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***.

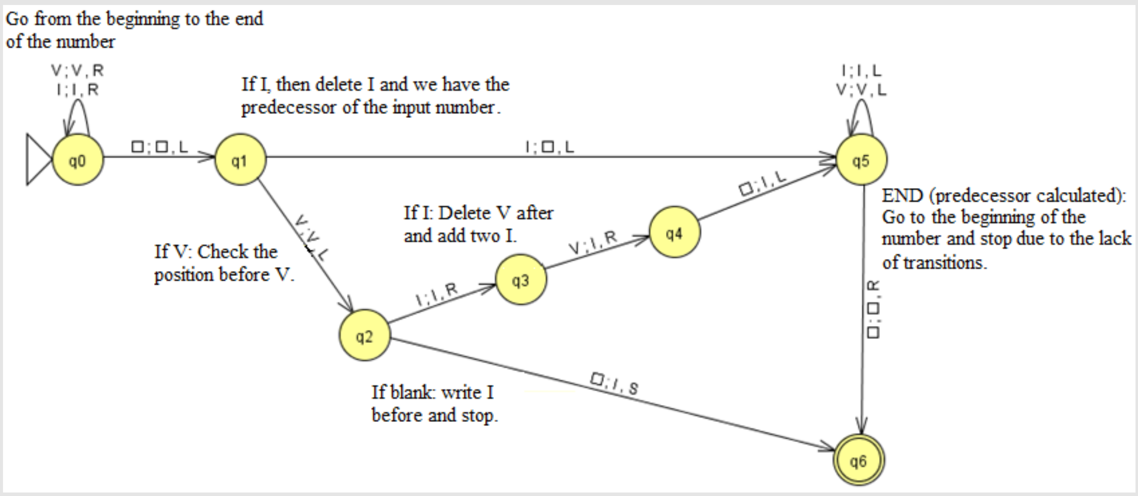
It is required:

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.

a) Formal definition:

TM=({I,V}, {I,V,□}, □, {q0,q1,q2,q3,q4,q5,q6}, q0, f, {q6}),

Where the transition function f is given by the following graph:


Go from the beginning to the end of the number
V;V,R
I;I,R
□;□,L
If I, then delete I and we have the predecessor of the input number.
I;□,L
I;I,L
V;V,L
If V: Check the position before V.
V;V,L
If I: Delete V after and add two I.
I;I,R
V;I,R
□;I,L
END (predecessor calculated): Go to the beginning of the number and stop due to the lack of transitions.
□;□,R
If blank: write I before and stop.
□;I,S
q0
q1
q2
q3
q4
q5
q6

b) Detailed description of the algorithm

General idea: Go from the beginning to the end of the number, verifying if it is only required to remove the last I (cases: II, III, VI, VII, VIII), or an additional operation is required when the number ends with a V (IV, V). The eight possible scenarios are:

Input	Result
I	☐ (blank)
II	I
III	II
IV	III
V	IV
VI	V
VII	VI
VIII	VII

Detailed algorithm:

1. Go from the beginning to the end of the number. Place the input header on the last symbol.
2. If the number ends with an I, remove the last I and we have obtained the predecessor. Go to step 4.
3. If the number ends with a V, check the position before the V.
    a. If it is an I, remove the V and add two Is (the result is the predecessor). Go to step 4.
    b. If it is a blank (i.e., the input is V), write an I in the current position and stop.
4. The predecessor has been calculated and we only have to leave the input header in the first symbol of the result.

c) Detailed explanation:

- **symbols:**

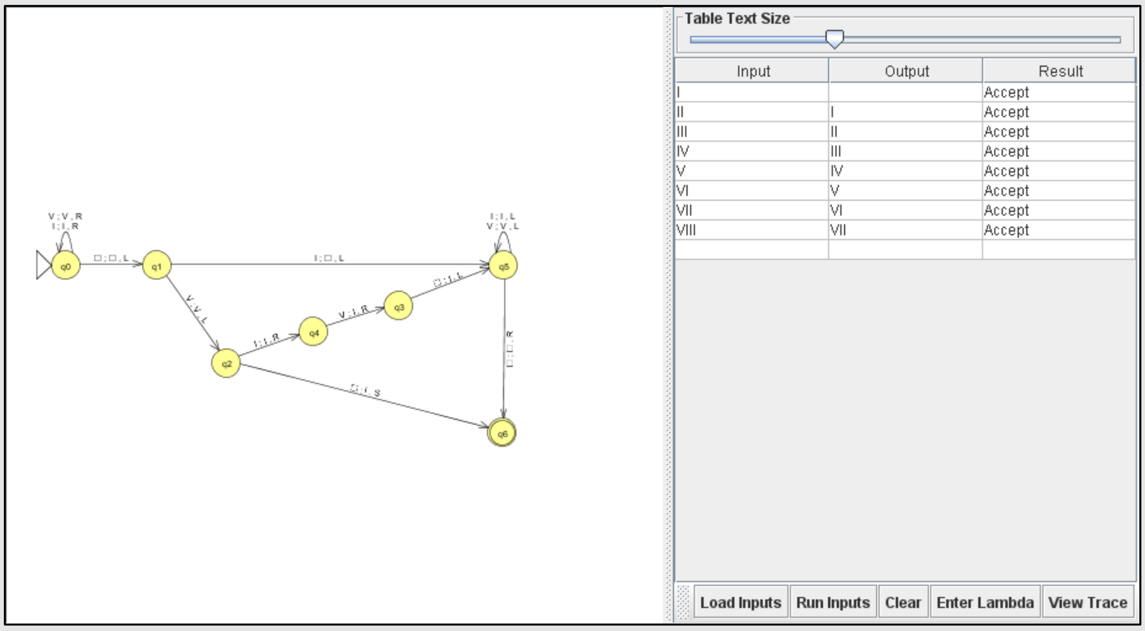
  ☐: blank symbol representing the empty cell. It is also used to represent the number 0.

  No additional symbols have been incorporated.

- **set of states and transitions:**

- **cycle q1 and transition a q2** [step 1 of the algorithm]: go to the end of the number.
- **transition q1 to q5** [step 2 of the algorithm]: the initial number ends with an I. Delete the I and we have the predecessor.
- **transitions q2-q3-q4-q5** [step 3.a of the algorithm]: case IV → III
- **transition q2-q6** [step 3.b of the algorithm]: case V → IV
- **cycle q5 and transition to q6** [step 4 of the algorithm]: go to the beginning of the result to place the input header there and stop (q6).

The following figure shows the result of the JFLAP analysis of the 8 different input words:



Input	Output	Result
I		Accept
II	I	Accept
III	II	Accept
IV	III	Accept
V	IV	Accept
VI	V	Accept
VII	VI	Accept
VIII	VII	Accept

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:

$X_0=aX_1+bX_1+\lambda$

$X_1= aX_1+bX_0+b$

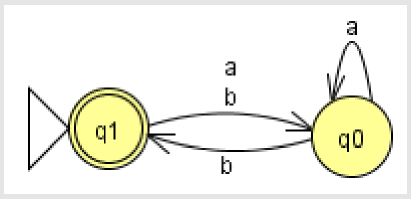
Indicate the complete message originally transmitted and explain why.

There are two main alternatives to solve the problem:

Alternative 1. Solve the system of equations to obtain the RE [(a+b)a*b]*:

$X_1 = aX_1+(bX_0+b) = a^*(bX_0+b)$
$X_0 = a(a^*(bX_0+b))+ b(a^*(bX_0+b))+ \lambda$
$X_0 = (aa^*b+ba^*b)X_0 + aa^*b + ba^*b + \lambda$
$X_0 = (aa^*b+ba^*b)^*. (aa^*b + ba^*b) + \lambda = (aa^*b + ba^*b)^+ = \boxed{(aa^*b + ba^*b)^* = ((a+b)a^*b)^*}$

Alternative 2. From the equations obtain the DFA



Using both alternatives, we can see that the recognized words must end with a *b*. This way, for the input word given by the exercise, the only possibility is that the last character is the one lost during the transmission. This character is a *b*.

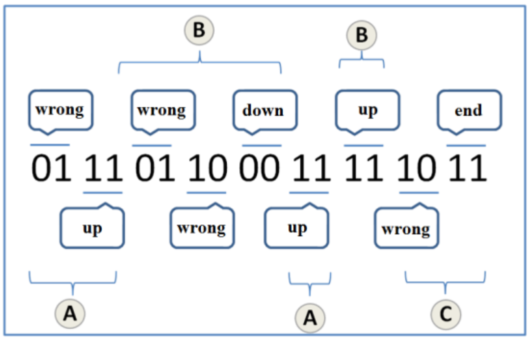
PROBLEM 3: Maximum mark 1.25 points.

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.

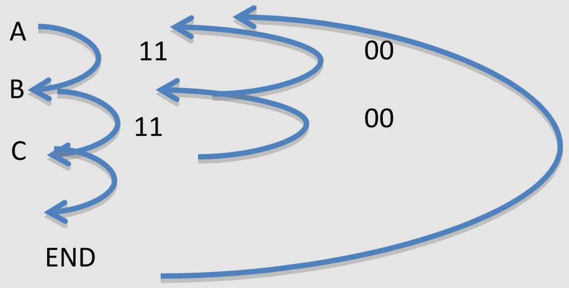
It is required to design a grammar that acepts the language whose words are the outputs of the program and**, using this grammar**, **obtain formally the automaton** that verifies if the outputs are correct. The different values of the transition function must be provided.

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

2 questions: right 1, wrong 0.
3 levels:



We can define a Type-2 grammar 2,

G= ({0,1}, {A,B,C}, A, P)
P = {A::= 00A / 01A / 10A / 11B
B::= 11C / 00A / 10B / 01B
C::= 11/ 11A / 00B / 01C / 10C
}

The corresponding well-formed grammar is:

G= ({0,1}, {A,B,CI,J}, A, P)
P = {A::= 0IA / 0JA / 1IA / 1JB
B::= 1JC / 0IA / 1IB / 0JB
C::= 1J/ 1JA / 0IB / 0JC / 1IC
I::= 0
J::=1
}

Using the corresponding algorithm we can obtain an equivalent PDA_E:

PDA_E = ({0,1}, {A,B,C,I,J}, A, {p}, Φ, f)

f(p, 0,A) = (p, IA)
f(p, 0,A) = (p, JA)
f(p, 1,A) = (p, IA)
f(p, 1,A) = (p, JB)
f(p, 0,B) = (p, JB)
f(p, 0,B) = (p, IA)
f(p, 1,B) = (p, IB)
f(p, 1,B) = (p, JC)

f(p,0,C) = (p, JC)
f(p,0,C) = (p, IB)
f(p,1,C) = (p, J)
f(p,1,C) = (p, JA)
f(p,1,c) = (p, IC)
f(p,0,I) = (p, λ)
f(p,1,J) = (p, λ)