IFT 1015 – Programmation 1

TP 1

- À faire en groupe de **deux** étudiants
- Remise le 23 mars à **23 :59** au plus tard

1. Objectif du TP

Le but du TP est d'implémenter trois algorithmes de cryptographie historiques, du plus ancien au plus récent :

- Le premier sera le code César
- Le suivant sera le code Vigenère
- Le dernier sera un code utilisant le binaire et xor

Dans le fichier de départ, trois constantes sont fournies :

- Message : La vie est belle
- ClefCesar : 5
- ClefAutre : secret

Les résultats obtenus sont les suivants :

- César : Qf anj jxy gjqqj
- Vigenère : De mmx iuk uwpnv
- Xor : ? C S␀ E (peut varier selon les implémentations)

Notez que pour César et Vigenère, la ponctuation et les espaces sont respectés, c'est-à-dire qu'ils apparaissent sans modification dans le message crypté. Ces informations sont là pour valider votre code. Pour Xor, les espaces et les signes de ponctuation existent dans le tableau ASCII et seront donc aussi codés.
Vous êtes encouragés à tester vos propres messages et vos propres clefs car des messages et des clefs différentes seront utilisées pour la correction.

2. Introduction

2.1 Le code César

Ce code consiste à décaler les lettres d'un message d'une certaine valeur appelée la clef. Si la clef est 5 alors le « a » devient « f » (décalage de 5 lettres). Le « w » devient « b » (on boucle pour revenir au début de l'alphabet).

Ce code a été décrypté au 15ème siècle par un mathématicien britannique en utilisant la fréquence des lettres dans chacune des langues.

2.2 Le code Vigenère

Le code Vigenère utilise comme clef un mot, par exemple « abri ». Il utilise une table de correspondance (voir ci-dessous). Ainsi si le message est « leci » alors on choisit le « L » sur la ligne supérieure puis le « A » (abri) sur la colonne de gauche et on trouve la lettre à l'intersection : « L ».
Pour le « E », on choisit le « E » sur la ligne supérieure puis le « B » (2ème lettre de abri) sur la colonne de gauche et on prend l'intersection : « F ».
Même chose pour le « C » avec le « R » de abri qui donne « T ». Et enfin le « I » combiné au « I » de abri, donne « Q ».
« LECI » donne donc « LFTQ ».
S'il y a des lettres supplémentaires dans le message, alors on recommence au début de la clef.

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

2.3 Implémentation de l'algorithme basé sur la représentation binaire et l'opération XOR

On utilise la valeur binaire des caractères ASCII (voir tableau ci-dessous).

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*	48	30	110000	60	**0**	96	60	1100000	140	`
1	1	1	1	*[START OF HEADING]*	49	31	110001	61	**1**	97	61	1100001	141	**a**
2	2	10	2	*[START OF TEXT]*	50	32	110010	62	**2**	98	62	1100010	142	**b**
3	3	11	3	*[END OF TEXT]*	51	33	110011	63	**3**	99	63	1100011	143	**c**
4	4	100	4	*[END OF TRANSMISSION]*	52	34	110100	64	**4**	100	64	1100100	144	**d**
5	5	101	5	*[ENQUIRY]*	53	35	110101	65	**5**	101	65	1100101	145	**e**
6	6	110	6	*[ACKNOWLEDGE]*	54	36	110110	66	**6**	102	66	1100110	146	**f**
7	7	111	7	*[BELL]*	55	37	110111	67	**7**	103	67	1100111	147	**g**
8	8	1000	10	*[BACKSPACE]*	56	38	111000	70	**8**	104	68	1101000	150	**h**
9	9	1001	11	*[HORIZONTAL TAB]*	57	39	111001	71	**9**	105	69	1101001	151	**i**
10	A	1010	12	*[LINE FEED]*	58	3A	111010	72	**:**	106	6A	1101010	152	**j**
11	B	1011	13	*[VERTICAL TAB]*	59	3B	111011	73	**;**	107	6B	1101011	153	**k**
12	C	1100	14	*[FORM FEED]*	60	3C	111100	74	**<**	108	6C	1101100	154	**l**
13	D	1101	15	*[CARRIAGE RETURN]*	61	3D	111101	75	**=**	109	6D	1101101	155	**m**
14	E	1110	16	*[SHIFT OUT]*	62	3E	111110	76	**>**	110	6E	1101110	156	**n**
15	F	1111	17	*[SHIFT IN]*	63	3F	111111	77	**?**	111	6F	1101111	157	**o**
16	10	10000	20	*[DATA LINK ESCAPE]*	64	40	1000000	100	**@**	112	70	1110000	160	**p**
17	11	10001	21	*[DEVICE CONTROL 1]*	65	41	1000001	101	**A**	113	71	1110001	161	**q**
18	12	10010	22	*[DEVICE CONTROL 2]*	66	42	1000010	102	**B**	114	72	1110010	162	**r**
19	13	10011	23	*[DEVICE CONTROL 3]*	67	43	1000011	103	**C**	115	73	1110011	163	**s**
20	14	10100	24	*[DEVICE CONTROL 4]*	68	44	1000100	104	**D**	116	74	1110100	164	**t**
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*	69	45	1000101	105	**E**	117	75	1110101	165	**u**
22	16	10110	26	*[SYNCHRONOUS IDLE]*	70	46	1000110	106	**F**	118	76	1110110	166	**v**
23	17	10111	27	*[END OF TRANS. BLOCK]*	71	47	1000111	107	**G**	119	77	1110111	167	**w**
24	18	11000	30	*[CANCEL]*	72	48	1001000	110	**H**	120	78	1111000	170	**x**
25	19	11001	31	*[END OF MEDIUM]*	73	49	1001001	111	**I**	121	79	1111001	171	**y**
26	1A	11010	32	*[SUBSTITUTE]*	74	4A	1001010	112	**J**	122	7A	1111010	172	**z**
27	1B	11011	33	*[ESCAPE]*	75	4B	1001011	113	**K**	123	7B	1111011	173	**{**
28	1C	11100	34	*[FILE SEPARATOR]*	76	4C	1001100	114	**L**	124	7C	1111100	174	**|**
29	1D	11101	35	*[GROUP SEPARATOR]*	77	4D	1001101	115	**M**	125	7D	1111101	175	**}**
30	1E	11110	36	*[RECORD SEPARATOR]*	78	4E	1001110	116	**N**	126	7E	1111110	176	**~**
31	1F	11111	37	*[UNIT SEPARATOR]*	79	4F	1001111	117	**O**	127	7F	1111111	177	*[DEL]*
32	20	100000	40	*[SPACE]*	80	50	1010000	120	**P**					
33	21	100001	41	**!**	81	51	1010001	121	**Q**					
34	22	100010	42	**"**	82	52	1010010	122	**R**					
35	23	100011	43	**#**	83	53	1010011	123	**S**					
36	24	100100	44	**$**	84	54	1010100	124	**T**					
37	25	100101	45	**%**	85	55	1010101	125	**U**					
38	26	100110	46	**&**	86	56	1010110	126	**V**					
39	27	100111	47	**'**	87	57	1010111	127	**W**					
40	28	101000	50	**(**	88	58	1011000	130	**X**					
41	29	101001	51	**)**	89	59	1011001	131	**Y**					
42	2A	101010	52	*****	90	5A	1011010	132	**Z**					
43	2B	101011	53	**+**	91	5B	1011011	133	**[**					
44	2C	101100	54	**,**	92	5C	1011100	134	****					
45	2D	101101	55	**-**	93	5D	1011101	135	**]**					
46	2E	101110	56	**.**	94	5E	1011110	136	**^**					
47	2F	101111	57	**/**	95	5F	1011111	137	**_**					

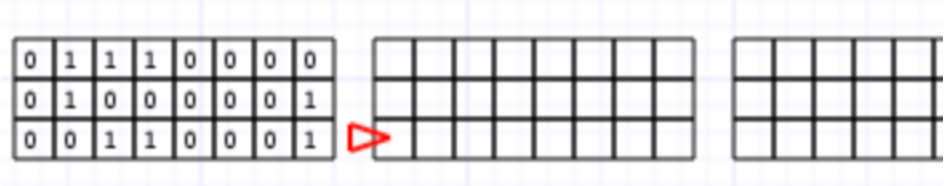
Si le message est « leci » et la clef « abri » alors on écrit le binaire de chacun des mots :

leci	01101100	01100101	01100011	01101001
abri	01100001	01100010	01110010	01101001

On applique ensuite l'opérateur XOR à chacun des encodages binaires des lettres :

leci	01101100	01100101	01100011	01101001
abri	01100001	01100010	01110010	01101001
code	00001101	00000111	00010001	00000000

Pour terminer, votre code devra être capable de dessiner à l'aide de la tortue le tableau ci-dessus pour les trois premières lettres de votre message.



Votre code devra être commenté correctement, et chacune des fonctions devra avoir ses tests unitaires configurés.

3. En résumé

- Coder le code César : crypter et décrypter.
- Coder le code Vigenère : crypter et décrypter.
- Coder le code avec XOR : crypter et décrypter.
- Dessiner les trois premières lettres de votre message et leur correspondance en binaire et avec le code utilisant XOR.

4. Spécifications

4.1 Fonction `cryptCesar(message, clef)`

Retourne le message crypté à l'aide de la clef en partant d'un message en clair. Les espaces et les signes de ponctuation sont préservés (ils ne sont pas cryptés).

4.2 Fonction `decryptCesar(message, clef)`

En partant de la même clef et d'un message crypté, retourne le message en clair.

4.3 Fonction `cryptVigenere(message, clef)`

Retourne le message crypté à l'aide de l'algorithme de Vigenère. La table de correspondance de Vigenère est fournie sous la forme d'un dictionnaire à deux dimensions :

- La première dimension correspond aux lettres du message
- La deuxième dimension correspond aux lettres de la clef

4.4 Fonction `decryptVigenere(message, clef)`

Retourne le message en clair à partir d'un message crypté et de la même clef que celle utilisée pour crypter.

4.5 Fonction `getBinaire(char)`

Retourne le binaire correspondant au caractère char. Le binaire retourné est une chaine de caractères contenant uniquement des 0 et des 1. Chaque caractère doit contenir 8 bits exactement.

Vous pourrez utiliser **`bin(ord(char))`** qui fournit le code binaire correspondant à un caractère.

4.6 Procédure `opXor(char1, char2)`

Prend deux chaînes de caractères binaires correspondant à deux caractères et retourne un texte représentant le résultat de l'opération XOR binaire. Vous devez coder vous-même la comparaison XOR en comparant caractère par caractère.

4.7 Procédure `cryptXor(message, clef)`

Affiche le message crypté à partir d'un message en clair et de la clef. Vous devrez convertir le binaire obtenu avec XOR en caractères ASCII. Vous pourrez utiliser : **`chr(int("00011001"))`** pour obtenir le caractère correspondant à une chaine en binaire.

4.8 Procédure `decryptXor(message, clef)`

Affiche le message décrypté à partir d'un message crypté par Xor et de la clef.

5 Évaluation

Voici les critères d'évaluation du travail :

- L'exactitude (respect de la spécification)
- L'élégance et la lisibilité du code
- La présence de commentaires explicatifs lorsque nécessaire
- Le choix des identificateurs
- La décomposition fonctionnelle et le choix des tests unitaires

Indications :

- Vous devez implémenter votre code dans l'environnement **CodeBoot**.
- La performance de votre code doit être raisonnable.
- Chaque fonction devrait avoir un bref commentaire pour indiquer ce qu'elle fait.
- Il devrait y avoir des lignes vides pour que le code ne soit pas trop dense.
- Les identificateurs doivent être bien choisis pour être compréhensibles (éviter les noms à une lettre à l'exception de i, j, … pour les variables d'itération des boucles).
- Vous devez respecter le standard de code pour ce projet (noms de variables en camelCase).
- Vous pouvez créer des fonctions intermédiaires dont vous pourriez avoir besoin.
- Utilisez les techniques vues en cours.