Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): Aug 2015
Brief notes on answers:

1. (a) Lexing: The input is program text [1].The output is a stream of lexemes paired with their lexical classes [1]. For example the expression `xyz 22` would result two lexeme stream `xyz, 22` with lexical classes `VAR` of variables and `INT` of integer literals respectively [1].

   Parsing: The input is stream of lexical classes (lexemes good enough) [1]. The output is a syntax tree [1]. For example, `VAR INT` might produce a tree such as `(EXP (FUN (VAR)) (ARGS (ARG INT) (ARGS` ϵ`)))` expressing the syntax of an expression formed as a (possibly iterated) function application [1].

   Type-checking. The input is a syntax tree [1]. The output is yes/no response as to whether the program type-checks, together with diagnostic error messages if not [1]. In the example above it would be checked that the variable `xyz` has a type of the form `Integer ->` τ, for some τ, allowing the function application to be formed.

   [Only minimal details are required. The examples need to be faithful the idea of lexing/parsing/type-checking, not necessarily faithful to any particular language.]

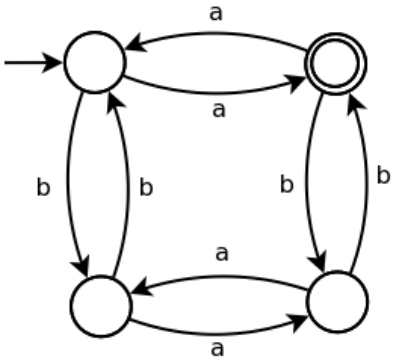
   (b) E.g., Compilation(/interpretation/evaluation/execution) [1].

2. (a) Let $M_1 = (Q_1, s_1, F_1, \delta_1)$ and $M_2 = (Q_2, s_2, F_2, \delta_2)$. The desired DFA has

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ s &= (s_1, s_2) \\ F &= F_1 \times (Q_2 - F_2) \end{aligned}$$

   and $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ whenever $q_1 \xrightarrow{a} q_1'$ in δ_1 and $q_2 \xrightarrow{a} q_2'$ in δ_2. [4: 1 mark for each component]

   (b) The resulting DFA:



   [4 marks: deduct 1 per mistake]

   (c) No it is not possible: context-free languages are not closed under complement [1]. An example is $\Sigma^* - \{xx \mid x \in \Sigma^*\}$ which is context free but its complement not [1].

   [The first mark needs more context than just the answer "no"]

   [The counterexample is bookwork]

3. (a) The PDA execution:

state	stack	unread input
$q1$	$\perp$	$aaaba$
$q1$	a	$aaba$
$q1$	$a\,a$	aba
$q1$	$a\,a\,a$	ba
$q2$	$a\,a\,a$	a
$q2$	$a\,a$	ϵ
$q2$	a	ϵ
$q2$	ϵ	ϵ

[8 marks: in principle 1 per step]

(b) The language is:

$$\{a^n b a^m \mid 1 \leq n, 0 \leq m \leq n\}$$

[2 marks: award 1 if idea right but some error in detail]

4. (a) Zipf's law is an empirical law stating that in a typical corpus of text, the number of occurrences of a given word type is (approximately) inversely proportional to its frequency rank. [Up to 2 marks]

(b) Suppose k is a constant such that the word with frequency rank r occurs around k/r times. We are given that $k + k/2 + k/3 + k/4 = (25/12)k = 2500$, whence $k = 1200$. To a reasonable approximation, $1/21 + \cdots + 1/30 = 10/25 = 2/5$, so the required number of tokens is around $2k/5 = 480$. [Up to 3 marks.]

(c) To compute the transition probability from the POS X to the POS Y, look at all occurrences in the corpus of tokens tagged as X, and calculate what proportion of these are followed by a token tagged as Y. To compute the emission probability for a POS X as a word w, look at all occurrences of tokens tagged as X, and calculate what proportion of these are the word w. [1 mark for each of these; 1 further mark for clarity of explanation.]

(d) We would expect a higher proportional accuracy for the infrequent word types. This is because most infrequent words have only one possible POS tag, whereas several of the most frequent words are grammatical function words with several possible tags. [2 marks. Any other logically reasonable point will be accepted here.]

5. The CYK parse chart is:

	boat	man	watches	fish
boat	NP 1	NP 1	NP 2, S 1	NP 5, S 2
man		NP 1, V 1	NP 1, VP 1	NP 2, VP 1, S 1
watches			NP 1, V 1	NP 1, VP 1
fish				NP 1, V 1

Up to 7 marks for the non-terminal entries; up to 3 marks for the numbers.

6. (a) The production S $\rightarrow$ S b is left recursive [1].

(b) For example

$$\begin{aligned} \mathsf{S} &\rightarrow \mathsf{S1\ T} \\ \mathsf{S1} &\rightarrow \mathsf{a} \mid \mathsf{(\ S\ S\ )} \\ \mathsf{T} &\rightarrow \epsilon \mid \mathsf{b\ T} \end{aligned}$$

[5 marks: in proportion]

(c)

$$\begin{aligned} \text{First}(\mathsf{S}) &= \{\mathsf{a}, \mathsf{(}\} \\ \text{First}(\mathsf{S1}) &= \{\mathsf{a}, \mathsf{(}\} \\ \text{First}(\mathsf{T}) &= \{\epsilon, \mathsf{b}\} \end{aligned}$$

[3: in principle 1 mark each]

(d)

$$\begin{aligned} \text{Follow}(\mathsf{S}) &= \{\mathsf{a}, (,), \$\} \\ \text{Follow}(\mathsf{S1}) &= \{\mathsf{a}, \mathsf{b}, (,)\} \\ \text{Follow}(\mathsf{T}) &= \{\mathsf{a}, (,), \$\} \end{aligned}$$

[3: in principle 1 mark each]

(e) Parse table:

	a	b	(	)	$
S	S1 T		S1 T		
S1	a		(S S)		
T	ϵ	b T	ϵ	ϵ	ϵ

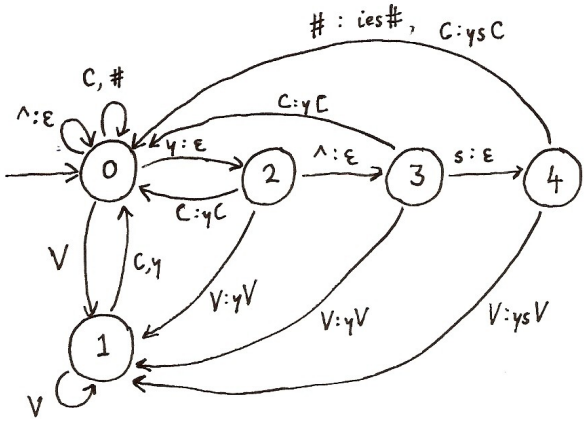
[6 marks: deduct 1 per mistake]

(f) Algorithm execution:

action	unread input	stack
	(a a) $	S
S → S1 T	(a a) $	S1 T
S1 → (S S)	(a a) $	(S S) T
match (	a a) $	S S) T
S → S1 T	a a) $	S1 T S) T
S1 → a	a a) $	a T S) T
match a	a) $	T S) T
T → ϵ	a) $	S) T
S → S1 T	a) $	S1 T) T
S1 → a	a) $	a T) T
match a	) $	T) T
T → ϵ	) $	) T
match)	$	T
T → ϵ	$	ϵ

[7 marks: in proportion]

7. (a) The following transducer does the job. Here V stands for any of the letters a,e,i,o,u, and C stands for any letter except a,e,i,o,u,y. A single symbol (without a colon) denotes a transition with the same input and output.



[6 marks for evidence of understanding, 6 marks for the details. Only deduct 1 or 2 marks if any or all of the following transitions are missing or incorrect: $3 \to 0$, $3 \to 1$, $4 \to 1$, and the transition $4 \to 0$ labelled with C:ysC.]

(b) he# parses as he#
plies# parses as plies#, plie^s#, ply^s#
trades# parses as trades#, trade^s#
[Up to 4 marks; deduct 1 mark per mistake.]

(c) he: PRN
plies: NPL, V3S
trades: NPL, V3S
[1 mark for each word.]

(d) The Viterbi matrix is as follows. Cells that are not consistent with the possibilities listed in part (c) are left empty.

	he	plies	trades
PRN	0.4		
NPL		0.4x0.1=0.04	↙ 0.16x0.5=0.08
V3S		0.4x0.4=0.16	↖ 0.04x0.1=0.004

So the most probable tag sequence is PRN V3S NPL. [4 marks for the numbers, 1 mark for the backtrace pointers, 1 mark for the correct tagging. Minor clerical errors will not be heavily penalized.]

8. (a) The complete grammar with semantic attachments is as follows:

S	→	Name is NP	{NP.Sem(Name.Sem)}
NP	→	Name	$\{\lambda x.\, x = \mathsf{Name.Sem}\}$
NP	→	NP 's Rel	$\{\lambda x.\exists y.\, \mathsf{NP}.Sem(y)\ \&\ \mathsf{Rel}.Sem(x,y)\}$
Rel	→	Rel1	{Rel1.Sem}
Rel1	→	father	$\{\lambda xy.\ Father(x,y)\}$
Rel1	→	daughter	$\{\lambda xy.\ Female(x)\ \&\ (Father(y,x) \vee Mother(y,x))\}$
Rel1	→	brother	$\{\lambda xy.\ Male(x)\ \&\ \neg(x=y)\ \&$ $(\exists z.Father(z,x) \wedge Father(z,y))\ \&$ $(\exists w.Mother(w,x) \wedge Mother(w,y))\}$
Rel1	→	grandchild	$\{\lambda xy.\exists z.\ (Father(y,z) \vee Mother(y,z))\ \&$ $(Father(z,x) \vee Mother(z,x))\}$
Name	→	Alice	$\{Alice\}$, etc.

[3 marks each for the clauses for 'brother' and 'grandchild'; 2 marks for the clause for NP 's Rel; roughly 1 mark each for the remaining clauses. Other solutions are equally acceptable for some of these clauses.]

(b) The root node of the tree will be annotated with

$$(\lambda x.\exists y.(\lambda x.x = Alice)(y)\ \&\ (\lambda xy.Father(x,y))(x,y))(Brian)$$

and the other nodes will be annotated with subexpressions of this in the evident way. [Roughly 1 mark per node.]

(c) The above λ-expression β-reduces in three steps (which may be done in various orders) to

$$\exists y.\ y = Alice\ \&\ Father(Brian, y)$$

[1 mark per reduction step.]

(d) We may define

$$\overline{R}(x,y)\ \equiv\ R(x,y)\ \&\ \forall w.\, R(w,y) \Rightarrow w = x$$

This suggests the semantic clause

$$\mathsf{Rel} \rightarrow \text{only Rel1} \qquad \{\lambda xy.\ \mathsf{Rel.Sem}(x,y)\ \&\ \forall w.\, \mathsf{Rel.Sem}(w,y) \Rightarrow w = x\}$$

[1 mark for the formula for $\overline{R}$, 2 marks for the semantic clause.]