TD n°6 : Cryptographie

1 Conversion entier - caractère

Dans un ordinateur, un caractère est représenté par un entier naturel, cette correspondance étant bijective. Par exemple, lorsqu'on utilise la table ASCII, les caractères "A" et "m" sont respectivement codés par les entiers 65 et 109. Commencez par télécharger cette table disponible sur le site de la mp1.

Nous allons dans ce TP uniquement nous intéresser aux caractères dont les codes ASCII sont compris entre 32 et 122. Il s'agit d'un ensemble E de $n = 91$ caractères qui commence par l'espace " " et se termine par la lettre minuscule "z". Les caractères accentués en sont exclus et nous ne pourrons pas les utiliser pour écrire un texte. En revanche, l'apostrophe " ' " (code ASCII 39), la virgule " , " (code 44) ou le point " . " (code 46) pourront être utilisés puisqu'ils font partie de E.

Afin de rendre les opérations sur les chaînes de caractères plus simples, nous souhaiterions réaliser la correspondance bijective suivante de E $\to [\![0, 90]\!]$ selon :

" " $\mapsto$ 0 , " | " $\mapsto$ 1 , ... etc..., " y " $\mapsto$ 89 , " z " $\mapsto$ 90

1. Que font les fonctions `chr(n)` et `ord(c)` ? On testera par exemple `chr(97)`, `chr(122)`, `ord("a")` et `ord("z")`.
2. Écrire une fonction `convCarToInt(c)` qui convertit un caractère `c` $\in$ E en un entier $n \in [\![0, 90]\!]$ selon la correspondance bijective ci-dessus. Cette fonction retourne `n`.
3. Inversement, écrire une fonction `convIntToCar(n)` qui retourne le caractère `c` $\in$ E associé à l'entier `n` $\in [\![0, 90]\!]$
4. Écrire une fonction `listOfIntToStr(L)` qui convertit une liste `L` d'entiers compris entre 0 et 90 en une chaîne de caractère. Cette fonction doit retourner la chaîne de caractères.
5. Écrire une fonction `strToListOfInt(ch)` qui retourne une liste d'entiers associés à la chaîne de caractères `ch`.

   Cette correspondant bijective va nous permettre de définir une addition $\underset{E}{+}$ et une soustraction $\underset{E}{-}$ dans l'ensemble E (ces deux opérations devant se faire modulo 91 bien sûr).

6. Écrire une fonction `somme(car1, car2)` qui renvoie le caractère obtenu en additionnant deux caractères `car1` et `car2`.
7. Écrire une fonction `difference(car1, car2)` qui renvoie le caractère obtenu en soustrayant le caractère `car2` au caractère `car1`, c'est à dire `c = car1` $\underset{E}{-}$ `car2`.

2 Chiffrement de César

Il s'agit d'une méthode de cryptographie attribuée à Jules César et basée sur la substitution d'une lettre par une autre. Caque lettre d'un **texte clair** est remplacée par une autre, obtenue par *décalage* d'un nombre fixe p de lettres, appelé **clé de chiffrement** (la clé sera donc comprise entre 1 et 90 pour notre texte).

Par exemple, avec une clé de chiffrement égale à 3, nous avons " a " $\mapsto$ " d", " b " $\mapsto$ " e ", etc... (ce décalage devant se faire modulo 91 aussi).

1. Écrire une fonction `chiffrementCesar(texteClair, cle)` qui crypte la chaîne de caractères `texteClair` à l'aide d'un entier `cle` selon la méthode de chiffrement de César. Cette fonction renvoie la chaîne de caractères cryptée.

2. Écrire la fonction `dechiffreCesar(texteCrypte, cle)` qui décrypte le texte `texteCrypte` en ayant connaissance de la `cle`.
3. Application : décrypter le texte qui figure au début de votre fichier programme.

Le chiffrement de César est une méthode qui résiste mal à la cryptanalyse : le petit nombre de clés (90 ici) laisse la possibilité de tester toutes les clés possibles jusqu'à obtenir le texte clair.

De plus, si le texte est suffisamment long, le chiffrement de César est sensible à l'analyse fréquentielle : les lettres de l'alphabet français ont des fréquences d'apparition bien distinctes, si bien qu'après avoir identifié quelque unes des lettres les plus fréquentes (deux ou trois suffisent), il est facile de déterminer la clé.

3 Chiffrement de Vigenère

Le chiffrement de Vigenère consiste à choisir une chaîne de caractères de taille m inférieure à celle du texte à crypter et qui fera office de **clé**. On découpe le texte clair en blocs de m caractères auxquels on additionne chacun des caractères de la **clé**.

Par exemple, en choisissant la clé `"cpge"`, le chiffrement d'un texte par la méthode de Vigenère du texte `"Totalement anticonstitutionnel"` consiste en l'opération suivante :

Texte clair	Tota	leme	nt a	ntic	onst	itut	ionn	el
+ clé	cpge	cpge	cpge	cpge	cpge	cpge	cpge	cp
Texte crypté	<d‘K	TZYO	VigK	ViUM	Wc_^	Qia^	QdZX	Ma

On pourra remarquer que la taille du texte clair n'est pas nécessairement un multiple du nombre m de caractères de la clé.

1. Si la clé est de taille m et le texte de taille n, combien de fois devra-t-on appliquer la clé en entier ? Combien de caractères resteront à la fin du texte (pour lesquels la clé ne sera pas appliquée dans sa totalité) ?
2. Écrire une fonction `cryptoVigenere(texteClair, cle)` qui encode la chaîne de caractères `texteClair` à l'aide de la chaîne de caractères `cle`.
3. Écrire une fonction `dechiffreVigenere(texteCode, cle)` qui décode la chaîne de caractères `texteCode` à l'aide de la chaîne de caractères `cle`. Vous utiliserez les fonctions écrites dans la partie 1.
4. Vérification : décryptez la chaîne de caractères qui figure au début de votre programme, la **clé** étant "cpge".