THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2018

COMP1600/COMP6260
(Foundations of Computation)

Writing Period: 3 hours duration

Study Period: 15 minutes duration

Permitted Materials: One A4 page with hand-written notes on both sides

Answer ALL questions
Total marks: 100

The questions are followed by labelled blank spaces into which your answers are to be written.

Additional answer panels are provided (at the end of the paper) should you wish to use more space for an answer than is provided in the associated labelled panels. If you use an additional panel, be sure to indicate clearly the question and part to which it is linked.

The following spaces are for use by the examiners.

Q1 (Logic)	Q2 (ND)	Q3 (SI)	Q4 (HL)	Q5 (FSA)	Q6 (CFL)

Q7 (TM)	Total

QUESTION 1 [14 marks] **Logic**

Recall that two formulae are *equivalent* if they have have the same truth values for all variable assignments, and consider the following set of formulae:

- $(p \wedge q) \vee (\neg p \wedge \neg q)$
- $\neg p \vee q$
- $(p \vee \neg q) \wedge (q \vee \neg p)$
- $\neg(p \vee q)$
- $(q \wedge p) \vee \neg p$

(a) Identify two formulae in the above set of formulae that are equivalent, and demonstrate their equivalence by means of a truth table.

QUESTION 1(a) **[4 marks]**

(b) Identify two formulae in the above set of formulae that are *not* equivalent, and demonstrate the fact that they are not equivalent by a variable assignment.

QUESTION 1(b) **[4 marks]**

(c) State whether the following formulae are true or false where x, y and z range over the integers, and justify your answer briefly.

(1). $\forall x \exists y (2x - y) = 0$

(2). $\exists x \forall y (2x - y) = 0$

(3). $\forall x \exists y (x - 2y) = 0$

QUESTION 1(c)	**[6 marks]**

QUESTION 2 [16 marks] **Natural Deduction**

The following questions ask for proofs using natural deduction. Present your proofs in the Fitch style as used in lectures. You may only use the introduction and elimination rules given in Appendix 1. Number each line and include justifications for each step in your proofs.

(a) Give a natural deduction proof of the formula $p \vee (p \rightarrow q)$. In this proof, you may use the law of excluded middle (LEM) $\quad p \vee \neg p$ in addition to the rules provided in the appendix. That is, you may state $p \vee \neg p$ at any line in the proof, where p can stand for an arbitrary formula, and justify this by (LEM).

QUESTION 2(a) **[8 marks]**

(b) Give a natural deduction proof of the rule

$$\frac{\exists x P(x) \land \forall x \forall y (R(x,y) \to P(y))}{\exists x \forall y (R(x,y) \to P(y))}$$

using only the rules in the appendix.

QUESTION 2(b) **[8 marks]**

(a) Consider the function `subl` that computes the list of sub-lists of a given list:

```
subl :: [a] -> [[a]]
subl [] = [[]]                                    -- S1
subl (x:xs) = (subl xs) ++ map (pref x) (subl xs)  -- S2
```

where the functions `map`, `pref` and `++` are given by:

```
map :: (a -> b) -> [a] -> [b]
map f [] = []                         -- M1
map f (x:xs) = (f x):(map f xs)       -- M2

pref :: a -> [a] -> [a]
pref x l = x:l                        -- P

(++) :: [a] -> [a] -> [a]
[]     ++ ys    = ys                  -- A1
(x:xs) ++ ys    = x : (xs ++ ys)      -- A2
```


Show, using structural induction, that

$$\texttt{length (subs l)} = 2^{\texttt{length l}}$$

for all lists `l` of type `a`.

Here, we assume the standard definition of the length function


```
length [] = 0                       -- L1
length (x:xs) = 1 + length xs       -- L2
```

and you may use the fact that `map` preserves `length`, and the fact that length is compatible with concatenation, that is the equations

```
length (map f xs) = length xs            -- LM
length (xs ++ ys) = length xs + length ys -- LA
```

in your proof, with justification as indicated.

In all proofs indicate the justification (eg, the line of a definition used) for each step.

(i) State and prove the base case.

QUESTION 3(a)(i) **[2 marks]**

(ii) State the inductive hypothesis.

QUESTION 3(a)(ii) **[1 mark]**

(iii) State and prove the step case goal.

QUESTION 3(a)(iii) **[5 marks]**

(b) Give an inductive proof the fact that left folding is compatible with list concatenation. Consider the following definition of left folding:

```
foldl :: :: (b -> a -> b) -> b -> [a] -> b
foldl f z [] = z                              -- F1
foldl f z (x:xs) = foldl f (f z x) xs         -- F2
```

and consider a fixed function `f` (of type `b -> a -> b`) and a fixed list `ys` (of elements of type `b`) and show that

$$P(\texttt{xs}) \equiv \forall \texttt{z} \, (\texttt{foldl f z (xs ++ ys) = foldl f (foldl f z xs) ys})$$

holds for all lists `xs` (of elements of type `a`).

(i) State and prove the base case goal

QUESTION 3(b)(i)	**[2 marks]**

(ii) State the inductive hypothesis

QUESTION 3(b)(ii)	**[1 mark]**

(iii) State and prove the step case goal.

QUESTION 3(b)(iii) **[5 marks]**

QUESTION 4 [16 marks] **Hoare Logic**

(a) Specify a precondition P and a postcondition Q such that the Hoare-Triple $\{P\}\ S\ \{Q\}$ holds precisely for all programs S that *never* terminate.

QUESTION 4(a) **[2 marks]**

(b) The following piece of code is called *Rem*

```
r := x;
q := 0;
while (r >=  n)
    r := r - n;
    q := q + 1
```


and computes two numbers, q and r, where

- q is the integer quotient of x by n
- r is the remainder of the division of x by n

We wish to use Hoare Logic (Appendix 3) to show that:

$$\{True\}\ Rem\ \{x = n * q + r\}$$

In the questions below (and your answers), we may refer to the loop code as *Loop*, the body of the loop (i.e. `r:-r-n;q:=q+1`) as *Body*, and the initialisation assignments (i.e. `r:=x;q:=0`) as *Init*.

(i) Given the desired postcondition $\{x = n * q + r\}$, what is a suitable invariant I for *Loop*?

QUESTION 4(b)(i) **[3 marks]**

(ii) Prove that your answer to the previous question is indeed a loop invariant. That is, if we call your invariant I, show that $\{I\}$ *Body* $\{I\}$. Be sure to properly justify each step of your proof.

QUESTION 4(b)(ii)	**[3 marks]**

(iii) Using the previous result and some more proof steps show that

$$\{True\}\ Rem\ \{x = n * q + r\}$$

Be sure to properly justify each step of your proof.

QUESTION 4(b)(iii)	**[2 marks]**

(iv) Explain why the corresponding Hoare-triple for total correctness, that is

$$[True]\,Rem\,[x = n * q + r]$$

is *not* valid by giving a counter-example that shows that the triple above does not hold in general.

QUESTION 4(b)(iv)	**[2 marks]**

(v) Identify a precondition P such that the Hoare triple

$$[P]\,Rem\,[x = n * q + r]$$

is valid. Explain why the Hoare-triple now holds (no formal proof in Hoare Logic required).

QUESTION 4(b)(v)	**[4 marks]**

QUESTION 5 [13 marks]

(a) Design a Finite State Automaton that recognises the language of all strings over the alphabet $\Sigma = \{a, b, c\}$ where both the string 'abc' *and* the string 'cba' occurs as a substring.

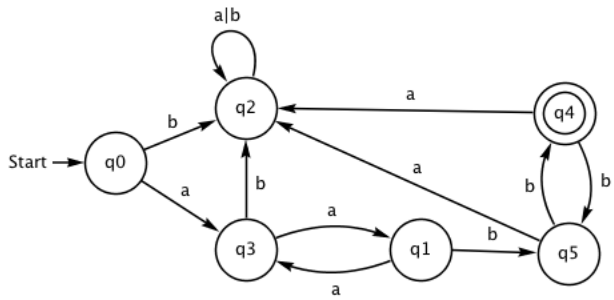
Here, a string s is a substring of a string w if w can be written as $w_1 s w_2$.

QUESTION 5(a) **[3 marks]**

(b) Is your Finite State Automaton (above) deterministic or non-deterministic? Explain.

QUESTION 5(b) **[1 mark]**

(c) What language is recognised by the following Finite State Automaton?



Describe the language in English, and give a regular expression defining the language.

QUESTION 5(c)	**[3 marks]**

(d) Consider the statement

$$\forall\, w \in \Sigma^* .\, N^*(q_2, w) = q_2$$

Express this property in English. Why might it be relevant?

QUESTION 5(d)	**[2 marks]**

(e) For the Finite State Automaton above, prove that

$$\forall n \in \mathbb{N} \,.\, N^*(q_1, (aa)^n) = q_1$$

and hence, or otherwise, conlude that

$$\forall n \in \mathbb{N} \,.\, N^*(q_0, (aa)^{n+1}) = q_1$$

QUESTION 5(e)	**[4 marks]**

QUESTION 6 [13 marks]

(a) Design a push-down automaton that recongnises precisely the language

$$\{a^m b^n \mid n > m > 0\}$$

QUESTION 6(a) **[4 marks]**

(b) Is your automaton deterministic, or non-deterministic? Briefly justify your answer.

QUESTION 6(b) **[1 mark]**

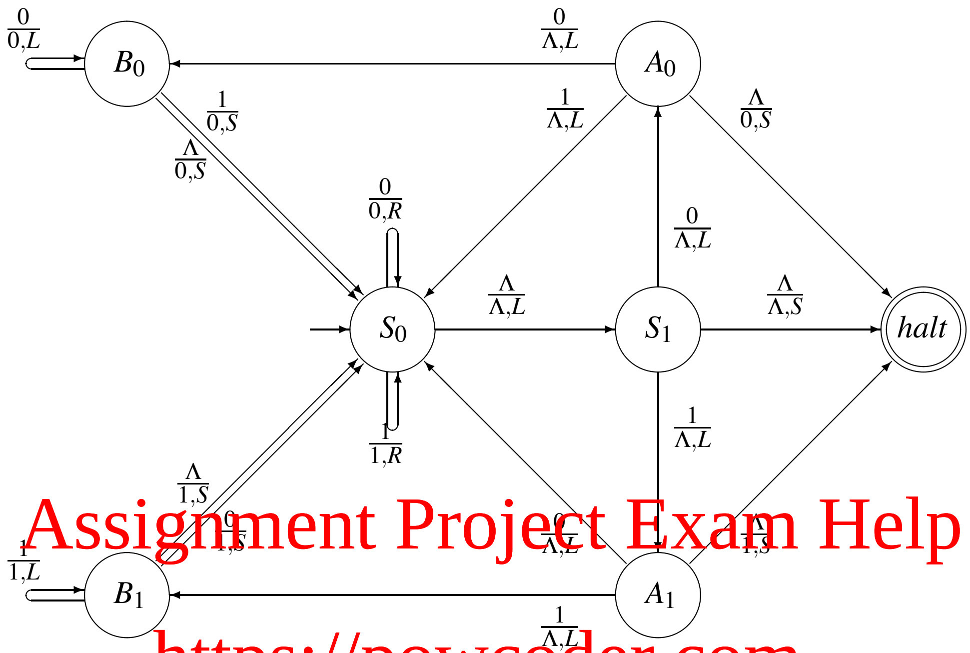
(c) Demonstrate, using the pigeon hole principle or otherwise, that the language given above is *not* regular.

QUESTION 6(c) **[4 marks]**

(d) Give a context-free grammar that generates precisely the language given above.

QUESTION 6(d) **[4 marks]**

(a) The following diagram shows a Turing machine, whose purpose is either to accept or reject the input string. The input string is a string of 0's and 1's and the tape is blank to the left and to the right of the input string. Initially the head is somewhere on the input string.



(i) For each of the strings 010101, 101100010 and 1111000 determine the content of the tape after the machine has terminated with the given string as an input.

QUESTION 7(a)(i)	**[3 marks]**

(ii) Given an input string s, describe the output after the machine has terminated on input string s.

QUESTION 7(a)(ii)	**[3 marks]**

(iii) Explain (no formal proof required) why the machine will always terminate, regardless of the given input string.

QUESTION 7(a)(iii)	**[3 marks]**

(b) Design a Turing Machine that will take a pair of binary numbers, separated by a hash symbol (#), and reverses their order. That is, `0101#1111` should be replaced by `1111#0101` on the tape. Assume that the tape is empty apart from the input and that the tape head is somewhere over the input initially. Include a brief description of the purpose of the individual states.

QUESTION 7(b)	**[3 marks]**

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.

Additional answers. Clearly indicate the corresponding question and part.

Additional answers: deliberately left like this for use in landscape mode. Clearly indicate the corresponding question and part.

Additional answers: deliberately left like this for use in landscape mode. Clearly indicate the corresponding question and part.

Appendix 1 — Natural Deduction Rules

Propositional Calculus

$$(\wedge I)\quad \frac{p \qquad q}{p \wedge q} \qquad\qquad (\wedge E)\quad \frac{p \wedge q}{p} \qquad \frac{p \wedge q}{q}$$

$$(\vee I)\quad \frac{p}{p \vee q} \qquad \frac{p}{q \vee p} \qquad\qquad (\vee E)\quad \frac{p \vee q \qquad \begin{array}{c}[p]\\ \vdots \\ r\end{array} \qquad \begin{array}{c}[q]\\ \vdots \\ r\end{array}}{r}$$

$$(\to I)\quad \frac{\begin{array}{c}[p]\\ \vdots \\ q\end{array}}{p \to q} \qquad\qquad (\to E)\quad \frac{p \qquad p \to q}{q}$$

$$(\neg I)\quad \frac{\begin{array}{c}[p]\\ \vdots \\ F\end{array}}{\neg p} \qquad\qquad (\mathrm{PC})\quad \frac{\begin{array}{c}[\neg p]\\ \vdots \\ F\end{array}}{p}$$

$$(\neg E)\quad \frac{p \qquad \neg p}{F} \qquad\qquad (\mathrm{T})\quad \frac{}{T}$$

Predicate Calculus

$$(\forall I)\quad \frac{P(a) \qquad (a \text{ arbitrary})}{\forall x.\, P(x)} \qquad\qquad (\forall E)\quad \frac{\forall x.\, P(x)}{P(a)}$$

$$(\exists I)\quad \frac{P(a)}{\exists x.\, P(x)} \qquad\qquad (\exists E)\quad \frac{\exists x.\, P(x) \qquad \begin{array}{c}[P(a)]\\ \vdots \\ q\end{array} \qquad (a \text{ arbitrary})}{q \quad (a \text{ is not free in } q)}$$

Appendix 2 — Truth Table Values

p	q	$p \vee q$	$p \wedge q$	$p \rightarrow q$	$\neg p$	$p \Leftrightarrow q$
T	T	T	T	T	F	T
T	F	T	F	F	F	F
F	T	T	F	T	T	F
F	F	F	F	T	T	T

Appendix 3 — Hoare Logic Rules

- Precondition Strengthening:

$$\frac{P_s \;\rightarrow\; P_w \qquad \{P_w\}\; S\; \{Q\}}{\{P_s\}\; S\; \{Q\}}$$

- Postcondition Weakening:

$$\frac{\{P\}\; S\; \{Q_s\} \qquad Q_s \;\rightarrow\; Q_w}{\{P\}\; S\; \{Q_w\}}$$

- Assignment:

$$\{Q(e)\}\; \texttt{x} := \texttt{e}\; \{Q(x)\}$$

- Sequence:

$$\frac{\{P\}\; S_1\; \{Q\} \qquad \{Q\}\; S_2\; \{R\}}{\{P\}\; S_1;\; S_2\; \{R\}}$$

- Conditional:

$$\frac{\{P \wedge b\}\; S_1\; \{Q\} \qquad \{P \wedge \neg\, b\}\; S_2\; \{Q\}}{\{P\}\; \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2\; \{Q\}}$$

- While Loop:

$$\frac{\{P \wedge b\}\; S\; \{P\}}{\{P\}\; \textbf{while } b \textbf{ do } S\; \{P \wedge \neg\, b\}}$$