Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Réseaux & Système

TP n°2

Python

Manipulation binaire

1 – Utilisation d'un GCL, « générateur à congruence linéaire », est un générateur de nombres pseudo-aléatoires basé sur des congruences et une fonction affine :

$$X_{n+1} = (aX_n + c) \bmod m, \text{ où le terme } X_0 \text{ est appelé « } seed \text{ ».}$$

- ▷ *Pour chaque seed, on obtient une nouvelle suite de nombres.*
- ▷ *Les nombres de la suite ont l'apparence de l'aléas.*
- ▷ *Cette suite est plus ou moins grande : tout nouveau nombre étant basé sur le précédent, si un nombre apparait une deuxième fois dans la suite alors la suite se répète entièrement à partir de ce nombre.*
- ▷ *Le nombre de valeurs de la suite étant fini, il dépend de m, la suite se répètera forcément.*
- ▷ *En utilisant, un même seed, on obtient la même séquence de nombres (d'où le nom de «pseudo»-aléatoire).*

Certaines valeurs bien choisies pour m, a et c permettent d'obtenir des séquences assez longues.

On utilisera les valeurs trouvées par Donald Knuth :

m	a	c
2^{64}	6364136223846793005	1442695040888963407

Questions :

a. Quelle est la taille maximale des valeurs données par le générateur avec les paramètres de D. Knuth ?

b. L'opérateur binaire « xor » permet de combiner une séquence binaire S_a avec une séquence binaire S_b en inversant les bits de S_a de même rang que les bits à 1 de S_b, ce qui donne la séquence S_r.
Vérifiez que si on combine avec un xor S_r avec S_b, on obtient bien S_a.

c. Écrire un programme de « chiffrement » permettant de combiner un message M avec une séquence de valeurs obtenues à l'aide du générateur à congruence linéaire pour un seed donné.
Vous vérifierez que l'opération de déchiffrement est possible en utilisant le même seed.

2 – Écrire un programme réalisant l'**encodage base64** d'un fichier conformément à la **RFC 2045**.
Description tirée de Wikipedia, dérivée de la RFC 2045

Le codage base64 est un codage de l'information utilisant 64 caractères, choisis pour être compréhensible sur la majorité des systèmes.

Il est défini en tant que **codage MIME**.

Un alphabet de **65 caractères** est utilisé pour permettre la représentation de 6 bits par caractère.

Le '=' (65e caractère) est utilisé dans le processus de codage pour les caractères finaux.

Le **processus de codage** substitue des groupes de 4 caractères, soient 24 bits, de données en entrée, par une chaîne en sortie de 4 caractères codés de la manière suivante :

- ▷ en procédant de gauche à droite, un groupe de 24 bits est créé en concaténant 3 octets (8 bits par octet) ;
- ▷ ces 24 bits sont traités comme 4 groupes de 6 bits ;
- ▷ chaque groupe de 6 bits, soit une valeur entière comprise de 0 à 63, est converti en un unique caractère dans l'alphabet de la base 64.

Chaque groupe de 6 bits est utilisé comme **index** dans la table des caractères de la base 64.

Le caractère référencé par l'index correspondant est utilisé comme **codage** de ce groupe de 6 bits.

Valeur	Codage	Valeur	Codage	Valeur	Codage	Valeur	Codage
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(complément)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Un **traitement spécial** est effectué si moins de 24 bits sont disponibles à la fin des données à coder.
Aucun bit ne devant être perdu, si **moins de 24 bits** sont disponibles alors des **bits à zéro** sont ajoutés à la droite des données pour former un **nombre entier de groupes de 6 bits**.

Seuls trois cas sont possibles pour la dernière partie des données d'entrée :

- ▷ **24 bits** restants, alors le dernier groupe de caractères de sortie sera de 4 caractères ;
- ▷ 16 bits restants, alors le dernier groupe de caractères de sortie sera composé de trois caractères de codage suivis d'un caractère '=' (exprimant un complément) ;
- ▷ **8 bits** restants, alors le dernier groupe de caractères de sortie sera composé de deux caractères de codage suivis de deux caractères '=' de complément.

Exemple de codage

Prenons le groupe de 3 caractères « Hi! ».

Ci-dessous la première ligne indique en binaire l'équivalence de ces 3 octets :

La transformation consiste à séparer les bits pour former en sortie 4 groupes de 6 bits.

```
01001000 01101001 00100001 <=> 010010 000110 100100 100001
```

Les 4 groupes de 6 bits en sortie nous donnent les valeurs 18, 6, 36 et 33.

En suivant la correspondance de la table indexée, nous obtenons les 4 caractères « SGkh ».