Thème 1 : Types de bases

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

{:.center width=350px}
Table ASCII

!!! exo “Exercice 1 :” Décoder l’expression suivante, écrite en ASCII :

```01001000 01000001 01010011 01010100 01000001 00100000 01001100 01000001 00100000 01010110 0100
```

ISO/CEI 8859-15																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*non utilisé*															
1x																
2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*non utilisé*															
9x																
Ax		¡	¢	£	€	¥	Š	§	š	©	ª	«	¬		®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

{:.center width=300px} *Latin-9*

!!! exo “Exercice 2 :” **Q.1.** Le mot représenté par les octets ci-dessous est-il codé en ASCII ou en Latin 9 ? Donner ce mot :

```
00000000 64 C3 A9 C3 A7 75 65 73 0A
```

Q.2. Représenter `goûté` en Latin-9

Définition du nombre d'octets utilisés dans le codage (uniquement les séquences valides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0xxxxxxx	00 à 7F	1 octet, codant 7 bits
U+0080 à U+07FF	110xxxxx 10xxxxxx	C2 à DF	2 octets, codant 11 bits
U+0800 à U+0FFF	1110**0000** 10**1**xxxxx 10xxxxxx	E0 (le 2e octet est restreint de A0 à BF)	3 octets, codant 16 bits
U+1000 à U+1FFF	1110**0001** 10xxxxxx 10xxxxxx	E1	
U+2000 à U+3FFF	1110**001x** 10xxxxxx 10xxxxxx	E2 à E3	
U+4000 à U+7FFF	1110**01xx** 10xxxxxx 10xxxxxx	E4 à E7	
U+8000 à U+BFFF	1110**10xx** 10xxxxxx 10xxxxxx	E8 à EB	
U+C000 à U+CFFF	1110**1100** 10xxxxxx 10xxxxxx	EC	
U+D000 à U+D7FF	1110**1101** 10**0**xxxxx 10xxxxxx	ED (le 2e octet est restreint de 80 à 9F)	
U+E000 à U+FFFF	1110**111x** 10xxxxxx 10xxxxxx	EE à EF	
U+10000 à U+1FFFF	11110**000** 10**01**xxxx 10xxxxxx 10xxxxxx	F0 (le 2e octet est restreint de 90 à BF)	4 octets, codant 21 bits
U+20000 à U+3FFFF	11110**000** 10**1**xxxxx 10xxxxxx 10xxxxxx		
U+40000 à U+7FFFF	11110**001** 10xxxxxx 10xxxxxx 10xxxxxx	F1	
U+80000 à U+FFFFF	11110**01x** 10xxxxxx 10xxxxxx 10xxxxxx	F2 à F3	
U+100000 à U+10FFFF	11110**100** 10**00**xxxx 10xxxxxx 10xxxxxx	F4 (le 2e octet est restreint de 80 à 8F)	

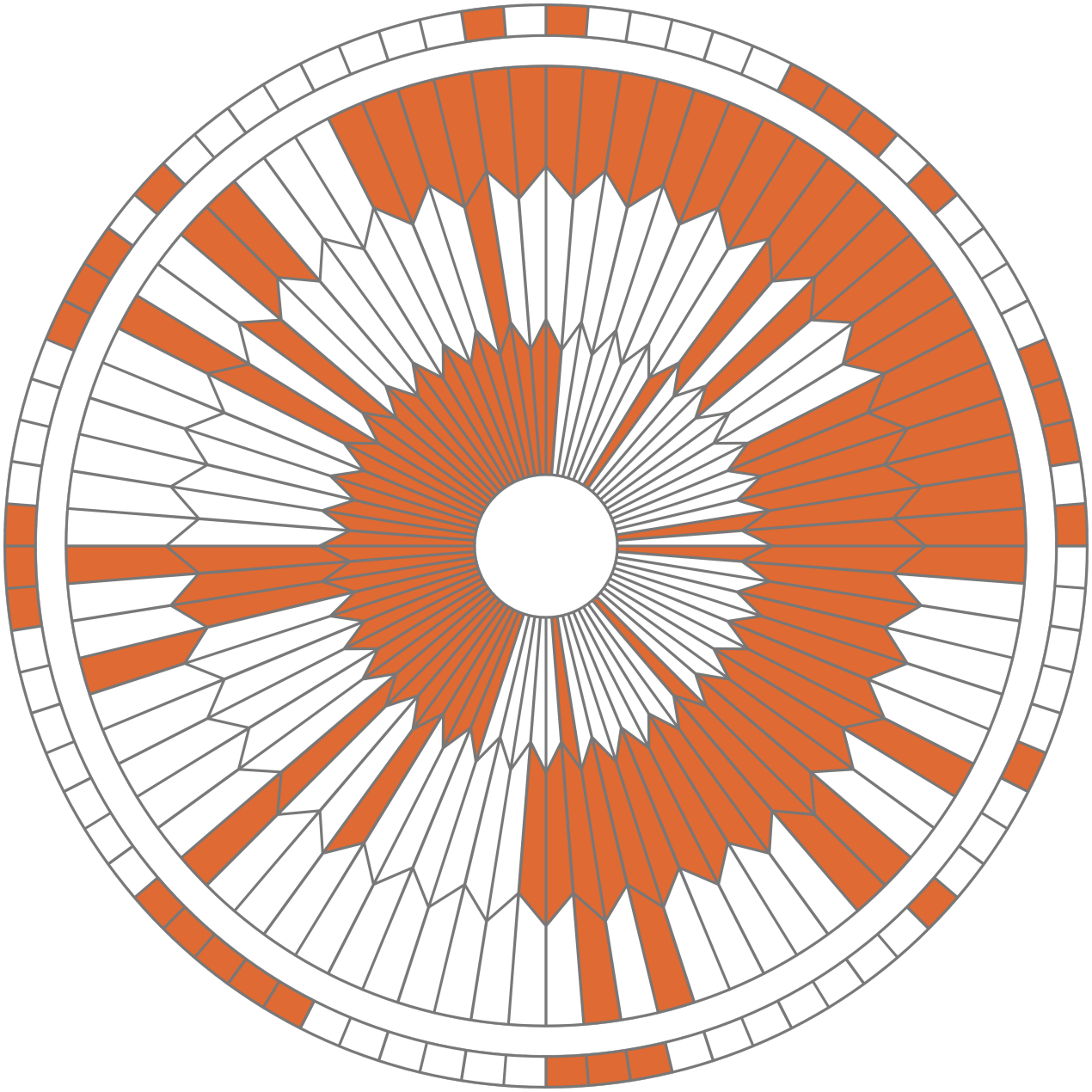
{:.center width=800px}

Latin étendu B

HEX		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
180	384	ƀ	Ɓ	Ƃ	ƃ	Ƅ	ƅ	Ɔ	Ƈ	ƈ	Ɖ	Ɗ	Ƌ	ƌ	ƍ	Ǝ	Ə
190	400	Ɛ	Ƒ	ƒ	Ɠ	ƔƔ	ƕ	Ɩ	Ɨ	Ƙ	ƙ	ƚ	ƛ	Ɯ	Ɲ	ƞ	Ɵ
1A0	416	Ơ	ơ	Ƣ	ƣ	Ƥ	ƥ	Ʀ	Ƨ	ƨ	Ʃ	ƪ	ƫ	Ƭ	ƭ	Ʈ	Ư
1B0	432	ư	Ʊ	Ʋ	Ƴ	ƴ	Ƶ	ƶ	Ʒ	Ƹ	ƹ	ƺ	Ƶ	ƶ	ƹ	ƺ	ƻ
1C0	448	ǀ	ǁ	ǂ	ǃ	DŽ	Dž	dž	LJ	Lj	lj	NJ	Nj	nj	Ǎ	ǎ	Ǐ
1D0	464	ǐ	Ǒ	ǒ	Ǔ	ǔ	Ǖ	ǖ	Ǘ	ǘ	Ǚ	ǚ	Ǜ	ǜ	ǝ	Ǟ	ǟ
1E0	480	Ǡ	ǡ	Ǣ	ǣ	Ǥ	ǥ	Ǧ	ǧ	Ǩ	ǩ	Ǫ	ǫ	Ǭ	ǭ	Ǯ	ǯ
1F0	496	ǰ	DZ	Dz	dz	Ǵ	ǵ	Ƕ	Ƿ	Ǹ	ǹ	Ǻ	ǻ	Ǽ	ǽ	Ǿ	ǿ
200	512	Ȁ	ȁ	Ȃ	ȃ	Ȅ	ȅ	Ȇ	ȇ	Ȉ	ȉ	Ȋ	ȋ	Ȍ	ȍ	Ȏ	ȏ
210	528	Ȑ	ȑ	Ȓ	ȓ	Ȕ	ȕ	Ȗ	ȗ	Ș	ș	Ț	ț	Ȝ	ȝ	Ȟ	ȟ
220	544	Ƞ	ȡ	Ȣ	ȣ	Ȥ	ȥ	Ȧ	ȧ	Ȩ	ȩ	Ȫ	ȫ	Ȭ	ȭ	Ȯ	ȯ
230	560	Ȱ	ȱ	Ȳ	ȳ	ȴ	ȵ	ȶ	ȷ	ȸ	ȹ	Ⱥ	Ȼ	ȼ	Ƚ	Ⱦ	ȿ
240	576	ɀ	Ɂ	ɂ	Ƀ	Ʉ	Ʌ	Ɇ	ɇ	Ɉ	ɉ	Ɋ	ɋ	Ɍ	ɍ	Ɏ	ɏ

!!! exo "Exercice 3 :"
Donnée le codage Unicode la lettre Ħ puis son codage en UTF-8

!!! exo "Exercice 4" Décoder le message suivant :



{:.center width=600px}

!!! exo “Exercice 5” Le défi du cours : codage UTF-8 (Latin-9), décoder le texte ci-dessous :

`56 65 72 73 20 6C 27 69 6E 66 69 6E 69 20 65 74 20 6C 27 61 75 2D 64 65 6C C3 A0``

!!! exo “Exercice 4 :” Codage XOR :

Q.1. Le nombre 65, donné ici en écriture décimale, s’écrit 01000001 en notation binaire. En détaillant la méthode utilisée, donner l’écriture binaire du nombre 97.

Q.2. La fonction logique **OU EXCLUSIF**, appelée **XOR** et représentée par le symbole , fournit une sortie égale à 1 si l’une ou l’autre des deux entrées vaut 1 mais pas les deux.

On donne ci-dessous la table de vérité de la fonction XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Poser et calculer l'opération : 11011101 01101011

On donne, ci-dessous, un extrait de la table ASCII qui permet d'encoder les caractères de A à Z.
On peut alors considérer l'opération XOR entre deux caractères en effectuant le XOR entre les codes ASCII des deux caractères.
Par exemple : 'F' XOR 'S' sera le résultat de 01000110 01010011.

Code ASCII Décimal	Code ASCII Binaire	Caractère
65	01000001	A
66	01000010	B
67	01000011	C
68	01000100	D
69	01000101	E
70	01000110	F
71	01000111	G
72	01001000	H
73	01001001	I
74	01001010	J
75	01001011	K
76	01001100	L
77	01001101	M

Code ASCII Décimal	Code ASCII Binaire	Caractère
78	01001110	N
79	01001111	O
80	01010000	P
81	01010001	Q
82	01010010	R
83	01010011	S
84	01010100	T
85	01010101	U
86	01010110	V
87	01010111	W
88	01011000	X
89	01011001	Y
90	01011010	Z

{:.center width=350px}

On souhaite mettre au point une méthode de cryptage à l'aide de la fonction XOR.
Pour cela, on dispose d'un message à crypter et d'une clé de cryptage de même longueur que ce message. Le message et la clé sont composés uniquement des caractères du tableau ci-dessus et on applique la fonction XOR caractère par caractère entre les lettres du message à crypter et les lettres de la clé de cryptage.

Question 3. Chiffrer **INFORMATIQUE** avec la clé **NSI**. Pour cela recopier et compléter le tableau ci-dessous :

LETTRE	I	N	F	O	R	M	A	T	I	Q	U	E
ASCII												
BINAIRE												
CLE	N	S	I	N	S	I	N					
ASCII												
BINAIRE												
XOR												
ASCII												

Q.4.
Recopier et compléter la table de vérité de ( ) .

E_1	E_2	$E_1 \oplus E_2$	$(E_1 \oplus E_2) \oplus E_2$
0	0	0	
0	1	1	
1	0	1	
1	1	0	

{:.center width=300px}

A l'aide de ce résultat, proposer une démarche pour décrypter un message crypté.

Q.5 Décoder le message suivant : `12 1 8 24 28 105 15 115 29 1 6 26`