Module Title: Informatics 2A
Exam Diet (Dec/April/Aug): Aug 2016
Brief notes on answers:

1. (a) The equations are

$$\begin{aligned} X_p &= 0X_s + (1-9)X_r + -X_q \\ X_q &= (1-9)X_r \\ X_r &= \epsilon + (0-9)X_r \\ X_s &= \epsilon \end{aligned}$$

[2 marks for the X_p equation, 1 mark each for X_q, X_r]

(b) Solving for X_r, X_q, X_p in turn, we have

$X_r = (0-9)^*, \quad X_q = (1-9)(0-9)^*, \quad X_p = 0 + (1-9)(0-9)^* + -(1-9)(0-9)^*$

[3 marks; roughly 1 mark for each of X_r, X_q, X_p]

(c) The sequence will get lexed as

$$0\,;\,0\,;\,10\,;\,-3450$$

but soon after this we get an error due to the absence of a 0 transition from q. [2 marks for the partial lexing, 1 mark for pinpointing the error]

2. (a) The three standard reasons are:
   - ambiguity (as exemplified by $S \rightarrow S + S$)
   - shared left prefixes (e.g. $S \rightarrow a \mid aT$)
   - left recursion (e.g. $S \rightarrow Sa$)

   [1 mark each for naming the problems, 1 mark each for the examples.]

(b) A suitable grammar is:

$$\begin{aligned} S &\rightarrow T\ R \\ R &\rightarrow \epsilon \mid + S \\ T &\rightarrow id\ U \\ U &\rightarrow \epsilon \mid (\ S\ ) \end{aligned}$$

[2 marks for curing the ambiguity in line 1; 2 marks for curing the left recursion in line 2.]

3. (a) Three criteria: notional (relying on the meaning of the word), morphological/formal (relying on morphological structure) and distributional (relying on its context). [1 mark per criterion.]

(b) Open word classes are productive, they can have words add to them over time. They denote mostly semantic content. Closed word classes do not grow very quickly over time. They are mostly function words. [2 marks for open classes, 2 marks for closed classes]

(c) The formula for Zipf's law is $f = k/r$ [1 mark]. In this case, inspection shows that the data conform to Zipf's law, taking $k = 360000$ [2 marks].

4. (a) "Every boy has access to a door". Two logical forms:

   $\forall x.(\text{boy}(x) \Rightarrow \exists y.(\text{door}(y) \wedge \text{hasAccessTo}(x, y)))$

   $\exists y.(\text{door}(x) \wedge \forall x.(\text{boy}(x) \wedge \text{hasAccessTo}(x, y)))$

   For "Every boy has access to Door 3", we can only use the following interpretation because it refers to a specific door:

   $\exists y.(\text{door3}(x) \wedge \forall x.(\text{boy}(x) \wedge \text{hasAccessTo}(x, y)))$

   [1 mark per correct logical form]

   (b) The chart is:

V	0	$0.1 \times 0.05 = 0.005$	$0.005 * 0.1 * 0.9 < 0.28 * 0.4 * 0.9 = 0.1008$	$0.000504 = 0.1008 * 0.1 * 0.05 > 0.0056 * 0.4 * 0.05$
N	0	$0.7 \times 0.4 = 0.28$	$0.005 * 0.5 * 0.2 < 0.28 * 0.1 * 0.2 = 0.0056$	$0.02016 = 0.1008 * 0.5 * 0.4 > 0.0056 * 0.1 * 0.4$
$\langle$ s $\rangle$	1.0	0	0	0
		boys	access	doors

   The POS sequence is N V N.

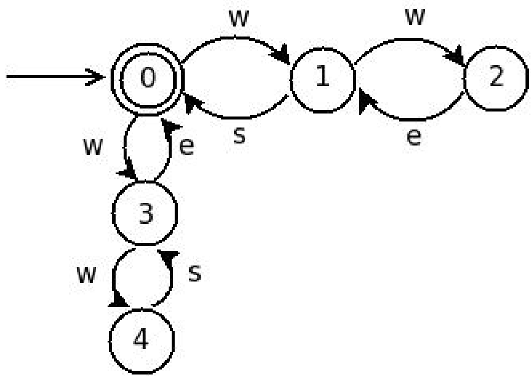
   [6 marks for the chart, being lenient on minor calculation errors. 1 mark for the POS sequence.]

5. (a)
```
def reverse(string):
    revstr = ""
    length = len(string)
    for index in range(length):
        revstr += string[(length-index)-1]
    return revstr
```
   [Up to 4 marks, being not too harsh on minor syntactic errors]

   (b)
```
def isPalindrome(string):
    return True if (string == reverse(string)) else False
```
   [Up to 3 marks; ditto]

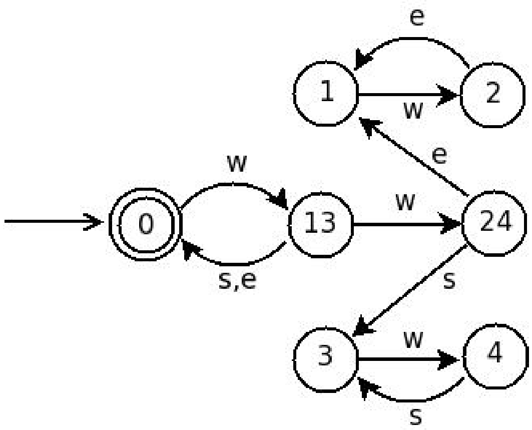
   (c) This is not a regular language since it requires an "unbounded memory". We need to be able to remember the first part of the string, which can be arbitrarily long, in order to match it against the second part. We can show this formally using the pumping lemma. [3 marks for evidence of understanding; the answer need not be too detailed or formal.]

6. (a) The state diagram is:



(The order of numbering the states is immaterial.) [3 marks for evidence of understanding; 3 more marks for correct details]

(b) Omitting the garbage state (which corresponds to $\emptyset$), the state diagram for the corresponding DFA is



[**Added later:** There should also be an s-transition from 1 to 0, and an e-transition from 3 to 0.] A solution should also show a garbage state as the target for some transition missing from the above diagram. [5 marks for right idea; 4 more marks for details. Deduct just 1 mark if garbage state not included.]

(c) M is minimal if $L_q \neq L_{q'}$ whenever $q \neq q'$. [2 marks for right idea, 1 mark for precise expression.]

(d) The DFA is minimal. The strings ϵ, e, s, es, se suffice to show this, since any two states differ with respect to their acceptance of at least one of these strings. That is, no two rows in the following table are identical (where $*$ denotes acceptance):

	ϵ	e	s	es	se
0	$*$				
13		$*$	$*$		
24				$*$	$*$
1			$*$		
2				$*$	
3		$*$			
4					$*$
$\emptyset$					

7. (a) The parse table is as follows (the columns for `else` and `endif` are identical to the one for `then`).

	+	==	`if`	`then`	*Int*	*Real*	*String*	$
Exp			Exp1 Ops		Exp1 Ops	Exp1 Ops	Exp1 Ops	
Ops	+Exp	ϵ		ϵ				ϵ
Exp1			`if` $\cdots$		*Int*	*Real*	*String*	
Cond			Exp == Exp		Exp == Exp	Exp == Exp	Exp == Exp	

[5 marks for evidence of reasonable understanding. Another 5 marks for the details; deduct roughly 0.5 marks for each incorrect cell.]

(b) Without `endif`, certain strings become ambiguous because the extent of an `if` expression cannot be determined. E.g.

`if` $3 == 4$ `then` 5 `else` $6 + 7$

The rules Ops → +Exp and Ops → ϵ will now be competing for the cell (Ops, +) in the table. [1 mark for the problem, 1 mark for the example, 2 marks for the parse table clash.]

(c) Typing rules are as follows:

- Exp1 phrases of the form *Int*, *Real*, *String* have type `Int`, `Real`, `String` respectively.
- An Exp1 phrase `if` C `then` E `else` E' `endif` is well-typed iff C has type `Bool` and E, E' have the same type t. In this case, the whole $\Longleftrightarrow$ expression has type t.
- A Cond phrase $E == E'$ is well-typed iff E, E' have the same type t. In this case, the Cond phrase has type `Bool`.
- An Exp phrase EO (of form Exp1 Ops) has type t' if E has some type t and t`->`t' is a possible type for O.
- The possible types of the Ops phrase ϵ are t`->`t for any t.
- For an Ops phrase $+E$, there are three subcases:
  - If E has type `String`, then $+E$ has type `String->String` (only).
  - If E has type `Int`, then $+E$ has the types `Int->Int` and `Real->Real`.
  - If E has type `Real`, then $+E$ has the types `Int->Real` and `Real->Real`.

[Up to 11 marks according to how many of the ingredients are in place. Any reasonable way of expressing the rules will be accepted. The number of marks allocated to this part takes account of the time it will take to digest the instructions.]

8. (a) The grammar is not recursive and as such generates a regular (even finite) language. Here is a regular expression for it, where D, N, Aux and Prep stand for the set of words that can be generated from these part-of-speech tags:

$$(\mathrm{DN} + \mathrm{N})(\mathrm{V} + \mathrm{VDN} + \mathrm{VN} + \mathrm{Aux\,VN} + \mathrm{Aux\,VDN})(\epsilon + \mathrm{Prep\,N} + \mathrm{Prep\,DN})$$

[2 marks for seeing that it is regular. 3 marks for the reason and a suitable regular expression.]

(b) The following is the adjusted grammar:

S → NP[sbjSg] VP[sg] | NP[sbj] VP[sg] PP
S → NP[sbjPl] VP[pl] | NP[sbjPl] VP[pl] PP[pl]
NP[sbjSg] → D[sg] N[sg] | N[sg]
NP[sbjPl] → D[Pl] N[Pl] | N[Pl]
N[pl] → boys | dogs
N[sg] → boy | dog
VP[pl] → V[base] | V[base] NP[obj] | Aux[pl] V[base] NP[obj]
VP[sg] → V[base] | V[sg] NP[obj] | Aux[sg] V[base] NP[obj]
Aux[sg] → doesn't
Aux[pl] → don't
NP[obj] → D N | N
V[sg] → stands | sits | chases
V[base] → stand | sit | chase
PP → Prep NP[prp]
NP[prp] → D N | N
Prep → in | on
N → table | tables
N → boy | boys
D[sg] → the | a | this| D[pl] → the | these|

[4 marks for the general idea of subject-verb agreement. 7 marks for a fully correct answer.]

(c) Any sentence which has the same bag of rules will get identical probability. For example, "The boy sits on the table" versus "The table sits on the boy". [4 marks for a suitable pair of sentences. 2 more marks for explaining the reason in terms of bags of rules.]

(d) The following is a table of counts for each rule with the total probabilities:

S → NP[sbj] VP | 3 | 3/4
S → NP[sbj] VP PP | 1 | 1/4
NP[sbj] → D N | 3 | 3/4
NP[sbj] → N | 1 | 1/4
N → boy | 1 | 1/6
N → boys | 2 | 1/3
N → dog | 1 | 1/6
N → dogs | 2 | 1/3
VP → V | 1 | 1/3
VP → V NP[obj] | 1 | 1/3
VP → Aux V NP[obj] | 1 | 1/3
Aux → doesn't | 1 | 1
Aux → don't | 0 | 0
NP[obj] → D N | 2 | 1
NP[obj] → N | 0 | 0
V → stand | 2 | 1/2
V → sit | 0 | 0
V → stands | 0 | 0
V → sits | 0 | 0

V → chase | 2 | 1/2
V → chases | 0 | 0
PP → Prep NP[prp] | 1 | 1
NP[prp] → D N | 0 | 0
NP[prp] → N | 1 | 1
Prep → in | 0 | 0
Prep → on | 1 | 1
N → table | 0 | 0
N → tables | 1 | 1/4
N → boy | 1 | 1/4
N → boys | 2 | 1/2
D → the | 4 | 4/5
D → a | 0 | 0
D → these | 1 | 1/5
D → this | 0 | 0

[7 marks. There are 34 rules, so about 0.2 marks per correct rule.]