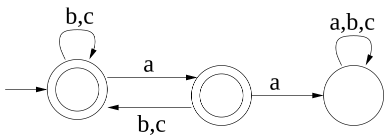
Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): Aug 2012
Brief notes on answers:

PART A

1. b.
2. e.
3. e. We want a string depending on k that belongs to L. An obvious choice would be $a^{2k}b^k$.
4. b. The grammar generates the given string, and is easily seen to be LL(1).
5. d.
6. a.
7. c. LHS is longer than RHS.
8. e. The halting set is r.e. but its complement is not.
9. c. The meaning of a sentence is determined by the meaning of its parts.
10. a. The sentence has 22 unique vocabulary items; the number of tokens is obtained by adding the individual word frequencies, only *the* and *by* are attested more than once, 3 and 2 times respectively. The number of types is 25.
11. d.
12. c.
13. d.
14. c.
15. b.
16. d.
17. b. LL(1) parsing can't be used e.g. if the attempt to build the LL(1) parse table gives rise to clashes.
18. d.
19. b. An HMM is a finite state automaton but there is no input, only output (emission).
20. d. (Note that 'can' is both an auxiliary and a noun in English.)

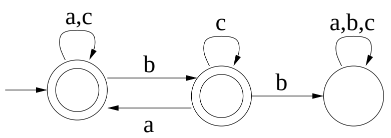
PART B

1. (a)



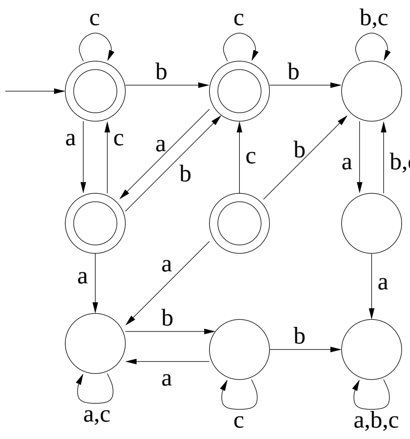
[Up to 3 marks.]

(b)



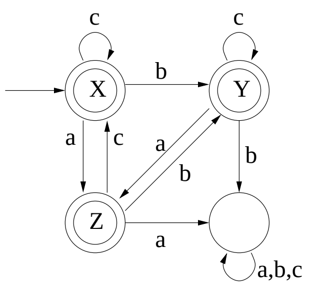
[Up to 3 marks.]

(c)



[This will be marked 'relative to' the answers given for (a) and (b). Up to 3 marks for evidence of correct understanding; 4 marks for getting the transitions right; 2 marks for the accepting states and initial state.]

(d)



[Up to 4 marks. Marks will be awarded if *either* the solution is correct outright, *or* it correctly minimizes the DFA given for (c).]

(e) The equations are

$$\begin{aligned} X &= cX + bY + aZ + \epsilon \\ Y &= cY + aZ + \epsilon \\ Z &= cX + bY + \epsilon \end{aligned}$$

Substituting for Z:

$$\begin{array}{rcl} X & = & (c+ac)X + (b+ab)Y + a + \epsilon \\ Y & = & (c+ab)Y + acX + a + \epsilon) \end{array}$$

By Arden's rule:

$$\begin{array}{rrcl} & Y & = & (c+ab)^*(acX + a + \epsilon) \\ \text{so} & X & = & (c+ac+(b+ab)(c+ab)^*ac)X \;+\; (b+ab)(c+ab)^*(a+\epsilon) + a + \epsilon \\ \text{so} & X & = & (c+ac+(b+ab)(c+ab)^*ac)^*((b+ab)(c+ab)^*(a+\epsilon) + a + \epsilon) \end{array}$$

(Solving the equations in a different order may lead to other expressions.)

[3 marks for formulating the equations; 3 marks for solving them. Again, this will be marked relative to the solution given for (d).]

2. (a) Bookwork. A grammar is LL(1) if, at each stage in the construction of a leftmost derivation for a given input string, there is only one possible choice of production to apply given the nonterminal to be expanded and the next input symbol. This leads to an efficient (linear-time) deterministic parsing algorithm. [Up to 2 marks for explanation of LL(1); 1 mark for the advantage.]

(b) There are five parse trees, as suggested by the following bracketings:

((fresh strawberry) and mango) in a hazelnut sauce
(fresh (strawberry and mango)) in a hazelnut sauce
fresh ((strawberry and mango) in a hazelnut sauce)
fresh (strawberry and (mango in a hazelnut sauce))
(fresh strawberry) and (mango in a hazelnut sauce)

[1 mark each.]

(c) Probably the simplest solution is as follows. Introduce additional nonterminals E, op, ops. The productions are

$$\begin{array}{rcl} D & \longrightarrow & E\ ops \\ E & \longrightarrow & \mathsf{Noun} \mid \mathsf{Adjective}\ E \\ ops & \longrightarrow & \epsilon \mid op\ ops \\ op & \longrightarrow & \mathtt{and}\ E \mid \mathtt{in\ a}\ D\ \mathtt{sauce} \end{array}$$

Other solutions are possible. [Award 0 to 5 marks.]

In general, LL(1) grammars are ill suited to natural languages because they artifically impose some fixed way of resolving ambiguities. For natural language, we typically want grammars that allow for ambiguities so that we can discriminate between alternative parses on other grounds. [2 marks.]

(d) For the above grammar, the First and Follow sets are

$$\begin{array}{ll} \mathsf{First}(D) = \{\mathsf{Noun}, \mathsf{Adjective}\} & \mathsf{Follow}(D) = \{\$, \mathtt{sauce}\} \\ \mathsf{First}(E) = \{\mathsf{Noun}, \mathsf{Adjective}\} & \mathsf{Follow}(E) = \{\$, \mathtt{sauce}, \mathtt{and}, \mathtt{in}\} \\ \mathsf{First}(ops) = \{\epsilon, \mathtt{and}, \mathtt{in}\} & \mathsf{Follow}(ops) = \{\$, \mathtt{sauce}\} \\ \mathsf{First}(op) = \{\mathtt{and}, \mathtt{in}\} & \mathsf{Follow}(op) = \{\$, \mathtt{sauce}, \mathtt{and}, \mathtt{in}\} \end{array}$$

[Up to 4 marks for First sets, up to 4 marks for Follow sets.]

(e) The parse table for the above grammar is:

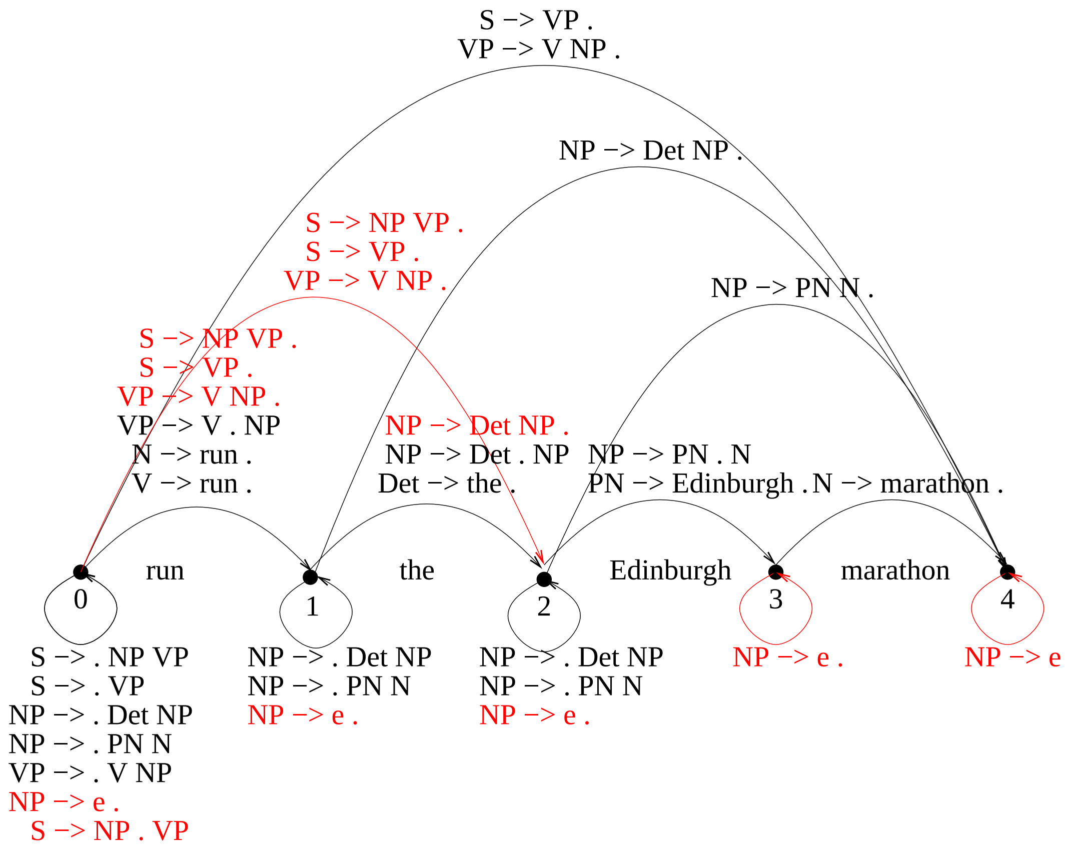
	Noun	Adjective	and	in	a	sauce	$
D	E *ops*	E *ops*					
E	Noun	Adjective E					
ops			*op ops*	*op ops*	ϵ	ϵ	
op			and E	in a D sauce			

[Up to 4 marks for evidence of understanding; another 3 marks for getting the details right.]

3. (a) The Earley chart can contain:

- Predictive edges of the form $A \to .BC\ [n, n]$
- Incomplete edges of the form $A \to B.C\ [n, m]$
- Complete edges of the form $A \to BC.\ [n, m]$

Marking guide: 1 mark for each question.



(b)

Marking guide: 2 marks for each chart position.

(c) Epsilon productions don't cause any particular problems for the Earley algorithm (except that they increase the size of the chart). The additional edges generated by the production NP $\to \epsilon$ are included in the solution for the previous questions in red.

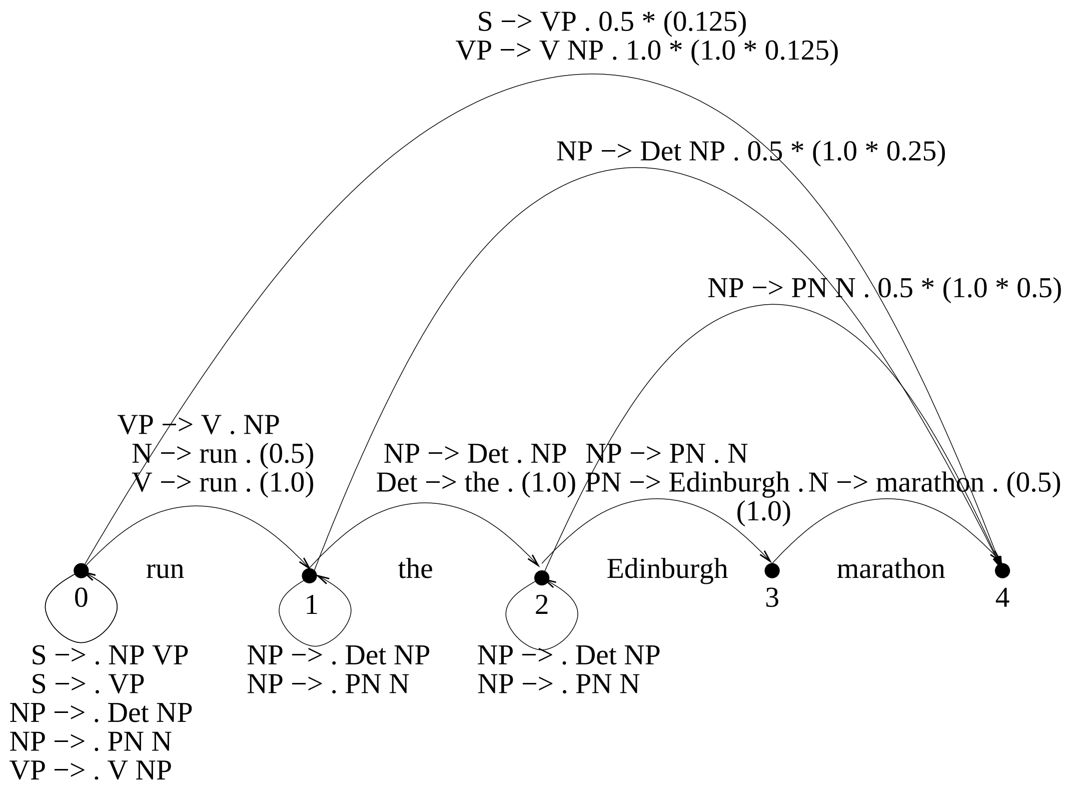
Marking guide: 2 marks for the statement that epsilon productions are not problematic, 3 marks for the red part of the chart.

(d) The simplest version of the probabilistic Earley algorithm (that doesn't compute prefix probabilities) works as follows:

For each completed edge $A \to BC$. compute a probability $P_e(A) = P(A \to BC)P_e(B)P_e(C)$, where $P(A \to BC)$ is the rule probability specified by the grammar and P_e are the edge probabilities computed by the algorithm.

Marking guide: 2 marks for the statement that we have to worry about completed edges only, 2 marks for the idea of using edge probabilities, 2 marks for their definition.

(e)


S −> VP . 0.5 * (0.125)
VP −> V NP . 1.0 * (1.0 * 0.125)
NP −> Det NP . 0.5 * (1.0 * 0.25)
NP −> PN N . 0.5 * (1.0 * 0.5)
VP −> V . NP
N −> run . (0.5)
V −> run . (1.0)
NP −> Det . NP
Det −> the . (1.0)
NP −> PN . N
PN −> Edinburgh . (1.0)
N −> marathon . (0.5)
run
the
Edinburgh
marathon
0
1
2
3
4
S −> . NP VP
S −> . VP
NP −> . Det NP
NP −> . PN N
VP −> . V NP
NP −> . Det NP
NP −> . PN N
NP −> . Det NP
NP −> . PN N