UNIVERSIDAD CARLOS III DE MADRID	**UNIVERSIDAD CARLOS III DE MADRID** **FORMAL LANGUAGES AND AUTOMATA THEORY.** **COMPUTER SCIENCE DEGREE. EXTRAORDINARY EXAM -26.06.2013** **Second name(s):**____________________ **First name:**____________________ **NIA:**__________ **Signature:**__________

PROBLEM 1: Maximum mark 1.25 points.

Design a Turing Machine to calculate the predecessor of a Roman number which only consists of the symbols I and V. The tape initially contains just only one Roman number. At the end of the process, the tape must contain just the predecessor of this number, that is to say, the original number in the tape is not preserved. In addition, it is mandatory to leave the input header in the first symbol of the Roman number when finishing the process

Examples:

- Initial number: ***III*** Result provided by the Turing Machine: ***II***; and
- Initial number: ***V*** Result provided by the Turing Machine: ***IV***.

It is required:

a) Formal definition with the seven elements of the Turing Machine. Include the transition diagram (not the list, nor the table of the transition function).
b) Detailed description of the algorithm implemented by the Turing Machine.
c) Explain the meaning of:
    - each symbol of the alphabet of the tape not defined in this wording,
    - each one of the states and transitions, or group of states and transitions.

PROBLEM 2: Maximum mark 1.25 points.

A transmitter ***A*** sends a message ***m*** to a receiver ***B***. It is known that the message is a sequence of characters taken from an alphabet {a, b}. The sequence *abbaa* has been received, but we know for certain that one additional character is missing due to the noise in the channel. Fortunately, we know the characteristic equations of the finite automaton that recognizes the transmitted words:

$X_0=aX_1+bX_1+\lambda$

$X_1= aX_1+bX_0+b$

Indicate the complete message originally transmitted and explain why.

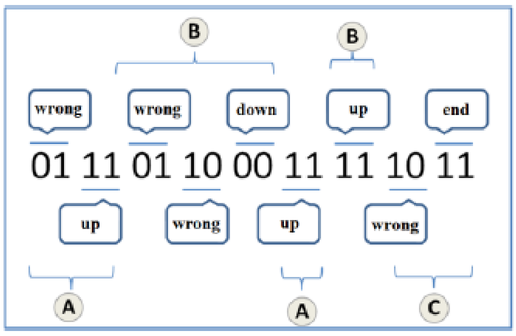
PROBLEM 3: Maximum mark 1.25 points.

We have designed an e-learning application that poses several questions to the user, which are organized in turns of 2 questions. Each user answer is evaluated as either *right*or *wrong*. There are 3 levels of difficulty: *A* (easy), *B* (intermediate), and *C* (difficult). The program always begins a turn of 2 questions of level *A*. When the user answers the two questions of the current turn correctly, then the system moves to the next level of difficulty. If he fails the two questions, then if the current level is A, the system continues in the level, and if the current level is B or C, the system downgrades to the inferior level. For example, if the user is in a turn of intermediate questions (level B), and fails the two responses, then the next turn will be two questions of level A. When the user answers correctly the two questions of a turn of level C, the program ends unless he decides to begin again at level A.

The output of the application is a sequence of zeros and ones that represent respectively the incorrect and correct answers provided by the user to the different questions.

It is required to design a grammar that acepts the language whose words are the outputs of the program and**, using this grammar**, **obtain formally the automaton** that verifies if the outputs are correct. The different values of the transition function must be provided.

Example of correct output:



wrong: it indicates that one of the answers in the turn is wrong
up: it indicates that both answers are right and it upgrades to the next level
down: it indicates that both answers are wrong and it downgrades to the previous level
finish: it indicates that in the C level the user has correctly answered both questions