Démonstration #2

Le but de la première démonstration est de familiariser les étudiants au système informatique ainsi qu'expliquer les exercices.

Système informatique.

Montrer comment se brancher au compte informatique et démarrer un fureteur (ceci étant nécessaire pour accéder à codeBoot et à Studium pour les exercices notés).

Montrer comment se brancher au système Studium.

Montrer comment démarrer le système codeBoot en suivant le lien pour les exercices notés sur Studium, puis faire l'évaluation d'une expression, par exemple 2*3+4*5; faire l'évaluation d'une expression pas-à-pas, pour établir le nombre de pas requis pour évaluer une expression

Exercices notés.

- Les étudiants doivent faire presque chaque semaine des exercices notés.
- En démonstration, des exercices non notés sont expliqués et résolus par le démonstrateur. S'il reste du temps avant la fin de la démonstration, les étudiants peuvent commencer à résoudre les exercices notés.

Exercices non notés

1. **Compter jusqu'à 20 en base 10, 8, 16 et 2**

(10)	(8)	(16)	(2)
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111
16	20	10	10000

(10)	(8)	(16)	(2)
17	21	11	10001
18	22	12	10010
19	23	13	10011
20	24	14	10100

2. Conversions de bases:

- Encoder 134_{10} en base 7, en binaire, en octal et en hexadécimal;

Base 7 :

134/7 = 19 et rest 1

19/7 = 2 et rest 5

2/7 = 0 et rest 2

$134_{10} = 251_7$

Base 2 : $134_{10} = 128(2^7)+4(2^2)+2(2^1)=10000110_2$

Base 8 : $134_{10} = 10\ 000\ 110_2 = 010\ 000\ 110_2 = 206_8$

Base 16 : $134_{10} = 1000\ 0110_2 = 1000\ 0110_2 = 86_{16}$

- Représenter 65_{10} et -28_{10} en complément à 2 sur 8 bits; Montrer le calcul en utilisant deux méthodes vues en classe.

  Méthode 1 et méthode 2 pour 65_{10}.

  $65_{10} = 64(2^6) + 1(2^0) = 1000001_2$

  Représentation c-à-2 sur 8 bits : 01000001

  -28_{10}

  $28_{10} = 16(2^4) + 8(2^3) + 4(2^2) = 11100_2$

  28_{10} sur 8 bits : 00011100

Méthode 1 :

 100000000
\- 00011100

11100100

Méthode 2 : 00011100 –> c-à-1 –> 11100011-> +1 -> **11100100**

Représentation c-à-2 sur 8 bits : 11100100

- Supposons que les valeurs suivantes sont encodées en complément à 2 sur 5 bits :
  - 10101
  - 01100

  Quelles valeurs en décimal sont encodées (3 méthodes)?

10101 – nombre négatif

Méthode 1 :

100000
-10101

01011_2 => $-1011_2 = -(2^3+2^1+2^0) = -11_{10}$

Méthode 2 : 10101 –> c-à-1 -> 01010 -> +1 -> 1011 => 10101 = $-1011_2 = -11_{10}$

Méthode 3 :

4 3 2 1 0

1 0 1 0 1 –> $-2^4+2^2+2^0 = -16+4+1 = -11_{10}$

- En base 16, "64" encode quelle valeur?

  $64_{16} = 6*16^1+4*16^0 = 0110\ 0100_2 = 2^6+2^5+2^2 = 100_{10}$

- En base 8, "100" encode quelle valeur?

  $100_8 = 1*8^2 = 64_{10}$

  $100_8 = 001\ 000\ 000_2 = 1000000_2 = 2^6 = 64_{10}$

2. Soit l'expression `4*+6/(3+-5)` . Dans quel ordre les opérateurs sont-ils exécutés?

Solution :

Le + unaire devant le 6 est exécuté avant la multiplication de 4 et 6, le contenu de la parenthèse est exécuté avant de réaliser la division. Le - unaire est exécuté avant l'addition de 3 et "-5".

+*-+/

3. Trouvez la plus petite expression JavaScript (ayant le minimum de caractères incluant les parenthèses et symboles, et pas de blancs) contenant les nombres 2, 3, 5, et 10 (exactement une fois chaque), et les opérateurs +, - et * (autant de fois que vous voulez), dont la valeur est 85.

Solution :

On peut procéder à l'envers :

On a que 85/5 = 17 et que 2*10-3 = 17

Donc, (2*10-3)*5 = 85