# IFT209 – Programmation système

Université de Sherbrooke

# Devoir 4

| | |
|---|---|
| Enseignant: | Michael Blondin |
| Date de remise: | mercredi 31 mars 2021 à 23h59 |
| À réaliser: | en équipe de deux |
| Modalités: | remettre en ligne sur Turnin |
| Pointage: | sur 20 points + 1,5 point bonus |

Ce devoir cherche à mettre en pratique la manipulation de chaînes de bits et de caractères, ainsi que la mise au point de sous-programmes.

**Problème.** Afin de mieux comprendre le fonctionnement interne des librairies de chaînes de caractères, cherchons à implémenter diverses manipulations de chaînes de caractères. Lorsque votre programme est exécuté, il doit lire une chaîne de caractères $s$ au clavier, puis un entier $n$ compris entre $0$ et $5$ (inclusivement). L'opération $n$ est ensuite effectuée sur $s$, puis le programme se termine. Vous devez implémenter ces six opérations:

| | **Taille d'une chaîne** |
|---|---|
| **Opération** | 0 |
| **Entrée** | Chaîne de caractères $s$ sous codage UTF-8 |
| **Effet** | Affiche le nombre de caractères (non nuls) de $s$ |
| **Tests** | `abcdef` $\rightarrow$ `6`<br>`é` $\rightarrow$ `1`<br>`aéケ▶` $\rightarrow$ `4` |

| **Casses et substitutions** | |
|---|---|
| **Opération** | 1 |
| **Entrée** | Chaîne de caractères $s$ sous codage ASCII |
| **Effet** | Affiche la chaîne obtenue en appliquant ces opérations à $s$:<br>• Les lettres aux positions paires sont mises en minuscule et les lettres aux positions impaires sont mises en majuscule (la première position est 0);<br>• Ces voyelles doivent être remplacées par ces chiffres:<br>A $\mapsto$ 4 a $\mapsto$ 4<br>E $\mapsto$ 3 e $\mapsto$ 3<br>I $\mapsto$ 1 i $\mapsto$ 1<br>O $\mapsto$ 0 o $\mapsto$ 0<br>• Les autres caractères sont inchangés. |
| **Tests** | `brUN` → `bRuN`<br>`bonjour!` → `b0nJ0Ur!`<br>`Ceci est une phrase formidable` → `c3c1 3sT Un3 PhR4S3 f0rM1D4Bl3` |

| **Hexadécimal vers décimal** | |
|---|---|
| **Opération** | 2 |
| **Entrée** | Chaîne de caractères $s$ sous codage ASCII représentant un nombre hexadécimal non signé préfixé par « `0x` » (valeur comprise entre $0$ et $2^{64}-1$ inclusivement) |
| **Effet** | Affiche la valeur décimale de $s$ |
| **Tests** | `0x5` → `5`<br>`0x0A` → `10`<br>`0xFF` → `255`<br>`0xABCDEF98` → `2882400152` |

| **Binaire vers décimal** | |
|---|---|
| **Opération** | 3 |
| **Entrée** | Chaîne de caractères $s$ sous codage ASCII représentant un entier signé binaire préfixé par « `0b` » (valeur comprise entre $-2^{63}$ et $2^{63}-1$ inclusivement) |
| **Effet** | Affiche la valeur décimale de $s$ |
| **Tests** | `0b0101` → `5`<br>`0b00111` → `7`<br>`0b10100` → `-12`<br>`0b111111` → `-1` |

| **Chiffrement par décalage** | |
|---|---|
| **Opération** | 4 |
| **Entrée** | Chaîne de caractères $s$ sous codage ASCII dont les caractères sont parmi: A, B, C, D, E, F, H, I, J, K, L, M, N, P, Q, R, S, T, U, V, X, Y, Z, [, \ et ] |
| **Effet** | La chaîne $s$ a été chiffrée à partir d'une chaîne $t$. Vous devez afficher la chaîne $t$; autrement dit, vous devez *déchiffrer* $s$.<br><br>La chaîne $t$ est constituée exclusivement de lettres majuscules (A à Z). La chaîne $s$ a été obtenue à partir de $t$ en appliquant les deux transformations consécutives suivantes à chacune des lettres de $t$:<br><br>(i) La lettre est décalée circulairement de 7 positions vers l'avant dans l'alphabet:<br>$\begin{array}{ccc} \mathsf{A} & \mapsto & \mathsf{H} \\ \mathsf{B} & \mapsto & \mathsf{I} \\ \vdots & \mapsto & \vdots \\ \mathsf{Y} & \mapsto & \mathsf{F} \\ \mathsf{Z} & \mapsto & \mathsf{G} \end{array}$<br>(ii) Les 5 bits de poids faible du caractère obtenu à l'étape (i) sont décalés circulairement de 3 bits vers la gauche.<br><br>Exemple de la transformation:<br><br>(i) $\mathsf{C} \mapsto \mathsf{J}$<br>(ii) $\mathsf{J} = 01001010_2 \mapsto 01010010_2 = \mathsf{R}$<br><br>*Attention*: vous devez effectuer la procédure *inverse* afin de retrouver $t$ |
| **Tests** | `R → C`<br>`RRR → CCC`<br>`HU]VCRNCH → TOPSECRET` |

| **Permutations** | |
|---|---|
| **Opération** | 5 |
| **Entrée** | Chaîne de caractères $s$ sous codage ASCII sans répétition de caractères |
| **Effet** | Affiche toutes les permutations de $s$ (dans l'ordre de votre choix)<br><br>*Indice: pensez à une procédure récursive.* |
| **Tests** | `ab → ab ba`<br>`abc → abc acb bac bca cba cab` |

**Directives.**

— Votre programme doit être obtenu en complétant le code partiel ci-dessous;
— Votre programme doit être remis dans un seul fichier nommé `devoir4.s`;
— Ne modifiez pas le point d'entrée ainsi que le format de la chaîne en entrée;
— Supposez que la chaîne $s$ respecte le format de l'opération choisie (aucune validation);
— N'utilisez *pas* `printf`, `scanf` ou d'autres fonctions d'une librairie afin d'implémenter les opérations.

**Pointage.** Vous pouvez obtenir jusqu'à 20 points répartis ainsi:

— 1 point pour la lecture d'une chaîne de caractères et d'un code d'opération, et l'affichage d'un résultat;
— 1 point pour chaque opération qui passe les tests donnés plus haut (donc 6 points au maximum);
— 1,5 points par opération bien implémentée (donc 9 points au maximum);
— 2 points pour l'indentation du code (codes d'opération, opérandes et commentaires alignés);
— 2 points pour la qualité et lisibilité du code (commentaires significatifs, usage des registres, organisation du code, pas de « code spaghetti », etc.)

**Bonus.** Vous obtenez 0,75 point bonus pour chacune de ces fonctionnalités additionnelles:

— l'opération 3 est implémentée sous forme de *sous-programme* avec *au plus dix* instructions (excluant étiquettes/commentaires), *aucune* instruction arithmétique, *au plus une* occurrence de `cbz`, *au plus une* occurrence de `b`, et *aucune* autre instruction de branchement (`b.cond`, `bl`, `blr`, `cbnz`, `tbz`, `tbnz`, etc.)
— l'opération 5 supporte les caractères UTF-8 (par exemple: `aéケ` → `aéケ aケé éaケ éケa ケéa ケaé`)

**Code partiel.**

```
.include "macros.s"
.global  main

main:
    adr     x0, fmtLecture
    adr     x1, chaine
    bl      scanf

    mov     x0, 0
    bl      exit

.section ".data"
// Mémoire allouée pour une chaîne de caractères d'au plus 1024 octets
chaine:     .skip   1024

.section ".rodata"
// Format pour lire une chaîne de caractères d'une ligne (incluant des espaces)
fmtLecture: .asciz  "%[^\n]s"
```