Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): Dec 2011
Brief notes on answers:

PART A

1. d. In the specified language, every 1 must be followed by 2.
2. a. This language allows *a* or *acc*, which the others don't.
3. b. The language accepted is precisely $\{a^n \mid n \geq 0\}$.
4. b. 00 is impossible.
5. d.
6. bS
7. b.
8. e. Note that both C and A can yield ϵ.
9. e. (Consider e.g. well-bracketed sequences involving two kinds of brackets). c is false but may seem plausible, because every regular *language* is LL(1).
10. c.
11. e. In fact, B is more abstract than A. (This exact scenario will be explicitly covered in lectures.)
12. b. Every r.e. set, including the halting set, can be generated by some grammar.
13. e. The regular expression matches the email address.
14. b. It appends 'not' at the end of the list, which becomes ['students', 'love', 'inf2a', 'not'], then inserts 'the' at index 0, the list becomes ['the', 'students', 'love', 'inf2a', 'not'], and then prints the list and the number 2
15. b.
16. a.
17. c. Zipf's law does not say anything specific about open and closed class words.
18. a. (clocks tick). The probabilities for a–e are respectively 0.392, 0.168, 0.072, 0.009, 0.021.
19. e. This one means *Everybody is hated by somebody.* The others all reduce to a.
20. d. Only G2 is in CNF.

PART B

1. (a) Easy bookwork. The four levels are
   - regular languages (corresponding to DFAs or NFAs)
   - context-free languages (corresponding to nondeterministic pushdown automata)
   - context-sensitive languages (corresponding to linear bounded automata)
   - unrestricted or r.e. languages (corresponding to Turing machines)

   [1 mark for each language family, 1 mark for each machine class.]

   (b) Slightly harder bookwork.
   - For regular languages, restrict to productions of the form $X \to \epsilon$, $X \to Y$, $X \to aY$, where X, Y are nonterminals and a is terminal.
   - For context-free languages, restrict to productions $X \to \beta$.
   - For context-sensitive languages, restrict to productions $\alpha \to \beta$ where $|\alpha| \leq |\beta|$.
   - For unrestricted languages, *any* productions $\alpha \to \beta$ are permitted.

   [2 marks for regular grammars, 1 mark for each of the others.]

   (c) (i). L_1 is context-free. A suitable grammar (with start symbol S) is

$$S \ \to \ T : T \qquad T \ \to \ \epsilon \ \mid \ (T)T$$

   [2 marks. They have seen several CFGs for well-matched bracket sequences in the lectures.]

   (ii). L_2 is context-sensitive [1 mark; this is the standard example given in the lectures.]

   (iii). L_3 is regular. A suitable regular grammar (with start symbol S) is

$$S \ \to \ A \ \mid \ B \qquad A \ \to \ \epsilon \ \mid \ aB \qquad B \ \to \ \epsilon \ \mid \ bA$$

   [2 marks. They have drawn a DFA for this language as a tutorial exercise but not written it as a regular grammar.]

   (iv). L_4 is context-free. A suitable grammar (with start symbol S) is

$$S \ \to \ +T \ \mid \ \bullet S \bullet \qquad T \ \to \ = \ \mid \ \bullet T \bullet$$

   [Slightly harder. 3 marks.]

   (d) A standard pumping lemma proof. Suppose L_4 were accepted by a DFA with k states, and consider the strings $x = \epsilon$, $y = \bullet^k$, $z = + = \bullet^k$; then $xyz \in L_4$. But now given any splitting of y as uvw with $v \neq \epsilon$, set $i = 0$; then $uv^iw = uw = \bullet^l$ for some $l < k$, so $xuv^iwz = \bullet^l + = \bullet^k \notin L_4$. Hence by the pumping lemma L_4 is not regular.

   [1 mark for mentioning the pumping lemma; 3 marks for the details.]

2. (a) The algorithm sets up a probability matrix, with one column for each observation t and one row for each state q. Each column thus has a cell for each state q_i. The algorithm first creates N state columns. The first column corresponds to

the observation for the first word i, the second to the second word, the third to the third word and so on. We begin in the first column by setting the Viterbi value i for each cell to the product of the transition probability (into it from the start state) ad the observation probability (of the first word), The we move on, column by column; for every state in column 1, we compute the probability of moving into each state in column 2, and so on. For each state q_j at time t we compute the value $Viterbi[s,t]$ by taking the maximum over the extensions of all the paths that lead to the current cell.

Marking guide: 2 marks for describing correctly how the matrix is created, and 2 marks for describing how the matrix cells are filled.

(b) The equation is:

$$v_t(j) = \text{argmax limits}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

Marking guide: full credit will be given to the correct equation, partial credit (i.e., 2 marks) will be given if they simply describe the multiplication without writing it down formally.

(c)

q_{end}	end					
q_4	PN	0	**1.0×0.4×1**	0.4×0×0.3	0.072×0×0.4	0.00504 × 0 × 0.2
q_3	V	0	1.0×0×0.4	**0.4×0.6×0.3**	0.072×0×0.3	0.00504 × 0.4 × 0.1
q_2	N	0	1.0×0×0.2	0.4×0.4×0.3	0.072×0.3×0.2	**0.00504×0.6×0.4**
q_1	ADJ	0	1.0×0×0	0.4×0×0.1	**0.072×0.7×0.1**	0.00504×0×0.3
q_0	start	1.0				
		<s>	John	likes	orange	dates
		o_0	o_1	o_2	o_3	o_4

The most likely sequence for the sentence is: PN V ADJ N. **Marking guide:** there are 16 cells to fill in, approximately 0.5 marks will be taken off for every wrong cell. 2 marks will be allocated for stating the correct POS-tag sequence.

(d) Unknown words are a problem as words for which no POS-tag exists will be not assigned any tag and $P(t_1^n|w_1^n)$ will be zero. The simplest possible unknown-word guessing is to pretend that each unknown word is ambiguous among all possible tags with equal probability.

Marking guide: 2.5 marks for explaining why unknown words are a problem and 2.5 marks for providing a solution.

3. (a) The grammar is in not CNF. Rules in CNF must be of the form $A \rightarrow BC$ or $A \rightarrow w$. Rules R2 and R5 are thus not in CNF and must be changed as follows:

R1	S	→	NP VP
R2	**S**	→	**V NP**
R2a	**S**	→	**cook**
R3	NP	→	Det NP
R4	NP	→	PN N
R5	**NP**	→	**French**
R6	VP	→	V NP
R7	Det	→	the
R8	N	→	cook | toast
R9	V	→	cook
R10	VP	→	cook
R11	PN	→	French

Marking guide: 1 mark for explaining why the grammar is not in CNF and 2 marks for changing the grammar correctly into CNF.

(b)

Cook S, V, VP, N [0,1]	the [0,2]	French VP [0,3]	toast VP [0, 4]
	Det [1,2]	 NP [1,3]	 NP [1,4]
		NP, PN [2,3]	NP [2,4]
			N [3,4]

There is one tree for the phrase:
[VP [V cook] [NP [Det the [NP [PN French [N toast]]]]]]

Marking guide: 2 marks for every cell of the grid filled correctly.

(c) A probabilistic CYK also assumes that the PCFG is in CNF. The modification is simple. Cells in the CKY matrix have a third dimension corresponding to each non-terminal that can be placed in this cell, and the value of the cell is then a probability for that non-terminal rather than a list of constituents. Each cell in the matrix is the probability of constituent A that spans positions i through j in the input.

Marking guide: 3 marks for explaining the modification and 3 marks for explaining how the matrix is filled.

(d) The Grammar G1 rules have now the following probabilities:

R1	S	→	NP VP	0.5
R2	**S**	→	**V NP**	0.25
R2a	**S**	→	**cook**	0.25
R3	NP	→	Det NP	0.33
R4	NP	→	PN N	0.33
R5	**NP**	→	**French**	0.33
R6	VP	→	V NP	0.5
R7	Det	→	the	1
R8	N	→	cook | toast	0.5
R9	VP	→	cook	0.5
R10	V	→	cook	1
R11	PN	→	French	1

And the chart looks as follows:

Cook S: 0.25, V:1, N:0.5, VP:0.5 [0,1]	the [0,2]	French VP: 0.5×0.33×1 S: 0.25×0.33×1 [0,3]	toast VP: 0.5×1× 0.055 S: 0.25 ×1× 0.055 [0, 4]
	Det: 1 [1,2]	NP: 0.33×1×0.33 [1,3]	NP: 1×0.33×0.166 [1,4]
		NP: 0.33, PN: 1 [2,3]	NP: 0.33 × 0.5 × 1 [2,4]
			N: 0.5 [3,4]

Marking guide: 3 marks for assigning the rule probabilities correctly and 3 marks for filling in the chart correctly.