THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2017

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.

Student Number:

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 [8 marks] **Logic**

(a) Consider the truth value assignment v that assigns the following truth values to the atomic propositions p, q and r: $v(p) = F$, $v(q) = T$, $v(r) = T$.

Which of the following formulae evaluate to T under the assignment v, i.e. when the truth values of p, q and r are given according to v?

(1). $(p \to \neg q) \vee \neg(r \wedge q)$

(2). $(\neg p \vee \neg q) \to (p \vee \neg r)$

(3). $\neg(\neg p \to q) \wedge r$

(4). $\neg(\neg p \to q \wedge \neg r)$

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

(b) Consider the boolean function given by the following truth table:

x	y	z	$f(x,y,z)$
F	F	F	T
F	F	T	F
F	T	F	F
F	T	T	T

x	y	z	$f(x,y,z)$
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

Give a formula (in variables x, y and z) that represents the boolean function given above. Briefly argue *why* the formula indeed represents the boolean function.

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

QUESTION 2 [14 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 $\neg a \wedge \neg b \to \neg(a \vee b)$

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

(b) Give a natural deduction proof of $\forall x(P(x) \to \exists y.P(y))$.

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

(c) Give a natural deduction proof of $\forall z.((\forall x.P(x) \to Q(x)) \wedge \neg Q(z) \to \neg P(z))$.

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

QUESTION 3 [16 marks]

Structural Induction

(a) Give an inductive proof the fact that consecutively mapping two functions over a list is equivalent to mapping their composition over the list. That is:

```
map f (map g xs)  =  map (f.g) xs
```

The definitions of `map` and compose `(.)` are:

```
map f []      = []                                        -- M1
map f (x:xs) = f x : map f xs                             -- M2
(f . g) x    =  f (g x)                                   -- C
```

(i) State and prove the base case goal

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) **[4 marks]**

(b) Consider the following functions defined on lists over an arbitrary type `a`

```
takew :: (a -> Bool) -> [a] -> [a]
takew p [] = []
takew p (x:xs) = if (p x) then x:(takew p xs) else []

dropw :: (a -> Bool) -> [a] -> [a]
dropw p [] = []
dropw p (x:xs) = if p x then (dropw p xs) else (x:xs)
```

together with the append function and the standard equations for `if`

```
(++) :: [a] -> [a] -> [a]
[]     ++ ys    = ys                -- A1     if True  then p else _ = p -- I1
(x:xs) ++ ys    = x : (xs ++ ys)    -- A2     if False then _ else q = q -- I2
```

Show, using structural induction on lists, that the property

```
P(xs) = takew p xs ++ dropw p xs = xs
```

holds for all lists `xs` and all functions `p :: a -> Bool`.

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

(i) State and prove the base case of the proof of `P`.

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

(ii) State the inductive hypotheses of the proof of `P`.

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

(iii) State and prove the step case goal of the proof of `P`.

QUESTION 3(b)(iii) **[6 marks]**

(a) For which programs S does $\{False\}\ S\ \{True\}$ hold?

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

(b) The following piece of code is called *Half*:

```
x := 0;
y := 0;
while (x < a)
    x := x + 2;
    y := y + 1;
```

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

$$\{True\}\ Half\ \{x = 2 * y\}$$

In the questions below (and your answers), we may refer to the loop code as *Loop*, the body of the loop (i.e. x:=x+2;y:=y+1;) as *Body*, and the initialisation assignments (i.e. `x:=0;y:=0;`) as *Init*.

(i) Given the desired postcondition $\{x = 2 * y\}$, what is a suitable invariant for *Loop*? (Hint: notice that the postcondition is independent of the value of a.)

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 P, show that $\{P\}\ \mathit{Body}\ \{P\}$. 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

$$\{\mathit{True}\}\ \mathit{Half}\ \{x = 2 * y\}$$

Be sure to properly justify each step of your proof.

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

(iv) To prove total correctness of the program *Half*, identify and state a suitable *variant* for the loop. Using the same invariant P as above, the variant E should have the following two properties:

- it should be ≥ 0 when the loop is entered, i.e. $P \wedge (x < a) \rightarrow E \geq 0$
- it should decrease every time the loop body is executed, i.e. $[P \wedge (x < a) \wedge E = k]$ `body` $[P \wedge E < k]$

You just need to state the variant, and do not need to prove the two bullet points above (yet).

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

(v) For the variant E you have identified above, give a proof of the premise of the while-rule for total correctness, i.e. give a Hoare-logic proof of $[P \wedge (x < a) \wedge E = k]$ `body` $[P \wedge E < k]$ and argue that $P \wedge (x < a) \rightarrow E \geq 0$.

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

QUESTION 5 [13 marks]

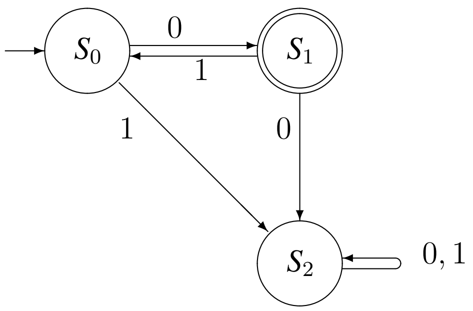
(a) Design a Finite State Automaton that recognises the language of all strings over the alphabet $\Sigma = \{a, b, c\}$ where no a is followed by b, and no b is followed by c, and no c is followed by a.

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

$$\forall w \in \Sigma^* \, . \, N^*(S_2, w) = S_2$$

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

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

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

$$\forall n \in \mathbb{N} \,.\, N^*(S_1, (10)^n) = S_1$$

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

(a) Give a context-free grammar that generates the following language:

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

QUESTION 6(a)	**[3 marks]**

(b) Demonstrate that the following grammar is ambiguous:

$$E \rightarrow 0 \mid 1 \mid E \,\&\, E \mid (E)$$

QUESTION 6(b)	**[3 marks]**

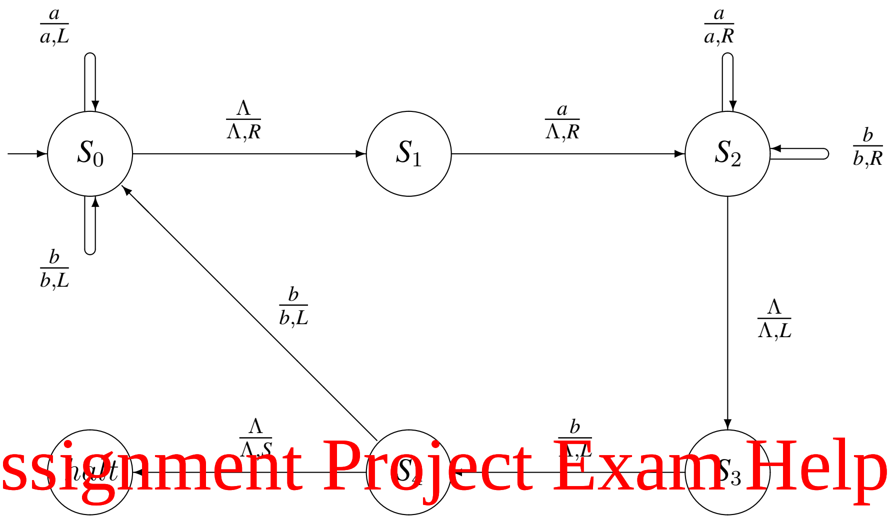
(c) Give a grammar that generates the same language as above, but is *not* ambiguous.

QUESTION 6(c) **[3 marks]**

(d) Demonstrate that the language generated by the grammar above is *not* regular.

QUESTION 6(d) **[3 marks]**

(a) The following diagram shows a Turing machine, whose purpose is either to accept or reject the input string. The input string consists of 'a's and 'b's, and the rest of the tape is blank. (A string accepted if the machine reaches the halt state and rejected if the machine gets stuck in another state.) Initially the head is somewhere on the input string.



(i) Give a general description of the purpose of states S_0 and S_1.

QUESTION 7(a)(i) **[2 marks]**

(ii) What change is accomplished on the tape if the machine moves from state S_1 to state S_4?

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

(iii) What is the language accepted by this machine?

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

(b) Design a Turing Machine which adds an even parity check bit to the right hand end of a bit string. The tape initially contains a non-empty string of binary digits and the read head is somewhere on the string. If there is an *odd* number of 1s in the string, the machine adds another *1* to the right hand end of the string and halts. If there is an *even* number of 1s in the string, the machine adds a *0* to the right hand end of the string and halts. For example:

$$\begin{array}{lcl} 0011000 & \Longrightarrow & 0011000\underline{0} \\ 0011010 & \Longrightarrow & 0011010\underline{1} \\ 0000000 & \Longrightarrow & 0000000\underline{0} \end{array}$$

QUESTION 7(b) **[5 marks]**

(c) Answer the following questions in one sentence.

(i) What is the class of languages recognised by Turing machines?

QUESTION 7(c)(i)	**[2 marks]**

(ii) If a language L is recursively enumerable but not recursive, is the problem P_L of L decidable?

QUESTION 7(c)(ii)	**[2 marks]**

(iii) Are decidable problems easy to solve? If not, give an example that is hard to solve.

QUESTION 7(c)(iii)	**[2 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

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

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

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

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

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

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:

$$\{Q(e)\}\ \mathtt{x} := \mathtt{e}\ \{Q(x)\}$$

- 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:

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