THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2019

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 (PL)	Q1 (FOL)	Q3 (SI)	Q4 (HL)	Q5 (FSA)	Q6 (G)

Q7 (TM)	Total

QUESTION 1 [14 marks] **Propositional Logic**

Consider the following formulae:

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

(a) Identify a formula in the given set of formulae above that is a *contingency*. Describe in one sentence what it means that a formula is a contingency. Hence, or otherwise, establish that the formula you have identified is a contingency.

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

(b) Identify a formula in the given set of formulae above that is a *tautology*. Give a proof of this formula in the calculus of natural deduction (using only the rules provided in Appendix 1, and justifying every inference step). Explain in one sentence why a natural deduction proof of a formula establishes that it is a tautology.

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

(c) Identify a formula in the given set of formulae above that is a *contradiction*. Give a proof of *the negation of this formula* in the calculus of natural deduction (using only the rules provided in Appendix 1, and justifying every inference step). Describe in one sentence what it means that a formula is a contradiction.

QUESTION 1(c) **[5 marks]**

QUESTION 2 [13 marks] **First Order Logic**

A relation $R \subset X \times X$ is called *Euclidean* if it satisfies the formula

$$\forall x \forall y \forall z (R(x,y) \wedge R(x,z) \to R(y,z))$$

and recall that a relation is called *symmetric* if it satisfies

$$\forall x \forall y (R(x,y) \to R(y,x)),$$

reflexive if the formula

$$\forall x (R(x,x))$$

is valid for the relation, and *connected* if

$$\forall x \forall y (R(x,y) \vee R(y,x))$$

is true for R.

(a) Consider the set $X = \mathbb{N} = \{0, 1, 2 \ldots\}$ of natural numbers. Is the relation R, defined by $R(x,y)$ if and only if $x \leq y$, a Euclidean relation? Justify your answer.

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

(b) It is known that every symmetric relation is reflexive. Give a proof of this fact, that is, a proof of the rule

$$\frac{\forall x \forall y (R(x,y) \lor R(y,x))}{\forall x R(x,x)}$$

in natural deduction (using only the rules provided in Appendix 1, and justifying every inference step).

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

(c) It is also known that every relation that is both Euclidean and reflexive is in fact symmetric. Give a proof of this fact, that is, a proof of the rule

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

in natural deduction (using only the rules provided in Appendix 1, and justifying every inference step).

QUESTION 2(c) **[6 marks]**

(a) Consider the following data type of integer-labelled binary trees

```
data Tree = Nul | Node Tree Int Tree
```

and the functions

```
add :: Tree -> Int
add Nul = 0                                        -- A0
add (Node l i r) = i + (add l) + (add r)           -- A1

swap :: Tree -> Tree
swap Nul = Nul                                     -- S0
swap (Node l i r) = Node ((swap r) i (swap l)) -- S1.
```

The aim of this question is to show that swapping subtrees does not affect the sum of the values in the tree, i.e. to prove the formula

$$\forall \mathtt{t}(\mathtt{add\ t = add\ (swap\ t)})$$

where `t` is of type `Tree`, by structural induction.

(i) State and prove the base case goal.

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

(ii) State the inductive hypotheses.

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

(iii) State the step case goal, including all quantifiers, and prove the step case goal.

QUESTION 3(a)(iii)	**[4 marks]**

(b) The evaluation of polynomials can be defined in Haskell using the function `ev` below

```
ev  :: [Int] -> Int -> Int
ev l x = pev l x 0                                   -- E

pev :: [Int] -> Int -> Int -> Int
pev [] x n = 0                                       -- P0
pev (a:as) x n = a * (x ^ n) + (pev as x (n+1))      -- P1
```


where `x^n` is exponentiation in Haskell, i.e. $\mathtt{x\hat{}n} = \mathtt{x}^{\mathtt{n}}$.

Here polynomials are given as a list of coefficients, that is, given $\mathtt{l} = [\mathtt{a}_0, \mathtt{a}_1, \ldots, \mathtt{a}_n]$ the function `ev l x` returns $\sum_{i=0}^{n} \mathtt{a}_i \cdot x^i$.

A different (and more efficient) way to evaluate polynomials is given by the so-called *Horner Scheme* that can be defined as follows:

```
hor :: [Int] -> Int -> Int
hor [] x = 0                                         -- H0
hor (a:as) x = a + x *  (hor as x)                   -- H1
```

The goal of this question is to prove that the Horner scheme indeed delivers the same value as the above sum. The first goal is to show that

$$\forall \mathtt{l} \forall \mathtt{x} \forall \mathtt{n} (\mathtt{pev\ l\ x\ n} = \mathtt{x}^{\mathtt{n}} * (\mathtt{hor\ l\ x})) \qquad (\dagger)$$

by structural induction.

(i) State and prove the base case goal, including all quantifiers, required to prove (†) by structural induction.

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

(ii) State the inductive hypothesis, including all quantifiers.

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

(iii) State the step case goal, including all quantifiers, and prove the step case goal.

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

(iv) Hence, or otherwise, prove that `ev l x = hor l x` for all lists `l` of integers, and for all integers `x`.

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

QUESTION 4 [15 marks] **Hoare Logic**

(a) Consider the following program called *log*, where all variables are natural numbers:

```
while 3*x < a do
  d := d + 1;
  x := x * 3
```

Starting with initial values $d = 0$ and $x = 1$, *log* calculates the integer logarithm of a to base 3, that is, the largest natural number d such that $3^d \leq a$.

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

$$\{d = 0 \ \wedge \ x = 1 \ \wedge \ a > 0\} \ log \ \{3^d \leq a\}$$

(i) Complete the table below by filling in the values of d and x *after* the first, second, etc. iteration of the loop, assuming that the loop executes at least four times. The initial values for d and x are given by the precondition of *log*.

QUESTION 4(a)(i) **[1 mark]**

loop iteration	d	x
0	0	1
1		
2		
3		
4		

(ii) Using the table above, derive an invariant P, that is, a relation between x, d and a, that holds both before the loop is being entered, and after each iteration of the loop body. The invariant needs to have the following three properties:

- it needs to be *strong* enough to imply the postcondition if the loop condition is false:
$$P \wedge \neg(3 * x < a) \rightarrow 3^d \leq a$$
- it needs to be *weak* enough so that it is implied by the precondition:
$$d = 0 \ \wedge \ x = 1 \ \wedge \ a > 0 \rightarrow P$$
- it must be an invariant, i.e
$$\{P \wedge (3 * x < a)\} \ \texttt{d := d+1; x := x*3} \ \{P\}$$
must be provable.

State the invariant, no formal proof is required yet.

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

(iii) Prove that

$$\{d = 0 \ \land \ x = 1 \ \land \ a > 0\} \; \mathit{log} \; \{3^d \leq a\}.$$

Be sure to properly justify each step of your proof.

QUESTION 4(a)(iii) **[3 marks]**

(b) Consider the following (annotated) program called *NpowerM*

$[m \geq 0 \wedge n > 0 \wedge r = 1 \wedge i = 0]$
while `i < m` **do**
```
  r := r * n;
  i := i + 1
```
$[r = n^m]$

The function computes the nth power of m and leaves the result in r.

(i) Complete the table below by filling in the values of r and i *after* the first, second, etc. iteration of the loop, assuming that the loop executes at least four times. The initial values of r and i are given by the precondition of *NpowerM*.

QUESTION 4(b)(i) **[1 mark]**

loop iteration	r	i
0	1	0
1		
2		
3		
4		

(ii) Identify an invariant P for the loop. The invariant needs to have the following three properties:

- it needs to be *strong* enough to imply the postcondition if the loop condition is false:

$$P \wedge \neg(i < m) \to r = n^m$$

- it needs to be *weak* enough so that it is implied by the precondition:

$$m \geq 0 \wedge n > 0 \wedge r = 1 \wedge i = 0 \to P$$

- it must be an invariant, i.e

$$\{P \wedge (i < m)\}\ \texttt{r := r * n; i := i + 1}\ \{P\}$$

must be provable.

State the invariant, no formal proof is required yet.

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

(iii) Identify a variant E for the loop. Using the same invariant P as in the previous exercise, the variant needs to have the following two properties:

- it must be ≥ 0 when the loop is entered, i.e.

$$P \wedge (i < m) \rightarrow E \geq 0$$

- it must decrease every time the loop body is executed, i.e.

$$[P \wedge i < m \wedge E = a]\ \texttt{r := r * n; i := i + 1}\ [P \wedge E < a]$$

State the variant, no formal proof is required yet.

QUESTION 4(b)(iii)	**[1 mark]**

(iv) Give a Hoare-logic (Appendix 3) proof of the total correctness of the while-loop *only*, that is, a proof of the following Hoare triple

$[P]$

while `i < m` **do**
```
   r := r * n;
   i := i + 1
```
$[P \wedge \neg b]$

where `b` is the loop condition `i < m`.

QUESTION 4(b)(iv)	**[5 marks]**

QUESTION 4(b)(iv), continued **[5 marks]**

(a) (i) Design a Finite State Automaton (DFA, NFA, or ϵ-NFA) that recognises the language denoted by the following regular expression:

$$(aa)^*(bb)^*b$$

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

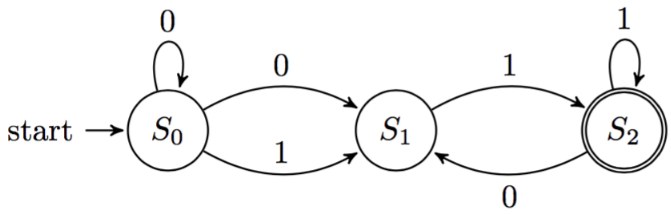
(ii) Describe in English the language that your Finite State Automaton accepts.

QUESTION 5(a)(ii) **[1 mark]**

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

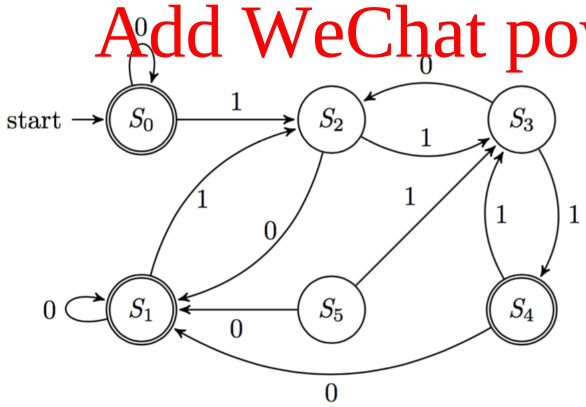
QUESTION 5(b) **[1 mark]**

(c) Use the 'subset construction' algorithm given in lectures to produce a deterministic finite automaton that recognises the same language as the NFA below.



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

(d) Consider the DFA given below as a state transition diagram, and as an equivalent transition table.



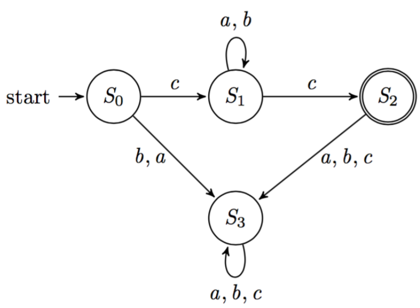
	State	0	1
→⊙	S_0	S_0	S_2
⊙	S_1	S_1	S_2
	S_2	S_1	S_3
	S_3	S_2	S_4
⊙	S_4	S_1	S_3
	S_5	S_1	S_3

(i) Are any of the states of this DFA equivalent to each other? Answer this question using the algorithm given in lectures, and show your working. Explain how you know that the algorithm has terminated, so that you have discovered all possible groups of equivalent states.

(ii) If you discovered any groups of states that are equivalent, use this and the procedure outlined in lectures to give a DFA that recognises the same language but is minimal.

QUESTION 5(d)(ii) **[3 marks]**

(e) Consider the following DFA A:



A recognises precisely the following language

$$L = \{cwc \mid w \in \{a, b\}^*\}.$$

Prove that

$$\forall\, \alpha \in \{a, b\}^*, N^*(S_0, c\alpha c) \in F.$$

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

QUESTION 6 [14 marks] **Grammars**

(a) The following right linear grammar describes the language denoted by the following regular expression $((01 \mid 10)^*11)^*$:

$$
\begin{aligned}
S &\rightarrow\ 0U \mid 1T \mid \epsilon \\
T &\rightarrow\ 1S \mid 0V \\
U &\rightarrow\ 1V \\
V &\rightarrow\ 0W \mid 1T \\
W &\rightarrow\ 1V
\end{aligned}
$$

Use the algorithm presented in lectures to convert this right linear grammar to a NFA. Note that the final state S_f in the algorithm does not play any role in this case, and can be omitted.

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

(b) (i) Design a deterministic push-down automaton that recongnises precisely the following language

$$L = \{0^n 1^m \# \mid n \geq 1, m > n + 2\}.$$

Give either a state transition diagram, or the state transition function δ.

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

(ii) Give a trace to show that your push-down automaton accepts the string $0011111\#$ and another trace to show that it rejects the string $00111\#$.

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

(c) Consider the following Context Free grammar

$$S \to S \otimes S \mid T$$
$$T \to p \mid q$$

over the alphabet $\Sigma = \{\otimes, p, q\}$. Demonstrate that this grammar is ambiguous.

QUESTION 6(c) **[2 marks]**

(d) Give a context-free grammar that generates precisely the language given by the grammar in QUESTION 6(c) (above), but is not ambiguous, and give a parse tree that shows that your grammar generates the string $p \otimes q \otimes p$.

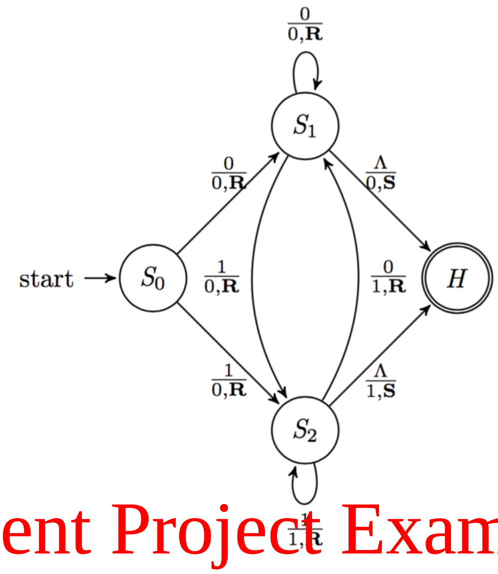
QUESTION 6(d) **[3 marks]**

(e) Following the algorithm presented in lectures, convert the following context-free grammar to a non-deterministic PDA

$$
\begin{aligned}
S &\to \; S \otimes S \;|\; T \\
T &\to \; p \;|\; q
\end{aligned}
$$

QUESTION 6(e) **[2 marks]**

(a) The following diagram shows a Turing machine. 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 at the leftmost character of the input string.



(i) For each of the strings 0101, 10110 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 arbitrary input string s, describe the content of the tape after the machine has terminated in terms of the input string s.

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

(b) Design a Turing Machine that takes a string of the form s`#` where $s \in \{\texttt{a, b}\}^*$ and converts it to uppercase and mirrors it after the hash symbol `#`. For example, `abbaa#` should be replaced by `ABBAA#AABBA` on the tape. Assume that the tape is empty apart from the input and that the tape head is at the hash symbol initially. Include a brief description of the purpose of the individual states.

QUESTION 7(b) **[6 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

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

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

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

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

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

Predicate Calculus

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

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

Appendix 2 — Truth Table Values

p	q	$p \vee q$	$p \wedge q$	$p \to q$	$\neg p$	$p \Leftrightarrow q$
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:

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

- While Loop (Total Correctness):

$$\frac{P \wedge b \rightarrow E \geq 0 \qquad [P \wedge b \wedge E = n]\; S\; [P \wedge E < n]}{[P]\; \textbf{while } b \textbf{ do } S\; [P \wedge \neg\, b]}$$

where n is an auxiliary variable not appearing anywhere else.