Module Title: INFORMATICS 1A
Exam Diet (Dec/April/Aug): DECEMBER 2005
Brief notes on answers:

1.
```
type Day = Int
type Month = String
type Date = (Day, Month)

months :: [Month]
months =  ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
           "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
```

(a)
```
index :: Month -> Int
index m  =  head [ i | (i,m') <- zip [1..] months, m==m' ]
```
or
```
index :: Month -> Int
index "Jan" = 1
index "Feb" = 2
...
index "Dec" = 12
```

(b)
```
sensible :: Date -> Bool
sensible (d,m) = 1 <= d && d <= 31 && elem m months
```

(c)
```
before :: Date -> Date -> Bool
before (d,m) (d',m') = index m < index m' || index m == index m' && d < d'
```

(d)
```
show2 :: Int -> String
show2 n  =  [intToDigit (n `div` 10), intToDigit (n `mod` 10)]

showDate :: Date -> String
showDate (d,m) | sensible (d,m) = show2 d ++ "/" ++ show2 (index m)
               | otherwise      = "**/**"
```

2. (a)
```
six :: Int -> Int
six n  =  (if n `mod` 2 == 0 then 2 else 1) * (if n `mod` 3 == 0
then 3 else 1)
```
or
```
six :: Int -> Int
six n  =  six' (n `mod` 6)

six' 0 = 6
six' 1 = 1
six' 2 = 2
six' 3 = 3
six' 4 = 2
six' 5 = 1
```
(b)
```
sixes :: [Int] -> [Int]
sixes xs  =  [ six x | x <- xs, x > 0 ]
```
(c)
```
sixes :: [Int] -> [Int]
sixes []  =  []
sixes (x:xs) | x > 0      =  six x : sixes xs
             | otherwise  =  sixes xs
```

3. (a)
```
befores :: [Date] -> Bool
befores xs  =  and [ before x x' | (x,x') <- zip xs (drop 1 xs) ]
```

(b)
```
befores :: [Date] -> Bool
befores []  =  True
befores [x]  =  True
befores (x:y:zs)  =  before x y && befores (y:zs)
```

Part B COMPUTATION AND LOGIC

4. The truth tables and outcomes are as follows:

   (a) This is contingent.

a	b	$E1$ $not(a)$	$E2$ $E1\ or\ b$	$E3$ $b \to a$	$E2 \leftrightarrow E3$
t	t	f	t	t	t
t	f	f	f	t	f
f	t	t	t	f	f
f	f	t	t	t	t

   (b) This is a tautology.

a	b	$E1$ $a \to b$	$E2$ $not(a)$	$E3$ $not(b)$	$E4$ $E3 \to E2$	$E1 \to E4$
t	t	t	f	f	t	t
t	f	f	f	t	f	t
f	t	t	t	f	t	t
f	f	t	t	t	t	t

   (c) This is an inconsistency.

a	b	$E1$ $a \to b$	$E2$ $not(a)$	$E3$ $E2\ or\ b$	$E4$ $not(E1)$	$E4\ and\ E3$
t	t	t	f	t	f	f
t	f	f	f	f	t	f
f	t	t	t	t	f	f
f	f	t	t	t	f	f

5. For this question the answer requires the circuit to be described as a logical expression; then the expression solved in two different ways.

   (a) The expression is:

$$not((not(a\ or\ b)\ or\ c)\ and\ (not(c\ and\ d)\ or\ e))\ or\ e$$

   The truth table is:

a	b	c	d	e	$E1$ $a\ or\ b$	$E2$ $not(E1)$	$E3$ $E2\ or\ c$	$E4$ $c\ and\ d$	$E5$ $not(E4)$	$E6$ $E5\ or\ e$	$E7$ $E3\ and\ E6$	$E8$ $not(E7)$	$E8\ or\ e$
t	t	t	t	t	t	f	t	t	f	t	t	f	t
t	t	t	t	f	t	f	t	t	f	f	f	t	t
t	t	f	t	t	t	f	f	f	t	t	f	t	t
t	t	f	t	f	t	f	f	f	t	t	f	t	t
t	f	t	t	t	t	f	t	t	f	t	t	f	t
t	f	t	t	f	t	f	t	t	f	f	f	t	t
t	f	f	t	t	t	f	f	f	t	t	f	t	t
t	f	f	t	f	t	f	f	f	t	t	f	t	t

   The expression is true (with a and d true) regardless of the truth values of the other inputs. Hence those other inputs are redundant as far as the behaviour of the circuit is concerned.

(b) An appropriate proof is via the following sequence of proof rules (some details of sub-proofs omitted here - the student should supply them):

- To prove $[b, \quad (a \text{ } or \text{ } b) \rightarrow c, \quad c \rightarrow e] \vdash e$:
- *imp_elim* gives $[b, \quad (a \text{ } or \text{ } b) \rightarrow c, \quad c \rightarrow e] \vdash c$
- *imp_elim* gives $[b, \quad (a \text{ } or \text{ } b) \rightarrow c, \quad c \rightarrow e] \vdash (a \text{ } or \text{ } b)$
- *or_intro_right* gives $[b, \quad (a \text{ } or \text{ } b) \rightarrow c, \quad c \rightarrow e] \vdash b$

6. Answers are:

(a) A plausible FSM is as follows, where *ic* is inserting a card; *rs* is requesting a statement; *ds* is dispensing a statement; *rm* is requesting money; *re* is refusing money; *dm* is dispensing money; and *ec* is ejecting a card.

(b) The transition relation is described as the following table:

	ic	rs	ds	rm	re	dm	ec
1	2						
2		5		3			
3					2	4	
4							1
5			2				

(c) This FSM is deterministic because there is no state at which the same action leads to two different states.

7. Answers are:

(a) An appropriate FSM is as follows:

(b) It is NOT possible to draw the FSM for this acceptor because it would need to be able to count arbitrarily large numbers of 0s.

8. Answers are:

(a) $((0|10)*)|((0|10) * 1)$.

(b) $(0(0|1) * 0)|(1(0|1) * 1)$.