. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

Non, il n’est pas utilisé dans la méthode

4) **(1 point)** Implémentez la fonction is\_upper\_case\_asm qui renvoie 1 si le caractère reçu en paramètre est une majuscule et 0 autrement. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

Non, il n’est pas utilisé dans la méthode

5) **(2 points)** Implémentez la fonction caesar\_encrypt\_asm qui utilise un chiffrement par décalage (chiffrement de César[[1]](#footnote-1)) pour encrypter une chaine de caractères src en utilisant une clé non nulle key. La chaine de caractères encryptée est retournée dans dst. Votre implémentation doit appeler les fonctions is\_lower\_case\_asm et is\_upper\_case\_asm. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

Non, il n’est pas utilisé dans la méthode

6) **(1 point)** Implémentez la fonction caesar\_decrypt\_asm qui utilise un chiffrement par décalage (chiffrement de César) pour décrypter une chaine de caractères src en utilisant une clé non nulle key. La chaine de caractères decryptée est retournée dans dst. Votre implémentation doit appeler les fonctions is\_lower\_case\_asm et is\_upper\_case\_asm. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

Non, il n’est pas utilisé dans la méthode

7) **(2 points)** Implémentez la fonction vigenere\_encrypt\_asm qui utilise un chiffrement de Vigenère[[2]](#footnote-2) pour encrypter une chaine de caractères src en utilisant une mot-clé keyword. La chaine de caractères encryptée est retournée dans dst. Votre implémentation doit appeler les fonctions is\_lower\_case\_asm et is\_upper\_case\_asm. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

8) **(1 point)** Implémentez la fonction vigenere\_decrypt\_asm qui utilise un chiffrement de Vigenère pour décrypter une chaine de caractères src en utilisant une mot-clé keyword. La chaine de caractères decryptée est retournée dans dst. Votre implémentation doit appeler les fonctions is\_lower\_case\_asm et is\_upper\_case\_asm. Indiquez ci-après si vous avez sauvegardé %ebp et pourquoi.

9) **Bonus** **(0.5 point)** Le programme a déterminé une phrase pour vous. Sachant que la phrase a été encodée avec un chiffrement de César, écrivez une fonction qui permet de la décoder par force brute. Vous pouvez écrire la fonction directement en C (pas d’ASM). Donnez ci-après la phrase décodée et la clé du chiffrement.

Phrase décodée : De baiser sur les lèvres, clé : 20

1. Voir <https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage>. [↑](#footnote-ref-1)
2. Voir <https://fr.wikipedia.org/wiki/Chiffre_de_Vigen%C3%A8re>. [↑](#footnote-ref-2)