Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): August 2014
Brief notes on answers:

PART A

1. (a) Regular languages; context-free languages; context-sensitive languages; recursively enumerable languages.

   [4 marks: 1 mark each]

   (b) Languages recognised by DFAs; languages recognised by NFAs; languages defined by regular expressions.

   [3 marks: 1 mark each]

   (c) An NFA can be converted to a DFA using the subset construction.
   A DFA can be converted to a regular expression by solving a set of simultaneous equations (using Kleene algebra).
   A regular expression is directly translatable to an NFA (e.g., using ϵ transitions then eliminating them).

   [3 marks: 1 mark each. Enough to give the key idea; i.e., non-bracketed part.]

2. (a) Definitions of lexical classes:

```
IF    :  [iI]f | IF
THEN  :  [tT]hen | THEN
ELSE  :  [eE]lse | ELSE
NUM   :  [0-9]+
OP    :  == | <= | < | > | >= | <>
VAR   :  ([a-z]|[A-Z])([a-z]|[A-Z]|[0-9])*
```

   [8 marks: deduct 1 per error]

   [There plenty of room for interpretation and variation here, so good solutions may look very different from the above. However, marks may be deducted for implausible choices, even when consistent with the example in the question.]

   (b) The three keywords `IF`, `THEN` and `ELSE` need to have higher priority than `VAR`.

   [2 marks: award 1 if right idea but imperfection in presentation]

3. (a)
   - The root of the tree is labelled with S. [1 mark]
   - Each leaf is labelled with either a terminal or ϵ. [2 marks]
   - Each internal node is labelled with a nonterminal X for which there is a production $X \rightarrow \alpha$ in P such that either: $\alpha \neq \epsilon$ and the node's children are labelled with the symbols in α in turn; or $\alpha = \epsilon$ and the node has a unique child labelled with ϵ. [2 marks]

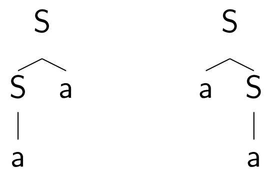
   [5 marks total]

   (b) There is some phrase x that has two distinct parse trees. [1 mark]

   (c) Grammar ($\Sigma = \{a\}$):

$$\mathsf{S} \rightarrow \mathsf{a} \mid \mathsf{a\,S} \mid \mathsf{S\,a}$$

The phrase *aa* has two distinct parse trees.



[4 marks: 1 for an ambiguous grammar, 1 for identifying ambiguous phrase, 2 marks for giving 2 distinct trees]

4. (a) The tagging obtained using the bigram tables is

   let/V her/OP duck/V fly/V

   (which is not the most natural tagging at all).

   [1 mark per correct tag; 1 mark overall for explanation.]

   (b) Bookwork. In an n-gram tagger, we look for the most frequent tag for each word given the tags already assigned to the preceding $n-1$ words. For $n = 4$, this can lead to more accurate results than bigram tagging, since more information about the context of the word comes into play. However, this supposes we have 4-dimensional frequency tables for each word type, and unless our corpus is very large indeed, these matrices may be too sparse to be really useful. [1 mark for concept of n-gram tagger, 1 mark for advantage, 1 mark for disadvantage. Other reasonable points accepted.]

   (c) Again bookwork. The Viterbi algorithm finds the tag sequence with highest *overall* probability, in contrast to bigram tagging, where the most probable local choice at one stage might be paid for by a very improbable choice at a later stage. For example, in the above tagging, the 'improbable' transition V– Vmight result in a low probability for the tag sequence as a whole, whereas the 'intended' tagging V PP N Vmight well emerge as the most probable. [1 mark for 'most probable overall'; 1 mark for some sensible reference to the example.]

5. (a) There are two parse trees. 'Tree 1' (corresponding to the natural interpretation) is

   (S (NP (A meterological)(N office))(V forecasts)(NP (A calm)(N seas)))

   This has probability $1.0 \times 0.3 \times 0.1 \times 0.2 \times 0.2 \times 0.3 \times 0.4 \times 0.4 \; = \; 576 \times 10^{-7}$. 'Tree 2' is

   (S (NP (NP (A meterological)(N office))(N forecasts))(V calm)(NP (N seas)))

   This has probability $1.0 \times 0.2 \times 0.3 \times 0.1 \times 0.2 \times 0.4 \times 0.2 \times 0.5 \times 0.4 \; = \; 192 \times 10^{-7}$. [1 mark for each parse tree; 1 mark for evidence of right method; 1 mark for each correct probability.]

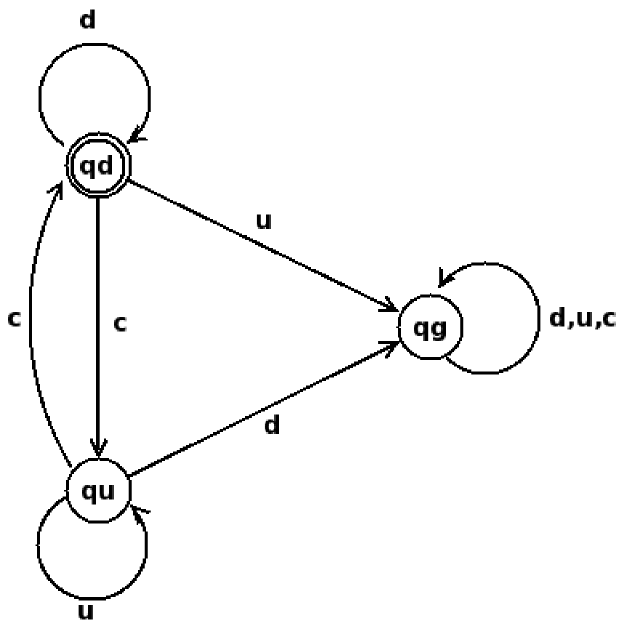
   (b) The most probable choice is 'calm'. To see this, first note that this is the only choice yielding two parse trees. For 'describe', we obtain a tree with exactly the same probability as Tree 2 above. For 'stormy', we obtain a tree more probable than Tree 1 by a factor of 0.5/0.4, i.e. 720×10^{-7}, but this is still less than the sum of the probabilities of Trees 1 and 2 (768×10^{-7}).

   [1 mark for the right answer, 4 marks for the explanation. Partial marks available for a wrong answer if some valid points are made.]

PART B

6. [A similar question to this was scrutinised last year but wasn't used because of the ITO burglary]

   (a) The DFA is below. (Corrections: qd is the start state. qu is also acceping.)



   [4 marks: deduct 1 per mistake]

   (b) The equations are:

$$\begin{aligned}X_d &= dX_d + cX_u + uX_g + \epsilon\\ X_u &= uX_u + cX_d + dX_g + \epsilon\\ X_g &= dX_g + uX_g + cX_g\end{aligned}$$

   The language we are interested in is X_d.
   First derive

$$X_g = \emptyset$$

   (This can be done using Arden's rule, but enough to spot the equality.)
   Thus the equations simplify to

$$\begin{aligned}X_d &= dX_d + cX_u + \epsilon\\ X_u &= uX_u + cX_d + \epsilon\end{aligned}$$

   By Arden' rule:

$$X_u = u^*(cX_d + \epsilon)$$

   So

$$\begin{aligned}X_d &= dX_d + cu^*(cX_d + \epsilon) + \epsilon\\ X_u &= (d + cu^*c)X_d + cu^* + \epsilon\end{aligned}$$

   Again by Arden' rule:

$$X_u = (d + cu^*c)^*(cu^* + \epsilon)$$

(c) *Not regular.*

[1 mark]

We show ¬ P (the negation of the pumping property).

Suppose $k \geq 0$.

Consider $x = \epsilon$, $y = d^k$ and $z = cu^k$. Then $xyz = d^k cu^k \in L$ and clearly $|y| \geq k$.

Suppose $y = uvw$ where $|v| \geq 1$.

Then $uv^0w = uw = d^m$ for some $m < k$. Whence $xyv^0wz = d^m cu^k \notin L$ since $m < k$.

Thus the pumping property fails for $i = 0$.

[6 marks: in proportion to completeness of argument]

(d) $\mathsf{S} \rightarrow d\ c\ u \mid d\ \mathsf{S}\ u$.

[3 marks: deduct 1 per mistake]

(e) The language is context sensitive (but not context free).

[1 mark: for unbracketed part]

7. (a) Parse table:

	`add`	`take-away`	(	)	int	\$
Exp			Exp1 Ops		Exp1 Ops	
Ops	add Exp1 Ops	`take-away` Exp1 Ops				ϵ
Exp1			(Exp)		int	

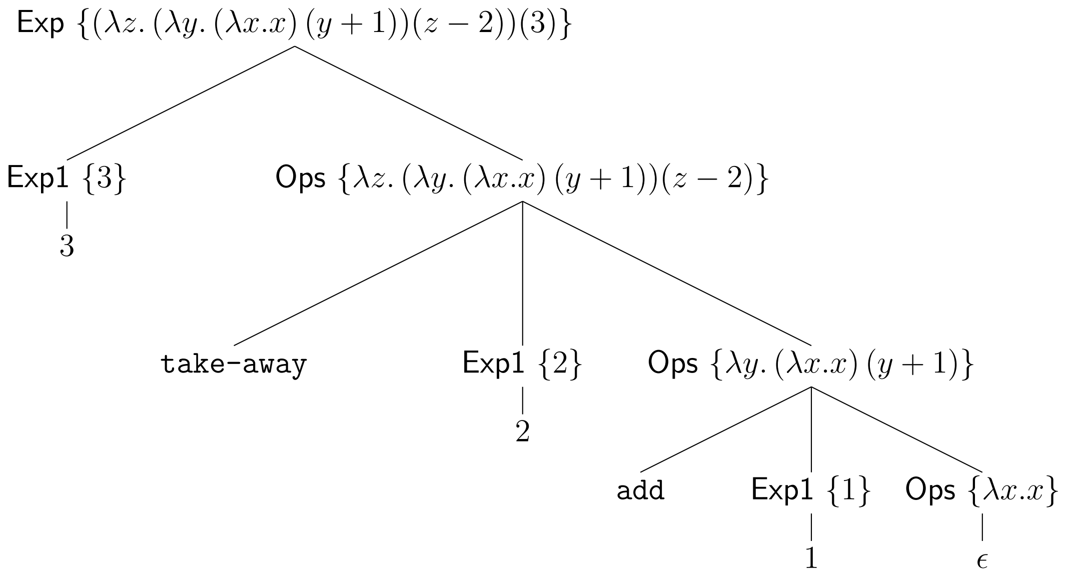
[6 marks: deduct 1 per mistake]

(b) Algorithm execution:

action	unread input	stack
	2 add 1 \$	Exp
Exp $\rightarrow$ Exp1 Ops	2 add 1 \$	Exp1 Ops
Exp1 $\rightarrow$ int	2 add 1 \$	int Ops
match int	add 1 \$	Ops
Ops $\rightarrow$ add Exp1 Ops	add 1 \$	add Exp1 Ops
match add	1 \$	Exp1 Ops
Exp1 $\rightarrow$ int	1 \$	int Ops
match int	\$	Ops
Ops $\rightarrow$ ϵ	\$	ϵ

[6 marks: deduct 1 per mistake]

(c) Annotated tree:



Exp $\{(\lambda z.\,(\lambda y.\,(\lambda x.x)\,(y+1))(z-2))(3)\}$
Exp1 $\{3\}$
3
Ops $\{\lambda z.\,(\lambda y.\,(\lambda x.x)\,(y+1))(z-2)\}$
take-away
Exp1 $\{2\}$
2
Ops $\{\lambda y.\,(\lambda x.x)\,(y+1)\}$
add
Exp1 $\{1\}$
1
Ops $\{\lambda x.x\}$
ϵ

[6 marks: 1 for sufficient correct annotation at bottom; 5 marks for the 3 main annotations, deducting 1 mark per mistake. Penalise by just 1 mark for incorrect tree structure.]

(d) β-reductions:

$$\begin{aligned}
&(\lambda z.\,(\lambda y.\,(\lambda x.x)\,(y+1))(z-2))(3)\\
&\quad\rightarrow\ (\lambda y.\,(\lambda x.x)\,(y+1))(3-2)\\
&\quad\rightarrow\ (\lambda x.x)\,((3-2)+1)\\
&\quad\rightarrow\ (3-2)+1
\end{aligned}$$

[4 marks: 1 for having roughly right idea about β-reduction, plus 1 mark per correct step]

[Reduction of the arithmetic operations, e.g., to produce correct result of 2, is allowable. This will neither be rewarded or penalised. However, incorrect such reductions will be penalised.]

(e) Just two productions need a change of semantics:

Ops $\rightarrow$	`add` Exp1 Ops	$\{\ \lambda x.\,x\ +$ Ops.Sem (Exp1.Sem) $\}$
Ops $\rightarrow$	`take-away` Exp1 Ops	$\{\ \lambda x.\,x\ -$ Ops.Sem (Exp1.Sem) $\}$

[3 marks: 1 for realising that just these 2 productions need the alteration; 2 for giving the correct definitions]

8. The parameterized grammar is:

	S	$\rightarrow$	NP[n] VP[n,a,x] , TagQ[n,a,x]
	NP[n]	$\rightarrow$	The N[n]
	VP[n,a,x]	$\rightarrow$	Aux[n,a] NegOpt1[x] V[a]
	TagQ[n,a,x]	$\rightarrow$	Aux[n,a] NegOpt2[x] Pron[n] ?
	NegOpt1[pos]	$\rightarrow$	ϵ
	NegOpt1[neg]	$\rightarrow$	-n't
	NegOpt2[pos]	$\rightarrow$	-n't
	NegOpt2[neg]	$\rightarrow$	ϵ
(a)	N[sing]	$\rightarrow$	eagle | rocket
	N[plur]	$\rightarrow$	eagles | rockets
	Aux[sing,have]	$\rightarrow$	has
	Aux[plur,have]	$\rightarrow$	have
	Aux[n,had]	$\rightarrow$	had, etc.
	V[have/had]	$\rightarrow$	landed | flown
	V[does/did]	$\rightarrow$	land | fly
	Pron[sing]	$\rightarrow$	it
	Pron[plur]	$\rightarrow$	they

Minor variations are possible. E.g. I would accept a solution that also included NegOpt2[pos] $\rightarrow \epsilon$.

[11 marks. Quite challenging, so be fairly generous where there is evidence of understanding.]

(b) The CYK parse chart (using a few abbreviations) is:

	The	rocket	did	-n't	fly	,	did	it	?
The		NP[sing]							S
rocket		N[sing]							
did			Aux[s/p,did]		VP[s/p,did,neg]				
-n't				NegOpt1[neg], NegOpt2[pos],					
fly					V[does/did]				
,									
did							Aux[s/p,did]	TagQ[s,did,neg]	
it								Pron[s]	
?									

(The use of CYK-style charts for non-CNF grammars is slightly non-standard, but they will have seen it in lectures.)

[About 2 marks per 3 correct entries; be fairly generous.]

(c) Modify the above grammar by removing TagQfrom the first production, and use this to parse the user's input. The resulting parse tree will reveal the appropriate values for n,a,x. Using these values, now expand TagQ[n,a,x] in the unique way possible using the parameterized rules; the result will be the appropriate tag question.

[1 mark for identifying need for a parsing and generation phase. 2 marks for remainder of explanation.]