Module Title: Informatics 1A
Exam Diet (Dec/April/Aug): August 2004
Brief notes on answers:

Part A FUNCTIONAL PROGRAMMING

1.

```
-- (a)
unlucky :: Date -> Bool
unlucky (d,m,y) =   dayOfWeek (d,m,y) == 5 && d == 13
                 || d == 6 && m == 6 && y `mod` 10 == 6

-- (b)
showDayOfWeek :: DayOfWeek -> String
showDayOfWeek 0  =  "Sunday"
showDayOfWeek 1  =  "Monday"
showDayOfWeek 2  =  "Tuesday"
showDayOfWeek 3  =  "Wednesday"
showDayOfWeek 4  =  "Thursday"
showDayOfWeek 5  =  "Friday"
showDayOfWeek 6  =  "Saturday"

-- (c)
showDate :: Date -> String
showDate (d,m,y)  =  show d ++ "/" ++ show m ++ "/" ++ show y

-- (d)
showDay :: Date -> String
showDay date =  showDayOfWeek (dayOfWeek date) ++
                if unlucky date then "(unlucky)" else ""
```

2.
```
-- (a)
kind :: Char -> Char
kind c | isDigit c   =  '1'
       | isAlpha c && isLower c  =  'a'
       | isAlpha c && isUpper c  =  'A'
       | otherwise =  '.'

-- (b)
kinds :: String -> String
kinds cs  =  [ kind c | c <- cs, isAlpha c || isDigit c ]

-- (c)
kinds' :: String -> String
kinds' []  =  []
kinds' (c:cs) | isAlpha c || isDigit c  =  kind c : kinds' cs
              | otherwise               =  kinds' cs
```

3.
```
-- (a)
ascending :: [Int] -> Bool
ascending xs  =  and [ x < y | (x,y) <- zip xs (tail xs) ]

-- (b)
ascending' :: [Int] -> Bool
ascending' []  =  True
ascending' [x]  =  True
ascending' (x:y:zs)  =  (x<y) && ascending' (y:zs)
```

Part B COMPUTATION AND LOGIC

4. An appropriate way of applying the proof rules is as follows (where $\mathcal{F}$ is the original list of axioms):

   - $\mathcal{F} \vdash a \rightarrow b$ by *imp_intro*
   - $[a|\mathcal{F}] \vdash b$ by *imp_elim* $(((f\ and\ e) \rightarrow b) \in \mathcal{F})$
   - $[a|\mathcal{F}] \vdash (f\ and\ e)$ by *and_intro*
   - $[a|\mathcal{F}] \vdash f$ by *imp_elim* $((d \rightarrow f) \in \mathcal{F})$
   - $[a|\mathcal{F}] \vdash d$ by *imp_elim* $(((a\ or\ c) \rightarrow d) \in \mathcal{F})$
   - $[a|\mathcal{F}] \vdash (a\ or\ c)$ by *or_intro_left*
   - $[a|\mathcal{F}] \vdash a$ by *immediate*
   - $[a|\mathcal{F}] \vdash e$ by *imp_elim* $((d \rightarrow e) \in \mathcal{F})$
   - $[a|\mathcal{F}] \vdash d$ as above

5. (a) An appropriate truth table is as follows:

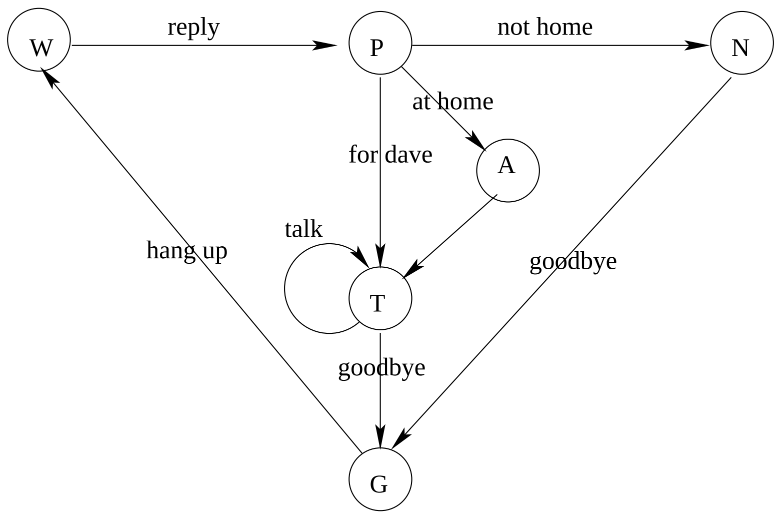
			1	2	3	4	5	
a	b	not(b)	not(not(b))	not(a)	2 or b	not(3)	1 or 4	5 or 2
t	t	f	t	t	t	f	t	t
t	f	t	f	f	f	t	t	t
f	t	f	t	t	t	f	t	t
f	f	t	f	t	t	f	f	t

   (b) The translated expression in full is $(not(not(b))\ or\ not(not(a)\ or\ b))\ or\ not(a)$. I expect to see each step of the translation so that each rule application can be checked.

6. (a) A basic set of appropriate states might be:

W	dave is waiting for a call
P	Dave has found out who the caller requests
N	The requested person is not at home
A	The requested person is at home
T	Someone is talking to the caller
G	Dave has said goodbye

   (b) An example of the right sort of machine is as follows:



(c) It is deterministic if every arc leaving a state leads to a unique successor state. The answer should show whether or not this is true for the machine given.

7. (a) The basic answer is that the machine needs be able to determine an arbitrarily large sequence of characters on the "left" of the string in order to balance the "right" hand side.

   (b) One way is to give it memory so that the progress through nodes can take place in context.

8. Appropriate regular expressions are:

   (a) $b * |b * ab * |b * ab * ab*$

   (b) $a|a(a|b) * a$

   (c) $((ba * b)|a*)*$