THE AUSTRALIAN NATIONAL UNIVERSITY

Second Semester 2015

COMP2600/COMP6260
(Formal Methods for Software Engineering)

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

WITH SOME SAMPLE SOLUTIONS

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.

<table>
<thead>
<tr>
<th>Q1 (StrInd)</th>
<th>Q2 (FOL)</th>
<th>Q3 (NatDed)</th>
<th>Q4 (FSA)</th>
<th>Q5 (CFL)</th>
<th>Q6 (TM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q7 (Hoare)</td>
<td>Q8 (WP)</td>
<td>Q9 (SL)</td>
<td></td>
<td></td>
<td>Total</td>
</tr>
</tbody>
</table>

COMP2600/COMP6260 (Formal Methods for Software Engineering) Page 1 of 29
Consider the following inductively defined type of binary trees

```haskell
data BTree = Emp | Nde BTree BTree
```

together with the following two recursively defined functions that measure the tree height and its number of nodes, respectively.

```haskell
height :: BTree -> Int
height Emp = 1 -- H1
height (Nde l r) = 1 + max (height l) (height r) -- H2

nnodes :: BTree -> Int
nnodes Emp = 1 -- N1
nnodes (Nde l r) = (nnodes l) + (nnodes r) -- N2
```

You are asked to prove that

\[ 1 + \text{nnodes } b \leq 2^{\text{height } b} \]

for all \( b \) of type \( \text{BTree} \).

In all proofs indicate the justification (eg, the line of a definition used) for each step. You may use basic arithmetic (that you can justify with “by arithmetic”).

(a) Using plain English, describe the property \( P \) you are about to prove

The number of nodes in a binary tree is exponentially bounded by its height.
(b) State and prove the base case of the proof of $P$:

**QUESTION 1(b)**

Show that $1 + \text{nodes Emp} \leq 2^{\text{height Emp}}$.

Proof.

\[
1 + \text{nodes Emp} \\
= 1 + 1 \\
= 2^1 \\
= 2^{\text{height Emp}} \\
\leq 2^{\text{height Emp}}
\]

(c) State the inductive hypotheses of the proof of $P$.

**QUESTION 1(c)**

\[
1 + \text{nodes } l \leq 2^{\text{height } l} \quad \text{-- IH}_l \\
1 + \text{nodes } r \leq 2^{\text{height } r} \quad \text{-- IH}_r
\]
(d) State and prove the step case goal of the proof of $P$.

**QUESTION 1(d)**

If both inductive hypotheses hold, then

$$1 + \text{nodes}(\text{Nde } l \ r) \leq 2^{\text{height } \text{Nde } l \ r}.$$ 

Proof.

$$1 + \text{nodes}(\text{Nde } l \ r)$$
$$= 1 + 1 + (\text{nodes } l) + (\text{nodes } r) \quad \text{-- N2}$$
$$\leq 2^{\text{height } l} + 2^{\text{height } r} \quad \text{-- IH}_l, \text{IH}_r$$
$$\leq 2 \times 2^{\max(\text{height } l)(\text{height } r)} \quad \text{-- arith}$$
$$= 2^{1 + \max(\text{height } l)(\text{height } r)} \quad \text{-- arith}$$
$$= 2^{\text{height } (\text{Nde } l \ r)} \quad \text{-- H2}$$
**QUESTION 2 [6 marks]**

(Propositional Logic)

For each of the formulae below, give a valuation of the two propositional variables $p$ and $q$ under which the given formula evaluates to true, and a valuation under which it evaluates to false.

1. $(p \land q) \rightarrow \neg p \land \neg q$
2. $\neg(p \rightarrow (p \rightarrow q))$
3. $\neg(p \lor q) \leftrightarrow (\neg p \land q)$

<table>
<thead>
<tr>
<th>QUESTION 2</th>
<th>[6 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1). true for $p = q = f$ and false for $p = q = t$</td>
<td></td>
</tr>
<tr>
<td>(2). true for $p = t, q = f$ and false for $p = q = f$</td>
<td></td>
</tr>
<tr>
<td>(3). true for $p = q = t$ and false for $p = q = f$.</td>
<td></td>
</tr>
</tbody>
</table>
QUESTION 3 [12 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 \( \neg a \land b \rightarrow \neg (a \lor \neg b) \).
(b) Give a natural deduction proof of

\[(\exists x. (\neg P(x))) \rightarrow ((\forall x. P(x)) \rightarrow D(z))\]

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>(\exists x. \neg P(x))</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>(\neg P(a))</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>(\forall x. P(x))</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>(\neg D(z))</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>(P(a))</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>(\forall x. P(x))</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>(D(z))</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>(\neg P(a))</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>(\exists x. \neg P(x))</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>((\forall x. P(x)) \rightarrow D(z)))</td>
</tr>
</tbody>
</table>

- **Line 1:** \(\exists x. \neg P(x)\)
- **Line 2:** \(\neg P(a)\), assuming \(a\)
- **Line 3:** \(\forall x. P(x)\), from \(\exists x. \neg P(x)\) and \(\neg P(a)\)
- **Line 4:** \(\neg D(z)\), \(\forall E\) with \(\forall x. P(x)\)
- **Line 5:** \(P(a)\), from \(\neg D(z)\) and \(\forall x. P(x)\)
- **Line 6:** \(\neg P(a)\), \(\neg I\) with \(P(a)\) and \(\neg D(z)\)
- **Line 7:** \(D(z)\), \(\wedge I\) with \(\neg P(a)\), \(\neg D(z)\), and \(\forall x. P(x)\)
- **Line 8:** \(\forall x. P(x)\) \(\rightarrow D(z)\), \(\rightarrow I\) with \(\forall x. P(x)\) and \(D(z)\)
- **Line 9:** \(\exists x. \neg P(x)\), \(\exists E\) with \(\neg P(a)\)
- **Line 10:** \((\forall x. P(x)) \rightarrow D(z))\), \(\rightarrow E\) with \(\forall x. P(x)\) and \(D(z)\)
(a) The following is a right linear grammar where $S$ is the start symbol:

$$
S \rightarrow \epsilon \\
S \rightarrow a \\
S \rightarrow b \\
S \rightarrow aP \\
S \rightarrow bR \\
P \rightarrow aQ \\
R \rightarrow bQ \\
Q \rightarrow \epsilon \\
Q \rightarrow aP \\
Q \rightarrow bR
$$

Convert this grammar to a non-deterministic finite automaton (NFA).
(b) The following is a non-deterministic finite automaton (NFA):

Convert this NFA to a deterministic finite automaton (DFA).

QUESTION 4(b) [3 marks]
The following is a deterministic finite automaton (DFA):

Minimise this DFA. Show all the steps of your work.

We first detect equivalent states by running the algorithm as below:

- Initial split: $[[S_0, S_1, S_2, S_3], [S_4, S_5]]$
- Testing with $a$: $[[S_0, S_1], [S_2, S_3], [S_4, S_5]]$
- Testing with $a$ or $b$: $[[S_0], [S_1], [S_2, S_3], [S_4, S_5]]$

These tests produce no further splits. Therefore $S_2, S_3$ are equivalent, and $S_4, S_5$ are equivalent.

The state $S_1$ is inaccessible, so it is deleted. The minimised DFA is as below:

COMP2600/COMP6260 (Formal Methods for Software Engineering) Page 11 of 29
(d) Let $L$ be the language of strings over $\{a, b\}$ that contain twice as many $a$’s as $b$’s. Prove that no finite state automaton can recognise $L$.

**QUESTION 4(d)**

Assume for a contradiction that such an automaton exists. Let the automaton be $A$, and let its start state be $S_0$, and its set of final states be $F$.

There infinitely many strings of the form $b^n$, but only finitely many states in $A$. Therefore, by the pigeonhole principle, there exists natural numbers $j$ and $k$, $j \neq k$ such that:

$$N^*(S_0, b^j) = N^*(S_0, b^k).$$

Consider the string $b^j a^{2k}$. This string is in $L$, because it has twice as many $a$’s as $b$’s. Since $A$ recognises $L$,

$$N^*(S_0, b^j a^{2k}) \in F.$$ 

Consider the string $b^j a^{2k}$. This string is not in $L$, thus $A$ must reject this string. Formally,

$$N^*(S_0, b^j a^{2k}) \notin F.$$ 

But by the append theorem,

$$N^*(S_0, b^j a^{2k}) = N^*(N^*(S_0, b^k), a^{2k}) = N^*(N^*(S_0, b^j), a^{2k}) = N^*(S_0, b^j a^{2k}).$$

Therefore this state must be both a final state and not a final state. Contradiction.

Such an automaton cannot exist.
QUESTION 5  [12 marks]  Context Free Languages and Pushdown Automata

(a) Consider the following context free grammar, where $S$ is the start symbol.

$$
S \rightarrow S! \mid S \circ S \mid S \diamond S \mid (S) \mid U
$$

$$
U \rightarrow x \mid y
$$

Give two parse trees for some string to show that the above grammar is ambiguous.

(b) Give an unambiguous grammar that generates the same language as the grammar given in sub question (a).

We assume that the precedence of logical connectives is as follows: $!$ binds tighter than $\diamond$, which in turn binds tighter than $\circ$. 

$$
S \rightarrow S \circ D \mid D
$$

$$
D \rightarrow D \diamond B \mid B
$$

$$
B \rightarrow B! \mid U
$$

$$
U \rightarrow x \mid y \mid (S)
$$
(c) Convert the context free grammar given in sub question (a) to a (non-deterministic) PDA.

**QUESTION 5(c)**

**Initialisation:** \( \delta(q_0, \epsilon, Z) \mapsto q_1/SZ \)

**Non-terminals:**
- \( \delta(q_1, \epsilon, S) \mapsto q_1/S \circ S \)
- \( \delta(q_1, \epsilon, S) \mapsto q_1/S \diamond S \)
- \( \delta(q_1, \epsilon, S) \mapsto q_1/S! \)
- \( \delta(q_1, \epsilon, S) \mapsto q_1/(S) \)
- \( \delta(q_1, \epsilon, U) \mapsto q_1/U \)
- \( \delta(q_1, \epsilon, U) \mapsto q_1/U \)

**Terminals:**
- \( \delta(q_1, a, a) \mapsto q_1/\epsilon \)

**Termination:** \( \delta(q_1, \epsilon, Z) \mapsto q_2/\epsilon \)

Where \( a \in \{\circ, \diamond, !, x, y\} \).

(d) Consider the following PDA defined by its transitions:

\[
\begin{align*}
\delta(q_0, \epsilon, Z) & \mapsto q_1/abZ \\
\delta(q_1, a, a) & \mapsto q_1/\epsilon \\
\delta(q_1, b, b) & \mapsto q_2/\epsilon
\end{align*}
\]

where the initial state is \( q_0 \). Write a trace for this PDA when processing the string \( ab \). State explicitly whether the string is accepted or rejected.

**QUESTION 5(d)**

\[
\begin{align*}
(q_0, ab, Z) & \Rightarrow (q_1, ab, abZ) \\
& \Rightarrow (q_1, b, bZ) \\
& \Rightarrow (q_2, \epsilon, Z) \\
\text{reject}
\end{align*}
\]
**QUESTION 6** [12 marks]

(a) Assume the input alphabet is \{a, b\}. Design a Turing machine that replaces every second \(a\) in the input with \(d\) and replaces every other \(a\) in the input with \(c\). For example, it should transform ‘aba’ into ‘cbd’. Assume that the pointer head points at the beginning of the input initially. You only need to draw a diagram of states and transitions for your Turing machine.

![Diagram of Turing machine](image)

Answer the following questions in one sentence.

(b) If a language \(L\) is recursive, what does that mean for the corresponding decision problem \(P_L\)?

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

\(P_L\) is decidable.

(c) What kind of algorithm exists if a problem \(P_L\) is partially decidable?

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

An algorithm which will terminate and accept if the input in is in \(L\), but may nor terminate if the input is not in \(L\).
(d) Do non-deterministic Turing Machines have extra power over deterministic Turing Machines?

No.
QUESTION 7  [12 marks]  

**Hoare Logic**

The following sequence is an example of a geometric sequence: 1, 2, 4, 8, 16, …

Let \( n > 1 \) be some fixed natural number. The sum \( S_n \) of the first \( n \) terms of the geometric sequence is given by \( S_n = 2^n - 1 \)

Consider the following code fragment where \( 2^i \) computes \( 2^i \):

\[
\begin{align*}
  &i := 1; \\
  &s := 1; \\
  \text{while } (i \neq n) \text{ do } \\
  &s := s + 2^i; \\
  &i := i + 1
\end{align*}
\]

\( \text{Init} \) \hspace{1cm} \text{Body} \hspace{1cm} \text{Loop} \hspace{1cm} \text{SumGeometric} \)

Let’s just plug in a few values to see how it works for some \( n > 1 \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>…</td>
</tr>
<tr>
<td>( s )</td>
<td>1 + 2</td>
<td>1 + 2 + 4</td>
<td>1 + 2 + 4 + 8</td>
<td>…</td>
</tr>
<tr>
<td>( 2^n - 1 )</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>…</td>
</tr>
</tbody>
</table>

Our task is to use the rules of **Hoare Logic** (in the handouts) to show that the value of \( s \), if and when this code terminates, will be the sum \( S_n \) of the first \( n \) numbers \( 1, 2, 4, \ldots, 2^n-1 \) in the geometric sequence.

That is, we will try to prove

\[ \{ \text{True} \} \text{ SumGeometric} \{ \text{Post} \} \]

where

\[ \text{Post} \equiv ( s = (2^n - 1) ) \]

The proof will have this structure where \( \text{Inv} \) is the loop-invariant:

\[
\begin{align*}
  &\vdots \\
  &\{ (i \neq n) \land \text{Inv} \} \text{ Body} \{ \text{Inv} \} \\
  &\{ \text{Inv} \} \text{ while } (i \neq n) \text{ do } \text{Body} \{ \text{Inv} \land \neg (i \neq n) \} \\
  &\vdots \\
  &\{ \text{True} \} \text{ Init} \{ \text{Inv} \} \\
  &\{ \text{Inv} \} \text{ while } (i \neq n) \text{ do } \text{Body} \{ \text{Post} \} \cdots \\
  &\{ \text{True} \} \text{ SumGeometric} \{ \text{Post} \} \\
\end{align*}
\]

While

You may assume that all variables are typed integer. In the questions below we will refer to the code fragment as SumGeometric, to the loop code as \( \text{Loop} \), and to the body of the loop as \( \text{Body} \). **Make sure that every step of your proof is numbered, and is justified by citing the rule, and any previous proof steps, that you are using.**
(a) The following algebraic equivalence will be useful in both this question, and the next question on Weakest Precondition Calculus. You may use it at any time, regardless of whether you successfully answer this question, so long as you explicitly refer to it as Lemma (a).

Prove, using standard algebraic manipulations, that
\[(s + 2^i = 2^{(i+1)} - 1) \iff (s = 2^i - 1)\]

Lemma (a)

<table>
<thead>
<tr>
<th>QUESTION 7(a)</th>
<th>[2 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[s + 2^i = 2^{(i+1)} - 1]</td>
<td>[\iff s + 2^i = 2 \cdot 2^i - 1]</td>
</tr>
<tr>
<td>[\iff s = 2 \cdot 2^i - 1 - 2^i]</td>
<td>[= 2^i(2 - 1) - 1]</td>
</tr>
<tr>
<td>[= 2^i - 1]</td>
<td></td>
</tr>
</tbody>
</table>

(b) We will need an invariant for Loop. We suggest
\[Inv \equiv (s = 2^i - 1)\].

Prove that \(\{Inv\} Body \{Inv\}\).

<table>
<thead>
<tr>
<th>QUESTION 7(b)</th>
<th>[4 marks]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1). ({s + 2^i = 2^{i+1} - 1} s := s + 2^i {s = 2^{i+1} - 1})</td>
<td>(Asst.)</td>
</tr>
<tr>
<td>(2). (s + 2^i = 2^{i+1} - 1) (\iff s = 2^i - 1)</td>
<td>(Lemma (a))</td>
</tr>
<tr>
<td>(3). ({s = 2^i - 1} s := s + 2^i {s = 2^{i+1} - 1})</td>
<td>(PreConEq)</td>
</tr>
<tr>
<td>(4). ({s = 2^{i+1} - 1} i := i + 1 {s = 2^i - 1})</td>
<td>(Asst.)</td>
</tr>
<tr>
<td>(5). ({s = 2^i - 1} s := s + 2^i ; i := i + 1 {s = 2^i - 1})</td>
<td>((3),(4), Seq.)</td>
</tr>
<tr>
<td>(6). ({Inv} Body {Inv})</td>
<td>(i.e.)</td>
</tr>
</tbody>
</table>
(e) Using part (b), prove that

\{Inv\} Loop \{Post\}.

QUESTION 7(c) [3 marks]

(1). \( (s = 2^i - 1 \wedge (i \neq n)) \Rightarrow (s = 2^i - 1) \) (Basic Logic)
(2). \{s = 2^i - 1 \wedge (i \neq n)\} Body \{s = 2^i - 1\} (b),(1), PreConStr
(3). \{s = 2^i - 1\} Loop \{(s = 2^i - 1) \wedge \neg (i \neq n)\} ((2), While)
(4). \((s = 2^i - 1) \wedge \neg (i \neq n)\) \Rightarrow (s = 2^n - 1) (Basic Maths)
(5). \{s = 2^i - 1\} Loop \{s = (2^n - 1)\} ((3),(4), PostConWk.)
(6). \{Inv\} Loop \{Post\} (i.e.)

(d) Using part (c), prove that

\{True\} SumGeometric \{Post\}.

QUESTION 7(d) [3 marks]

(1). \{1 = 2^1 - 1\} i := 1 \{1 = 2^i - 1\} (Asst)
(2). \{1 = 2^i - 1\} \Leftrightarrow True (Basic Arith)
(3). \{True\} i := 1 \{1 = 2^i - 1\} ((1),(2), PreConEq)
(4). \{1 = 2^i - 1\} s := 1 \{s = 2^i - 1\} (Asst)
(5). \{True\} i := 1; s := 1 \{s = 2^i - 1\} ((3), (4) Seq)
(6). \{True\} i := 1; s := 1; Loop \{s = (2^n - 1)\} ((5), c, Seq)
(7). \{True\} SumGeometric \{Post\} (i.e.)
QUESTION 8 [12 marks]

Weakest Precondition Calculus

As with the previous question, we will consider the code fragment `SumGeometric`:

```plaintext
i := 1;
s := 1;
while (i ≠ n) do
  s := s + exp(2, i)
i := i + 1;
```

We will use the rules of the Weakest Precondition Calculus (Appendix 4) to calculate

\[ wp(SumGeometric, s = (2^n - 1)). \]

As in the previous question we will use the abbreviations Loop and Body for the indicated parts of the code. You may continue to make use of the lemma from part (a) of the previous question by referring to it explicitly as Lemma (a). Remember show all your working when you do so.

(a) We will want to calculate

\[ wp(\text{Loop}, s = (2^n - 1)). \]

\( P_0 \) is the predicate expressing success for this weakest precondition after zero loop iterations. Using the definitions in the appendix for the definition of \( P_0 \), show that

\[ P_0 \equiv (i + 0 = n) \land (s = 2^i - 1) \]

QUESTION 8(a) [1 mark]

\[
\begin{align*}
P_0 &\equiv \neg b \land Q \\
P_0 &\equiv \neg (i \neq n) \land s = (2^n - 1) \\
&\equiv (i = n) \land (s = 2^i - 1) \\
&\equiv (i + 0 = n) \land (s = 2^i - 1)
\end{align*}
\]
(b) $P_1$ is the predicate expressing success after one loop iteration. Using the definition in the appendix for $P_1$, show that $P_1 \equiv (i + 1 = n) \land (s = 2^i - 1)$

**QUESTION 8(b) [3 marks]**

$P_1 \equiv (i \neq n) \land wp(Body, P_0)$

$\equiv (i \neq n) \land wp(s := s + \exp(2, i); \ i := i + 1, P_0)$

$\equiv (i \neq n) \land wp(s := s + \exp(2, i), \ wp(i := i + 1, P_0))$

$\equiv (i \neq n) \land wp(s := s + \exp(2, i), \ wp(\ i := i + 1, (i + 0 = n) \land (s = 2^i - 1)))$

$\equiv (i \neq n) \land (i + 1 = n) \land (s + 2^i = 2^{i+1} - 1)$

$\equiv (i \neq n) \land (i + 1 = n) \land (s = 2^i - 1)$  \quad \text{Lemma (a)}$

$\equiv (i + 1 = n) \land (s = 2^i - 1)$
(c) \( P_k \) is the predicate expressing success after \( k \) loop iterations that holds for all \( k \geq 0 \). Using induction, prove that

\[ \forall k \geq 0. \ P_k \equiv (i + k = n) \land (s = 2^i - 1) \]

Ensure that you state the Base Case, Inductive Hypothesis and Inductive Step correctly.

**QUESTION 8(c)** [5 marks]

**Base Case:** We already have \( P_0 \equiv (i + 0 = n) \land (s = 2^0 - 1) \)

**I.H.:** Assume for some \( j > 0 \) that \( \forall 0 \leq k \leq j. \ P_k \equiv (i + k = n) \land (s = 2^i - 1) \)

**Inductive Step:** We have to show that \( P_{j+1} \equiv (i + (j + 1) = n) \land (s = 2^i - 1) \)

\[
P_{j+1} \equiv (i \neq n) \land wp(Body, P_j)
\equiv (i \neq n) \land wp(s := s + \exp(2, i); i := i + 1, P_j)
\equiv (i \neq n) \land wp(s := s + \exp(2, i), wp(i := i + 1, P_j))
\equiv (i \neq n) \land wp(s := s + \exp(2, i), wp(i := i + 1, (i + j = n) \land (s = 2^i - 1)))
\equiv (i \neq n) \land wp(s := s + \exp(2, i), (i + 1 + j = n) \land (s = 2^{i+1} - 1))
\equiv (i \neq n) \land (i + 1 + j = n) \land (s + 2^i = 2^{i+1} - 1)
\equiv (i \neq n) \land (i + 1 + j = n) \land (s = 2^i - 1) \quad \text{Lemma (a)}
\equiv (i + (j + 1) = n) \land (s = 2^i - 1)
\]

(d) Given your answer to part (c), state

\[ wp(\text{Loop}, s = (2^n - 1)). \]

Do not attempt any simplification at this stage.

**QUESTION 8(d)** [1 mark]

\[
\exists k. \ k \geq 0 \land P_k
\equiv \exists k. \ k \geq 0 \land (i + k = n) \land (s = 2^i - 1)
\]
(e) Hence calculate

\[ \text{wp}(\text{SumGeometric}, s = (2^n - 1)). \]

and state this result in the simplest form possible. Stating an answer without showing your working out is not sufficient.

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

\[
\begin{align*}
\text{wp}(\text{SumGeometric}, s = (2^n - 1)) \\
\equiv \text{wp}(i := 1, \text{wp}(s := 1, \text{wp}(\text{Loop}, s = (2^n - 1)))) \\
\equiv \text{wp}(i := 1, \text{wp}(s := 1, \exists k. k \geq 0 \land (i + k = n) \land (s = 2^i - 1))) \\
\equiv \text{wp}(i := 1, \exists k. k \geq 0 \land (i + k = n) \land (1 = 2^i - 1)) \\
\equiv \exists k. k \geq 0 \land (1 + k = n) \land (1 = 2^i - 1) \\
\equiv \exists k. k \geq 0 \land k + 1 = n \land \text{True} \\
\equiv n \geq 1
\end{align*}
\]
QUESTION 9 [10 marks] Separation Logic.

Let Π be the following program:

\[\begin{align*}
\text{x := cons(1,2)} \\
\text{dispose(x + 1)}
\end{align*}\]

Our goal is to prove

\[\{\text{emp}\} \; \Pi \; \{x \mapsto 1\}\]

But we are going to do it bit by bit.

You may need the following properties of separation logic:

Commutativity: \((A * B) \iff (B * A)\)

Abbreviation: \((x \mapsto e_1, e_2) \equiv (x \mapsto e_1 * (x + 1) \mapsto e_2)\)

Unit: \((\text{emp} * A) \iff A\)

Given: \(((x + 1) \mapsto 2) \Rightarrow ((x + 1) \mapsto -)\)

(a) Prove the triple below:

\[\{\text{emp}\} \; x := \text{cons}(1,2) \; \{x \mapsto 1 * (x + 1) \mapsto 2\}\]

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. Do *not* use the frame rule. My solution is three lines long but any proper proof without the frame rule will do.

QUESTION 9(a) [2 marks]

(1). \{\text{emp}\} \; x := \text{cons}(1,2) \; \{x \mapsto 1,2\} \quad \text{(DerAllAssAxm)}

(2). \; x \mapsto 1,2 \equiv x \mapsto 1 * (x + 1) \mapsto 2 

\quad \text{(Abbrev)}

(3). \{\text{emp}\} \; x := \text{cons}(1,2) \; \{x \mapsto 1 * (x + 1) \mapsto 2\} 

\quad \text{((1), (2) PostConEq)}
(b) Prove \( \{ (x + 1) \mapsto 2 \} \) dispose \( x + 1 \) \{ emp \}

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. Do *not* use the frame rule. My solution is three lines long but any proper proof will do.

**QUESTION 9(b) [3 marks]**

1. \( \{ (x + 1) \mapsto \} \) dispose \( x + 1 \) \{ emp \} \hspace{1cm} \text{(DispAxm)}
2. \( \{ (x + 1) \mapsto 2 \} \Rightarrow \{ (x + 1) \mapsto \} \) \hspace{1cm} \text{(Given)}
3. \( \{ (x + 1) \mapsto 2 \} \) dispose \( x + 1 \) \{ emp \} \hspace{1cm} \text{(PreConStr)}

---

(c) Using (a) and (b) prove:

\( \{ \text{emp} \} \ x := \text{cons}(1, 2) \ ; \ \text{dispose}(x + 1) \{ x \mapsto 1 \} \)

Set out the proof as a linear sequence of numbered lines with justifications on the right hand side as usual. Please use first line (a) and second line (b). My solution then has six more lines, but any proper proof will do. You will need the frame rule.

**QUESTION 9(c) [5 marks]**

1. \( \{ \text{emp} \} \ x := \text{cons}(1, 2) \{ x \mapsto 1 \ast (x + 1) \mapsto 2 \} \) \hspace{1cm} \text{(a)}
2. \( \{ (x + 1) \mapsto 2 \} \) dispose \( x + 1 \) \{ emp \} \hspace{1cm} \text{(b)}
3. \( \{ (x + 1) \mapsto 2 \ast x \mapsto 1 \} \) dispose \( x + 1 \) \{ emp \ast x \mapsto 1 \} \hspace{1cm} \text{(2) FrameRule}
4. \( (A \ast B) \Leftrightarrow (B \ast A) \) \hspace{1cm} \text{(Commutativity)}
5. \( \{ x \mapsto 1 \ast (x + 1) \mapsto 2 \} \) dispose \( x + 1 \) \{ \text{emp} \ast x \mapsto 1 \} \hspace{1cm} \text{(3), (4) PreConEqv}
6. \( (\text{emp} \ast A) \Leftrightarrow A \) \hspace{1cm} \text{(Unit)}
7. \( \{ x \mapsto 1 \ast (x + 1) \mapsto 2 \} \) dispose \( x + 1 \) \{ x \mapsto 1 \} \hspace{1cm} \text{(5), (6) PostConEqv}
8. \( \{ \text{emp} \} \ x := \text{cons}(1, 2) \ ; \ \text{dispose}(x + 1) \{ x \mapsto 1 \} \) \hspace{1cm} \text{((1), (7) Seq)}
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.
This material to be given to the students as a separate handout, not an appendix to the exam paper!

Handout 1 — Natural Deduction Rules

Propositional Calculus

\[(\land I)\] \[\frac{p \quad q}{p \land q}\]

\[(\land E)\] \[\frac{p \land q}{p} \quad \frac{p \land q}{q}\]

\[(\lor I)\] \[\frac{p}{p \lor q} \quad \frac{p}{q \lor p}\]

\[(\lor E)\] \[\frac{p \lor q \quad r}{r} \quad \frac{p \lor q \quad r}{r}\]

\[([p] \quad [q] \quad \vdots \quad \vdots)\]

\[(\Rightarrow I)\] \[\frac{q}{p \Rightarrow q}\]

\[(\Rightarrow E)\] \[\frac{p \quad p \Rightarrow q}{q}\]

\[([p] \quad [\neg p] \quad \vdots \quad \vdots)\]

\[([\neg I])\] \[\frac{q \land \neg q}{\neg p}\]

\[([\neg E])\] \[\frac{q \land \neg q}{p}\]

Predicate Calculus

\[(\forall I)\] \[\frac{P(a) \quad (a \text{ arbitrary})}{\forall x. P(x)}\]

\[(\forall E)\] \[\frac{\forall x. P(x)}{P(a)}\]

\[([P(a)] \quad \vdots \quad \vdots)\]

\[(\exists I)\] \[\frac{P(a)}{\exists x. P(x)}\]

\[(\exists E)\] \[\frac{\exists x. P(x) \quad q \quad (a \text{ arbitrary})}{q \quad (a \text{ is not free in } q)}\]
### Handout 2 — Truth Table Values

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>$q$</td>
<td>$p \lor q$</td>
<td>$p \land q$</td>
<td>$p \Rightarrow q$</td>
<td>$\neg p$</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Handout 3 — Hoare Logic Rules

- **Precondition Strengthening:**
  \[
  \begin{align*}
  P_s & \Rightarrow P_w & \{P_w\} S \{Q\} \\
  \{P_s\} S \{Q\} & \end{align*}
  \]

- **Postcondition Weakening:**
  \[
  \begin{align*}
  \{P\} & S \{Q_s\} & Q_s & \Rightarrow Q_w \\
  \{P\} S \{Q_w\} & \end{align*}
  \]

- **Assignment:**
  \[
  \{Q(e)\} x := e \{Q(x)\}
  \]

- **Sequence:**
  \[
  \begin{align*}
  \{P\} S_1 \{Q\} & \{Q\} S_2 \{R\} \\
  \{P\} S_1; S_2 \{R\} & \end{align*}
  \]

- **Conditional:**
  \[
  \begin{align*}
  \{P \land b\} & S_1 \{Q\} & \{P \land \neg b\} S_2 \{Q\} \\
  \{P\} & \text{if } b \text{ then } S_1 \text{ else } S_2 \{Q\} & \end{align*}
  \]

- **While Loop:**
  \[
  \begin{align*}
  \{P \land b\} & S \{P\} \\
  \{P\} & \text{while } b \text{ do } S \{P \land \neg b\} & \end{align*}
  \]
Handout 4 — Weakest Precondition Rules

\[ wp(x := e, Q(x)) \equiv Q(e) \]
\[ wp(S_1; S_2, Q) \equiv wp(S_1, wp(S_2, Q)) \]
\[ wp(\text{if } b \text{ then } S_1 \text{ else } S_2, Q) \equiv (b \Rightarrow wp(S_1, Q)) \land (\neg b \Rightarrow wp(S_2, Q)) \]
\[ \equiv (b \land wp(S_1, Q)) \lor (\neg b \land wp(S_2, Q)) \]

\( P_k \) is the weakest predicate that must be true before while \( b \) do \( S \) executes, in order for the loop to terminate after exactly \( k \) iterations in a state that satisfies \( Q \).

\[ P_0 \equiv \neg b \land Q \]
\[ P_{k+1} \equiv b \land wp(S, P_k) \]
\[ wp(\text{while } b \text{ do } S, Q) \equiv \exists k. (k \geq 0 \land P_k) \]
Handout 5: Separation Logic Rules

Floyd Store Axiom for Separation Logic: replaces Hoare (Store) Axiom
\[ \{ x = v \land \text{emp} \} \ x := e \ \{ x = e(v/x) \land \text{emp} \} \]
where \( v \) is an auxiliary variable which does not occur in \( e \)

Derived Floyd Store Axiom for Separation Logic:
\[ \{ \text{emp} \} \ x := e \ \{ x = e \land \text{emp} \} \]
where \( x \) does not occur in \( e \)

Fetch Assignment Axiom
\[ \{ (x = v_1) \land (e \mapsto v_2) \} \ x := [e] \ \{ (x = v_2) \land (e(v_1/x) \mapsto v_2) \} \]
where \( v_1 \) and \( v_2 \) are auxiliary variables which do not occur in \( e \)

Derived Fetch Assignment Axiom
\[ \{ (e \mapsto v_2) \} \ x := [e] \ \{ (x = v_2) \land (e \mapsto v_2) \} \]
where \( v_2 \) and \( x \) do not occur in \( e \)

Heap Assignment Axiom
\[ \{ e \mapsto - \} \ [e] := e_1 \ \{ e \mapsto e_1 \} \]
where \( (e \mapsto -) \) abbreviates \( \exists z. \ e \mapsto z \) and \( z \) does not occur in \( e \)

Allocation Assignment Axiom
\[ \{ x = v \land \text{emp} \} \ x := \text{cons}(e_1, e_2, \ldots, e_n) \ \{ x \mapsto e_1(v/x), e_2(v/x), \ldots, e_n(v/x) \} \]
where \( v \) is an auxiliary variable different from \( x \) and not appearing in \( e_1, e_2, \ldots, e_n \)

Derived Allocation Assignment Axiom
\[ \{ \text{emp} \} \ x := \text{cons}(e_1, e_2, \ldots, e_n) \ \{ x \mapsto e_1, e_2, \ldots, e_n \} \]
where \( x \) does not appear in \( e_1, e_2, \ldots, e_n \)

Dispose Axiom
\[ \{ e \mapsto - \} \ \text{dispose}(e) \ \{ \text{emp} \} \]
where \( (e \mapsto -) \) abbreviates \( \exists z. \ e \mapsto z \) and \( z \) does not occur in \( e \)

The Frame Rule:
\[
\frac{\{P\} S \{Q\}}{\{P \ast R\} S \{Q \ast R\}} \quad \text{where no variable modified by } S \text{ appears free in } R
\]

Other Rules: the Hoare Logic rules from Handout 3, except the Assignment rule, are also in the calculus for Separation Logic.