Faculté des Sciences & Techniques — Université de Limoges

Master 1st year

Réseaux & Système

Examination – January 2020

Duration : 2h — All documents authorized

Python programming — (8 points)

1 – 8pts « *Hamming distance* » is a **metric** for comparing two binary data strings. While comparing two binary strings of equal length, Hamming distance is the **number of bit positions** in which the two bits are **different**.

The Hamming distance between two strings, a and b is denoted as $d(a, b)$.

In order to calculate $d(a, b)$ between two strings, and , we perform their XOR operation, $a \oplus b$, and then count the total number of 1s in the resultant string.

Example : Suppose there are two strings `1101 1001` and `1001 1101` :
▷ $11011001 \oplus 10011101 = 01000100$, where $\oplus$ is the « *xor* » ;
▷ the result is `0100 0100` containing two 1s, the Hamming distance is 2.
$\Longrightarrow d(11011001, 10011101) = 2$

In a set of strings of **equal lengths**, the **minimum Hamming distance** is the **smallest** Hamming distance between all possible pairs of strings in that set.

a. Let the set of binary strings be : $1110010, 1001001, 1001111, 0101100$ *(1pt)*
Compute the minimum Hamming distance for this set.

b. In Python, the « *xor* » operation is denoted ^ and is used according to : *(1pt)*

```
xterm
>>> 12^3
15
```

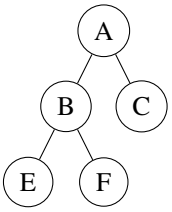
Explain how, in Python, we could compute a « *xor* » between two binary strings.

c. Write a Python function that takes two binary strings as arguments and return the Hamming distance between these two string. *(2pts)*

d. Write a Python program that uses the previous function to compute the « *minimum Hamming distance* » for a set of binary strings given as a list. *(2pts)*

e. Write a Python program that, giving a random binary string, return the nearest binary string from a set S of known binary strings (the random binary string doesn't belong necessary to the set S). *(2pts)*

2 – 3pts We represent a **binary tree** as a list of embedded lists :
`node = [ head, child1, child2]` (an empty node is given by an empty list).




```
1  arbre = ['A',
2           ['B',
3              ['E', [], []],
4              ['F', [], []]
5           ], # fin de B
6           ['C',
7              [],
8              []
9           ] # fin de C
10       ]
```

Write a Python program that display each node of the tree.

Unix — (2 points)

3 – 2pts a. For a 64bits computer under Linux, the size of page is 4096 bytes : how much pages do we get ? *(1pt)*

b. Does it matter if the starting address of a program in the central memory is shifted from one byte before its execution by the processor ? *(1pt)*
Explain what happends.

▬▬▬ Networking — (5 points)

4 – 5pts

a. We want to design a **TCP server** that receives a value denoted in hexadecimal and return a value expressed as an integer. *(1pt)*
What information must be **shared** with the client using this server ?

b. Write a Python program acting as a TCP server that **for each received connection** performs the conversion from hexadecimal to integer (only one conversion for one connection). *(3pts)*

c. If we want to extend the capabilities of the server to perform the **reverse operation** « *integer*$\Longrightarrow$ *hexadecimal* », how to proceed ? *(1pt)*