	UNIVERSIDAD CARLOS III DE MADRID
	FORMAL LANGUAGES AND AUTOMATA THEORY
	COMPUTER SCIENCE DEGREE. CONTINUOUS ASSESSMENT - PARTIAL 2
	Date: 2011 November 11th
	Last name(s): ___________________________
	First name: ___________________________
	NIA: ___________________________

Duration: 45 minutes

1. Indicate whether the following statements are true or false.

Correct Answer:+0.3 Wrong Answer:-0.3 No answer =0

Minimum Mark: 0 **Maximum Mark: 3 points**

	True	False
In a grammar, the word λ can be included in the set of terminal symbols.		X
A sentential form can include nonterminal symbols.	X	
There is an algorithm to determine whether a grammar is ambiguous or not.		X
A::=aB is a rule in CNF.		X
For every Type-1 grammar it is possible to find an equivalent Push-Down Automaton.		X
Type-2 languages are also called context-free languages.	X	
Every type-2 grammar is also a type-3 grammar.		X
If the number of sentences of a Language is infinite, then the grammar is recursive.	X	
Every grammar is GNF is a Type-3 grammar.		X
S→AbbB→BaA is a derivation of order 3.		X

2. Indicate whether the following statements are true or false.

Correct Answer:+0.3 Wrong Answer:-0.3 No answer =0
Minimum Mark: 0 **Maximum Mark: 3 points**

	True	False
Given a grammar (Σ={0,1}) with the following production rules { S ::= A 1 |0 B | 0, A ::= A 1 | 1, B ::= λ | 0 S}, it is a Type-3 grammar.		X
The grammar with axiom S and production rules P={S::=B0; B::=1 | C1; C::=B2; D::=S1} is a recursive grammar.	X	
Given a grammar (G = {{0,1},{S,A,B},S, P}) with the following production rules P = { S ::= 1 S | 1 A | 0 B | 0 | λ, A ::= 1 A | 1, B ::= λ | 0 B}, it can be transformed into a Type-3 left-linear grammar.	X	
A::=A is a unit production or redenomination rule		X
The production rule AaC ::= AbabC, does not retain the context.	X	
Given the grammar, G=({a,b} , {A,B,C, S}, S, P={S::= aA; A::= aA | bB; B::= bB | b;},), aabb is a sentence of the language.	X	
CcdEA::=CCEb is a valid rule for a Type-1 grammar.		X
Every Type-3 grammar has an equivalent NON-Deterministic FA.	X	
Every Type-3 grammar has an equivalent Deterministic FA.	X	
Given a grammar (G = {{0,1},{S,A,B},S, P}) with the following production rules P = { S ::= 1 S | 1 A | 0 B | 0 | λ, A ::= 1 A | 1, B ::= λ | 0 B}, if the grammar is cleaned and becomes well-formed, then it would be a type-3 right-linear grammar.	X	

3. (4 points) Given the grammar G = ({0, 1, 2} , {S, A, B, C, D, E}, S, P)

P = {(S ::= AABC), (A::= λ | 1A0), (B::=1B | 1), (C::= 1C1 | 0C0 | λ), (D ::= 0)}

a) Which is the type of grammar in the Chomsky Hierarchy?

It is a Type-2 Grammar given that there is a single nonterminal symbol on the left side of every production rule and the structure of the productions rules does not fulfill the requirements to have a type-3 right-linear grammar or a left-linear grammar.

b) Obtain an equivalent grammar in GNF. Explain in detail the process that you have followed.

There is not left-recursion.

The first problem that is detected is related to useless/unreachable nonterminal and terminal symbols. This way, nonterminals D and E and the terminal 2 can be removed. The useless rule D ::= 0 can then also be removed.

There are two not-generating rules:

A::= λ can be removed by adding:

S ::= ABC | BC
A ::= 10

C::= λ can be removed by adding:

S ::= AAB | AB | B
C::= 11 | 00

Now we have introduced the unit production S ::= B. This production rule can be removed by adding the following production rules:

S ::= 1B | 1

The equivalent well formed grammar includes the following production rules:

S ::= AABC | ABC | BC | AAB | AB | 1B | 1
A::= 1A0 | 10
B::=1B | 1
C::= 1C1 | 0C0 | 11 | 00

The productions not in GNF are:

S ::= AABC S ::= 1A0ABC | 10ABC We add: Z ::= 0 Then: S ::= 1AZABC | 1ZABC	**S ::= ABC** S ::= 1A0BC | 10BC Then: S ::= 1AZBC | 1ZBC	**S ::= BC** S ::= 1BC | 1C
S ::= AAB S ::= 1A0AB | 10AB Then: S ::= 1AZAB | 1ZAB	**S ::= AB** S ::= 1A0B | 10B Then: S ::= 1AZB | 1ZB	

A::= <u>1A0</u> A::= 1AZ	A::= <u>10</u> A::= 1Z

C::= <u>1C1</u> We add: U ::= 1 Then: C::= 1CU	**C::= <u>0C0</u>** C::= 0CZ
C::= <u>11</u> C::= 1U	**C::= <u>00</u>** C::= 0Z

The equivalent grammar in GNF is;

G = ({0, 1} , {S, A, B, C, Z, U}, S, P)
P = {

S ::= 1AZABC | 1ZABC | 1AZBC | 1ZBC | 1BC | 1C | 1AZAB | 1ZAB | 1AZB | 1ZB | 1B | 1
B::=1B | 1
C::= 1CU | 0CZ | 1U | 0Z
Z ::= 0
U ::= 1
}