Algorithms and Computational Thinking

TP-3: Boolean algebra and conditional branching

Objective: The objective of this TP is to get comfortable with Boolean logic, and conversion from one base to another. The exercises also aim to strengthen you understanding of different data types and conversion from one data type to another. Finally, you will get a basic understanding of testing for certain conditions in the program and taking decisions based on those conditions.

Exercise-I: Boolean logic

a. Use Boolean logic to complete the following truth tables:

AND	T	F
T		
F		

OR	T	F
T		
F		

NOT	
T	
F	

b. Using the above truth tables to perform the following **bitwise operations**.

A **bitwise operation** operates on one or more bit patterns or binary numerals at the level of their individual bits. It is a quick and simple operation supported by the processor and used to manipulate values for comparisons and calculations. **A bit's position** is always counted from the **right** (least significant) side, advancing left (most significant). For example, the binary value 0001 (decimal 1) has zeroes at every position but the first one.

Bitwise Not is an operation that calculates the one's complement of the given binary value. Bits that are 0 become 1, and those that are 1 become 0. For example:

```
NOT 0111 (decimal 7)
  = 1000 (decimal 8)
```

Similarly, A **bitwise AND** takes two equal-length binary representations and performs the logical AND operation on each pair of the corresponding bits, by multiplying them.

```
   0101 (decimal 5)
AND 0011 (decimal 3)
  = 0001 (decimal 1)
```

Bitwise operations are performed on binary numbers of equal length. If a number represented in binary does not fill up all the allocated bits, zero's should be padded to

the left. For example, a decimal number 5 is represented in 8bit binary format as 00000101.

A simple analogy of such operations are set theory functions such as the Union (logical OR), Intersection (logical AND) and Negation (logical NOT).

Using the above information, perform the following bitwise operations.

- **NOT** T F T T F T F F
- **NOT** F F T F F T
- **NOT** 1 0 0 1 0 0 1 1
- F F T F T T F T **AND**
  F T F F T F T T
- T T F F T F F T T F **AND**
  T F F T T F F T F F
- 0 1 0 0 0 1 1 0 1 1 **AND**
  1 1 0 1 0 0 1 0 1 0
- F T F T T F T T F F **OR**
  F T F F T F T T F T
- T F T F F F T F T T **OR**
  F F T T F T F T F F
- 0 0 1 0 1 1 1 0 1 0 **OR**
  0 0 1 1 0 1 1 0 0 1

c. Represent the following signed integers (base 10) in their equivalent base 2 (8 bit) representations.

98, -65, 29, 36, -50, -2, 99, -22

d. Verify the results of the above question by writing a program in Scala and Python.

Hint: In Scala use the Integer.toBinaryString() function and in python use the bin () function.

Exercise–II: Conditional Branching

Objective: Understanding how to check for certain conditions during program execution and performing operations based on the conditions.

Problem Statement: The “Nuit du Crime” is taking place at the D! Club in Lausanne! However, the bouncer has some problems calculating the age of the students that want to enter the club. Write a program that checks if a student can enter or not (minimum age 18 years) given his/her birth date. Check the “type” of the user input and perform the required type conversions.

Hint: Use the libraries used in the last session to manipulate dates. Consider all years

as 365 days for simplicity.

Sample Output: **(You are free to choose the format of the date entered by the user)**

```
Enter the student birthday:
Year:
>> 1997
Month:
>> 09
Day:
>> 10
Old enough to enter!

Enter the student birthday:
Year:
>> 2002
Month:
>> 05
Day:
>> 03
Too young!
```