Cryptographie

Tu vas jouer le rôle d'un espion qui intercepte des messages secrets et tente de les décrypter.

Cours 1 (Chiffre de César)**.**
Le ***chiffre de César*** est une façon simple de coder un message afin de conserver le secret du contenu jusqu'à son destinataire. Il s'agit tout simplement de décaler chaque lettre du message. Voyons l'exemple d'un décalage de trois lettres : **A** devient **D** ; **B** devient **E** ; etc.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	lettres du message en clair
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C	lettres du message codé

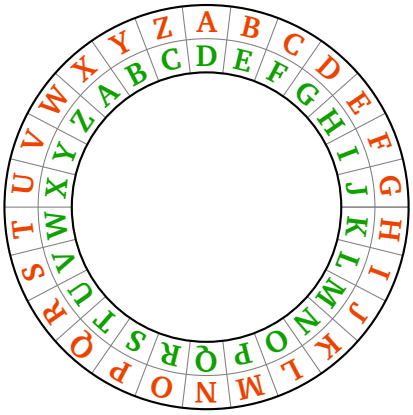
Par exemple le message :

CAPTUREZ IDEFIX

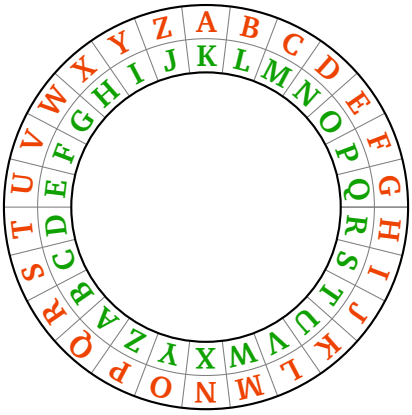
se chiffre en :

FDSWXUHC LGHILA

Une autre façon de présenter le décalage est de placer les alphabets en clair (anneau extérieur en rouge) et codé (anneau intérieur en vert) sur deux roues concentriques. À gauche un décalage avec $k = 3$ et à droite un décalage avec $k = 10$.



Décalage $k = 3$



Décalage $k = 10$

Pour chiffrer des messages tu passes des lettres rouges à l'extérieur aux lettres vertes à l'intérieur. Pour déchiffrer des messages il suffit de faire l'opération inverse : passer des lettres vertes aux lettres rouges !
Déchiffre à la main les messages suivants :

- **EORTXHC DVWHULA** chiffré avec un décalage $k = 3$.
- **YE OCD ZKXYBKWSH** chiffré avec un décalage $k = 10$.

Cours 2 (Passer d'un caractère à un nombre et inversement)**.**
On préfère travailler avec des nombres qu'avec des lettres !

- **Alphabet.** Tu vas définir une constante globale `Alphabet` qui contient toutes les lettres de l'alphabet :
  `Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"`
- **Numérotation.** On numérote chaque lettre : 0 pour **A**, 1 pour **B**, 2 pour **C**, ..., 25 pour **Z**.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- **D'un numéro vers une lettre.** Pour récupérer la lettre correspondant au rang i, il suffit d'écrire :
  `Alphabet[i]`
  Par exemple `Alphabet[2]` vaut `'C'`.
- **D'une lettre vers son numéro.** Pour récupérer le rang d'une lettre, notée `car`, il suffit d'écrire :
  `Alphabet.index(car)`
  Par exemple `Alphabet.index('D')` vaut 3.
- **César.** Le chiffrement de César de décalage k correspond juste à une addition : le numéro j du caractère chiffré est égal au numéro i du caractère en clair auquel on ajoute le décalage k, le tout modulo 26 :
  $$j = (i + k) \ \% \ 26$$
  Par exemple avec un décalage $k = 3$:
  — le caractère numéro 0 (**A**) est chiffré en le caractère numéro $0 + 3 = 3$ (**D**),
  — le caractère numéro 1 (**B**) est chiffré en le caractère numéro $1 + 3 = 4$ (**E**),
  — ...
  — le caractère numéro 25 (**Z**) est chiffré en le caractère numéro $25 + 3 = 28$ qui vaut 2 modulo 26 (c'est donc bien **C**).

Activité 1 (Chiffre de César)**.**

Objectifs : programmer le chiffrement et le déchiffrement du chiffre de César.

1. Programme une fonction `chiffre_cesar_caractere(car,k)` qui renvoie le caractère de l'alphabet situé k rang après (modulo 26).
   Indications.
   - Récupère le rang i dans `Alphabet` du caractère à chiffrer.
   - Ajoute k modulo 26 par la formule : $j = (i + k)$ % 26.
   - Renvoie le caractère correspondant au rang j de `Alphabet`.

   Exemple. Avec un décalage de $k = 3$, **A** devient **D** et **Z** devient **C**.
2. Programme une fonction `chiffre_cesar_phrase(phrase,k)` qui renvoie la phrase codée par un décalage de César k.
   Indication. Si un caractère n'est pas une lettre majuscule alors conserve-le tel quel dans la phrase chiffrée. Le test se fait par :
   `if car not in Alphabet:`
   Exemple. Avec un décalage de $k = 3$, **CAPTUREZ IDEFIX !** devient **FDSWXUHC LGHILA !**
3. Programme une fonction `dechiffre_cesar_phrase(phrase,k)` pour le déchiffrement.
   Indication. Deux solutions : soit tu recommences tout, mais au lieu d'ajouter k (modulo 26) tu soustrais k (modulo 26) ou bien tu chiffres le message avec un décalage de $-k$ (au lieu de $+k$).
   Vérifie que, après avoir chiffré un message, tu le retrouves par déchiffrement !

4. Tu te places dans la peau d'un espion qui a intercepté un message codé par un chiffre de César, mais qui ne connaît pas la clé k. Programme une fonction `attaque_cesar(phrase)` qui affiche les déchiffrements, en testant toutes les clés k possibles.

   Tu as intercepté le message envoyé par le camp Babaorum à Jules César :

   HSFGJSEAP F S HDMK VW HGLAGF

   Que va maintenant faire César ?

Cours 3 (Chiffrement par substitution)**.**
Le chiffre de César est trop facile à attaquer, même si on ne connaît pas le décalage. Pour compliquer la tâche d'un espion, on introduit le ***chiffrement par substitution***. À chaque lettre de l'alphabet en clair (ici en rouge, en majuscule), on associe une lettre au hasard (ici en vert, en minuscule). Voici un exemple :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
y	k	c	o	d	m	f	j	g	z	a	x	r	n	b	u	t	q	i	p	h	w	e	s	v	l

- **Chiffrement.** On remplace chaque caractère du message par sa lettre substituée. Le message **BONJOUR** devient **kbnzbhq**. Exercice : chiffre la phrase **AU REVOIR**.
- **Déchiffrement.** On fait l'opération inverse : **kbnkbn** donne **BONBON**. Exercice : déchiffre **ywd cdiyq**.
- Bien évidemment pour les lettres d'arrivée on aurait pu choisir un autre mélange. L'expéditeur et le destinataire du message doivent au préalable se mettre d'accord sur la substitution choisie.

Activité 2 (Chiffrement par substitution)**.**

Objectifs : programmer le codage et le décodage d'un chiffrement par substitution.

On se donne une substitution par un alphabet de départ suivi de son remplacement. On prendra l'exemple suivant, défini par deux constantes globales :

```
Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
 Melange = "ykcodmfjgzaxrnbutqiphwesvl"
```

1. Programme une fonction `chiffre_substitution_caractere(car)` qui renvoie le caractère de l'alphabet chiffré par substitution.

   Indications.
   - Récupère le rang i dans `Alphabet` du caractère à chiffrer.
   - Renvoie le caractère correspondant au rang i de `Melange`.

   Exemple. **A** devient **y** et **B** devient **k**.
2. Programme une fonction `chiffre_substitution_phrase(phrase)` qui renvoie la phrase codée par substitution.

   Exemple. Après substitution **PAS DE POTION POUR OBELIX !** devient **uyi od ubpgbn ubhq bkdxgs !**
3. Programme une fonction `dechiffre_substitution_phrase(phrase)` pour le déchiffrement.

Cours 4 (Attaque statistique)**.**
Pour un espion qui intercepterait un message codé, sans connaître la substitution choisie, il n'est plus possible de tester toutes les possibilités. En effet, il y a $26 \times 25 \times 24 \times \cdots \times 2 \times 1$ choix possibles pour l'alphabet mélangé, ce qui fait environ 4×10^{26} clés !

Statistiques.
Pour un texte assez long, les lettres n'apparaissent pas toutes avec la même fréquence. En français, les lettres les plus rencontrées sont dans l'ordre :

E S A I N T R U L O D C P M V Q G F H B X J Y Z K W

La *fréquence d'apparition d'une lettre* est donnée par la formule :

$$\text{fréquence d'apparition d'une lettre} = \frac{\text{nombre d'occurrences de la lettre}}{\text{nombre total de lettres}} \times 100$$

Dans un texte en français les fréquences sont proches de :

E	S	A	I	N	T	R	U	L	O	D
14.69%	8.01%	7.54%	7.18%	6.89%	6.88%	6.49%	6.12%	5.63%	5.29%	3.66%

Attaque.
On a intercepté un message, mais on ne connaît pas la substitution. Comment utiliser les statistiques pour décrypter le message ? Voici une méthode d'attaque : dans le texte chiffré, on cherche la lettre qui apparaît le plus, et si le texte est assez long, cela devrait être le chiffrement du **E**, la lettre qui apparaît ensuite dans l'étude des fréquences devrait être le chiffrement du **S**, puis le chiffrement du **A**, **I**, **N**, **T**...
On obtient ainsi un déchiffrement partiel du message, sous la forme d'un texte à trous et il faut ensuite deviner les lettres manquantes.

Un exemple.
Par exemple, déchiffrons la phrase :

jm dw ug jddbhbwm y jmlj cjtljtdjd

On compte les apparitions des lettres :

j : 7 **d** : 5 **m** : 3 **b**, **l**, **t**, **w** : 2

On suppose donc que le **j** crypte la lettre **E**, le **d** la lettre **S**, ce qui donne :

E* S* ** ESS*** * E**E *E**E*SES**

Ensuite la lettre qui apparaît le plus est le **m**. D'après les fréquences, elle devrait correspondre à **A**, **I**, **N** ou **T**. Ainsi le premier mot serait **EA**, **EI**, **EN** ou **ET**. Seuls les deux derniers sont des mots valides.
Si **m** → **N**, la phrase se déchiffre en :

EN S* ** ESS**N * EN*E *E**E*SES**

ce qui n'est pas très clair, alors qu'avec **m** → **T** c'est mieux !

ET S* ** ESS**T * ET*E *E**E*SES**

En cherchant où placer les lettres les plus fréquentes suivantes (**A**, **I**, **N**) puis les autres, avec un peu de patience et de bon sens, on décrypte le message :

ET SI ON ESSAYAIT D ETRE HEUREUSES

Activité 3 (Attaque statistique).

Objectifs : utiliser la fréquence d'apparition des lettres pour décrypter un message en essayant de deviner la substitution.

1. **Substitution.** Programme une fonction :

   `substitution(phrase,Alphabet_depart,Alphabet_arrivee)`

   qui substitue dans la phrase donnée les lettres de l'alphabet de départ par celles de l'alphabet d'arrivée.

   Indications.
   - C'est presque la même fonction que l'activité précédente, sauf qu'ici on a plus de souplesse sur les alphabets qui sont passés en paramètres.
   - En particulier `substitution(phrase,Alphabet,Melange)` fait la

même chose que `chiffre_substitution_phrase(phrase)` alors que `substitution(phrase,Melange,Alphabet)` fait la même chose que `dechiffre_substitution_phrase(phrase)`.

- En particulier la fonction `substitution()` doit permettre un déchiffrement partiel d'une phrase. Voir l'exemple ci-dessous.

Exemple (Utilisation pour un décryptage partiel.)**.**

- On veut décrypter :

  jdolwt jd tm vb ?

- Admettons que l'on ait identifié que **j** codait **E**, **d** codait **S**, **m** codait **T**.
- On définit donc un alphabet de départ et celui (partiel) d'arrivée :

```
Alphabet_depart = "abcdefghijklmnopqrstuvwxyz"
Alphabet_arrivee = "...S.....E..T............."
```

- Alors avec `phrase = "jdolwt jd tm vb ?"`, la commande

  `substitution(phrase,Alphabet_depart,Alphabet_arrivee)`

  renvoie :

  `"ES...T ES T. .. ?"`

- Il reste à deviner **ESPRIT ES TU LA ?**

2. **Statistiques.**

   Programme une fonction `statistiques(phrase)` qui calcule et renvoie le nombre d'apparitions des lettres dans une phrase.

   Indications.

   - On suppose que la phrase est écrite en minuscules, on ne tient pas compte des autres caractères.
   - La fonction renvoie la liste des nombres correspondant à chaque lettre de l'alphabet. Par exemple $[3, 0, 2, \ldots]$ signifie qu'il y a 3 lettres **a**, pas de lettres **b**, 2 lettres **c**, etc.

   Déduis-en une fonction `affiche_statistiques(phrase)` qui affiche proprement les lettres qui apparaissent dans la phrase avec leur nombre d'apparitions.

3. **Fréquences.** Modifie tes fonctions précédentes pour écrire deux fonctions `frequences(phrase)` et `affiche_frequences(phrase)` qui calculent et affichent les fréquences d'apparitions des lettres (on ne tient compte que des lettres en minuscules).

4. **Énigmes.** Essaie de décrypter les trois citations suivantes. Chacune a été chiffrée par une substitution différente.

   Les frères Goncourt :

   ay dmymndmnxlv vdm ay shvjnvhv fvd dznvgzvd ngvcyzmvd

   Charles Darwin :

   apy pywpfpy tdv ydjsvspng np ybng woy apy pywpfpy apy wady lbjgpy nv apy wady vngpaavzpngpy movy fpaapy tdv y okowgpng ap mvpdc odc fionzpmpngy

   Albert Einstein :

   kw yjnzfcn, i nmy lqwpx zp mwcy yzqy ny lqn fcnp pn azpiyczppn. kw ofwyclqn, i nmy lqwpx yzqy azpiyczppn ny lqn onfmzppn pn mwcy ozqflqzc. cic, pzqm wszpm fnqpc yjnzfcn ny ofwyclqn : fcnp pn azpiyczppn ny onfmzppn pn mwcy ozqflqzc !

Cours 5 (Chiffrement de Vigenère)**.**
Un des principaux défauts du chiffre de César (et du chiffrement par substitution) est qu'une lettre (par exemple **A**) est toujours chiffrée par la même lettre (par exemple **D**). Le chiffrement de Vigenère est une version améliorée du chiffre de César.
On regroupe d'abord les lettres de notre message par blocs, par exemple ici par blocs de longueur 3 :

IL ETAIT UNE FOIS

devient

ILE TAI TUN EFO IS

(les espaces sont purement indicatifs, dans la première phrase ils séparent les mots, dans la seconde ils séparent les blocs).
Si n est la longueur d'un bloc, alors on choisit une clé constituée de n nombres de 0 à 25 : $[k_1, k_2, \dots, k_n]$.
Le chiffrement consiste à effectuer un chiffrement de César, dont le décalage dépend du rang de la lettre dans le bloc :

- un décalage de k_1 pour la première lettre de chaque bloc,
- un décalage de k_2 pour la deuxième lettre de chaque bloc,
- ...
- un décalage de k_n pour la n-ème et dernière lettre de chaque bloc.

Pour notre exemple, si on choisit comme clé $[4, 2, 3]$ alors pour le premier bloc « **ILE** » :

- un décalage de 4 pour **I** donne **M**,
- un décalage de 2 pour **L** donne **N**,
- un décalage de 3 pour **E** donne **H**,

Ainsi « **ILE** » devient « **MNH** ». On recommence avec le bloc « **TAI** » qui devient « **XCL** ». Le chiffrement complet donne :

MNH XCL XWQ IHR MU

autrement dit la phrase chiffrée est :

MN HXCLX WQI HRMU

Exercice. Chiffre la phrase **UNE PRINCESSE GEEK** avec le chiffrement de Vigenère et la même clé $[4, 2, 3]$.

Déchiffrement. Pour déchiffrer, il suffit d'inverser la clé. C'est-à-dire que c'est la même procédure que le chiffrement mais avec la clé $[-k_1, -k_2, \dots, -k_n]$. Par exemple l'opposé de la clé $[4, 2, 3]$ est la clé $[-4, -2, -3]$. Comme on travaille modulo 26, cette dernière clé vaut aussi $[22, 24, 23]$.
Exercice. Déchiffre la phrase **QKOSW HX VLRVLR**, chiffrée avec la même clé $[4, 2, 3]$.

Activité 4 (Chiffrement de Vigenère)**.**

Objectifs : programmer le chiffrement de Vigenère et éventuellement trouver une attaque.

Programme une fonction `chiffre_vigenere(phrase,cle)` qui chiffre la phrase donnée selon le chiffrement de Vigenère, pour la clé donnée sous la forme d'une liste de décalages, `cle` $= [k_1, k_2, \dots, k_n]$.

Exemple. La phrase **AAA ABC** avec la clé $[1, 2, 3]$ renvoie la phrase **BCD BDF**.

Indications.

- Il faut décaler chaque lettre selon un décalage k_i comme pour le chiffre de César.
- Pour savoir quel indice i convient, tu peux utiliser un compteur i, initialisé à 0, auquel tu ajoutes 1 à chaque lettre de l'alphabet rencontrée, compteur qui prend ses valeurs modulo n (qui est la longueur de la clé).

Bonus. Attaque du chiffrement de Vigenère.
Essaie de décrypter le message suivant (et trouve son auteur) qui a été codé par un chiffrement de Vigenère avec une clé (inconnue !) de longueur 4.

DL ZHGIVUEL OD UL LQK TYDVL OIL XEU DLC YEIOSASOI KVXJ KBBI WA PYWYC T WBH QDVBI IBO BWZ QUFZ SDLLVBANODLZ CEFA OCHSSI VL NEMAOI