University of New South Wales
School of Computer Science and Engineering
System Modelling and Design (COMP2111)

FINAL TAKE-HOME EXAM — Term 1, 2021

Instructions:

- This exam has nine questions, for a total of 100 marks.
- **Produce a single PDF of your answers to all questions, with size less than 6MB and entitled "exam.pdf".**
- Submit your work using the `give` web interface, or by running (on a CSE machine): `give cs2111 exam exam.pdf` .
- You must include the following statement in your PDF file:
  I declare that all of the work submitted for this exam is my own work, completed without assistance from anyone else.

- The time allowed is **24 hours**: from **8AM** on Monday 3rd May to **8AM** on Tuesday 4th May.
- A reference sheet of logical laws is provided for your convenience.
- You must adhere to the UNSW student conduct requirements listed at `https://student.unsw.edu.au/conduct` .
- Remember that you can ask us questions on Ed during the exam! We will check Ed often, except when we sleep (10PM-6AM). In a pinch, you can also e-mail questions to cs2111@cse.unsw.edu.au.
- Don't leave submission until the last minute—submit early and often! Remember that `give` allows you to resubmit several times before the deadline.

Laws of Boolean Algebra

Associativity	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
Commutativity	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
Distribution	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Identity	$x \vee 0 = x$	$x \wedge 1 = x$
Complement	$x \vee x' = 1$	$x \wedge x' = 0$

Derived laws

Idempotence	$x \vee x = x$	$x \wedge x = x$
Annihilation	$x \vee 1 = 1$	$x \wedge 0 = 0$
Double complement	$(x')' = x$	

Inference rules for Hoare Logic

$$\frac{}{\{\varphi[e/x]\}\; x := e\; \{\varphi\}} \quad \text{(ass)}$$

$$\frac{\{\varphi\}\, P\, \{\psi\} \qquad \{\psi\}\, Q\, \{\rho\}}{\{\varphi\}\, P; Q\, \{\rho\}} \quad \text{(seq)}$$

$$\frac{\{\varphi \wedge g\}\, P\, \{\psi\} \qquad \{\varphi \wedge \neg g\}\, Q\, \{\psi\}}{\{\varphi\}\ \textbf{if}\ g\ \textbf{then}\ P\ \textbf{else}\ Q\ \textbf{fi}\ \{\psi\}} \quad \text{(if)}$$

$$\frac{\{\varphi \wedge g\}\, P\, \{\varphi\}}{\{\varphi\}\ \textbf{while}\ g\ \textbf{do}\ P\ \textbf{od}\ \{\varphi \wedge \neg g\}} \quad \text{(loop)}$$

$$\frac{\varphi' \to \varphi \qquad \{\varphi\}\, P\, \{\psi\} \qquad \psi \to \psi'}{\{\varphi'\}\, P\, \{\psi'\}} \quad \text{(cons)}$$

Inference rules for Natural Deduction

$\dfrac{A \qquad B}{A \wedge B}$ (∧-I)	$\dfrac{A}{A \vee B}$ (∨-I1)	$\dfrac{B}{A \vee B}$ (∨-I2)
$\dfrac{A \wedge B}{A}$ (∧-E1)	$\dfrac{A \wedge B}{B}$ (∧-E2)	$\dfrac{A \qquad \neg A}{\bot}$ (¬-E)
$\dfrac{A \to B \qquad A}{B}$ (→-E)	$\dfrac{A \leftrightarrow B \qquad A}{B}$ (↔-E1)	$\dfrac{A \leftrightarrow B \qquad B}{A}$ (↔-E2)
$\dfrac{A \vee B \qquad \begin{matrix}[A]\\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C\end{matrix}}{C}$ (∨-E)	$\dfrac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix}}{A \to B}$ (→-I)	$\dfrac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ A\end{matrix}}{A \leftrightarrow B}$ (↔-I)
$\dfrac{\begin{matrix}[A]\\ \vdots \\ \bot\end{matrix}}{\neg A}$ (¬-I)	$\dfrac{\begin{matrix}[\neg A]\\ \vdots \\ \bot\end{matrix}}{A}$ (IP)	$\dfrac{\bot}{A}$ (X)

$\dfrac{}{a = a}$ (=-I)	$\dfrac{a = b \qquad A(a)}{A(b)}$ (=-E1)	$\dfrac{a = b \qquad A(b)}{A(a)}$ (=-E2)
$\dfrac{A(c) \qquad \text{(1,2,3)}}{\forall x A(x)}$ (∀-I)	$\dfrac{A(c) \qquad \text{(2)}}{\exists x A(x)}$ (∃-I)	$\dfrac{\forall x A(x)}{A(c)}$ (∀-E)
$\dfrac{\exists x A(x) \qquad \begin{matrix}[A(c)]\\ \vdots \\ B\end{matrix} \qquad \text{(1,2,4)}}{B}$ (∃-E)		(1): c is arbitrary (2): x is not free in $A(c)$ (3): c is not free in $A(x)$ (4): c is not free in B

Questions

Each answer should be justified where appropriate, and 2–3 sentences should be sufficient.

Question 1 (5 marks)

Consider the following λ-term:

$$(\lambda x.\, f\ x)\ (\lambda x.\, (\lambda x.\, f\ x)\ x)\ y$$

(a) What are its free variables? (1 mark)

(b) What is its normal form? (i.e. its fully β-reduced form) (4 marks)

Question 2 (15 marks)

Let's add the operator $\oplus$, denoting *exclusive or*, to Boolean algebra. It has one law:

$$A \oplus B = (A \land B') \lor (A' \land B) \qquad (xor)$$

Which of the following algebraic identities are valid in all Boolean algebras? For the valid ones, prove them using the laws of Boolean algebra given in the reference sheet. For the invalid ones, show a counterexample. For the counterexamples, remember to state which Boolean algebra you're using, and how you instantiate the variables A, B.

(a) $A \oplus A = 0$ (5 marks)

(b) $A \oplus A' = 1$ (5 marks)

(c) $(A \oplus B)' = A' \oplus B'$ (5 marks)

Question 3 (13 marks)

Here are three alternative definitions of an interleaving function, which is a binary operator over the language Σ^* for some alphabet Σ.

$$\begin{array}{lcll}
\mathsf{interleave}_1(\lambda, w) & = & w & \\
\mathsf{interleave}_1(aw, w') & = & a(\mathsf{interleave}_1(w', w)) & \\
\\
\mathsf{interleave}_2(\lambda, w) & = & w & \\
\mathsf{interleave}_2(w, \lambda) & = & w & \text{if } w \neq \lambda \\
\mathsf{interleave}_2(aw, bw') & = & ab(\mathsf{interleave}_2(w, w')) & \\
\\
\mathsf{interleave}_3(w, \lambda) & = & w & \\
\mathsf{interleave}_3(\lambda, w) & = & \mathsf{interleave}_3(w, \lambda) & \text{if } w \neq \lambda \\
\mathsf{interleave}_3(aw, w') & = & a(\mathsf{interleave}_3(w', w)) & \text{if } w' \neq \lambda
\end{array}$$

(a) Show that the recursive definitions above will always terminate, by giving either a variant or a well-founded order, for

  (i) $\mathsf{interleave}_1$ (1 marks)

  (ii) $\mathsf{interleave}_2$ (1 marks)

  (iii) $\mathsf{interleave}_3$ (3 marks)

(b) Prove using induction that

$$\forall w, w'.\ \mathsf{interleave}_1(w, w') = \mathsf{interleave}_2(w, w')$$

Hint: induction on what? Structural induction is unlikely to be helpful. (8 marks)

Question 4 (10 marks)

Show, using Natural Deduction, that:

$$A \leftrightarrow (B \vee A) \quad \vdash \quad B \to A$$

Question 5 (12 marks)

Consider the predicate logic formula

$$\exists x \exists y (\neg(x = y))$$

(a) Is this formula *satisfiable*? Justify your answer. (4 marks)

(b) Is this formula a *logical validity*? Justify your answer. (4 marks)

(c) Is it possible to prove

$$\vdash \exists x \exists y (\neg(x = y))$$

using natural deduction? Justify your answer.

Hint: you don't have to *use* natural deduction to find the answer. (4 marks)

Question 6 (8 marks)

A robot moves around a 2-dimensional grid, starting at location $(0, 0)$. At any point it is able to make one of four moves:

$$(+2, -1) \qquad (+1, -2) \qquad (+1, +1) \qquad (-3, 0)$$

Here are some properties we could state about this transition system. For each property, state whether it's a safety property, a liveness property, or a reachability property.

(a) The robot will reach $(-2, +1)$. (1 marks)

(b) The robot can reach $(-2, +1)$. (1 marks)

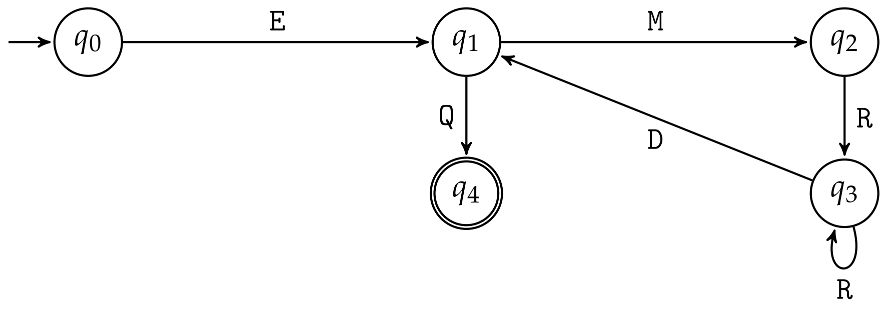
(c) The robot cannot reach $(-2, +1)$. (1 marks)

Then, finally:

(d) Explain, informally and in your own words, what the difference is between a safety property and a liveness property. (5 marks)

Question 7 (12 marks)

Here is a (simplified) automaton (called A) for the SMTP protocol from Assignment 2, where we use $\{\texttt{E}, \texttt{M}, \texttt{R}, \texttt{D}, \texttt{Q}\}$ to abbreviate $\{\texttt{EHLO}, \texttt{MAIL}, \texttt{RCPT}, \texttt{DATA}, \texttt{QUIT}\}$.



(a) Define a regular expression r over the alphabet $\{\texttt{E}, \texttt{M}, \texttt{R}, \texttt{D}, \texttt{Q}\}$ such that $\mathcal{L}(r) = \mathcal{L}(A)$. (4 marks)

(b) What is the largest number n such that A rejects every word of length exactly n? (4 marks)

(c) Prove that for every $m > n$ (where n is your answer for the previous question), there exists a word w of length m such that $w \in \mathcal{L}(a)$. (4 marks)

Question 8 (8 marks)

The (Python) program fragment at (1) below contains four variables `q`,`r`,`s`,`t` and three constants `Q`,`R`,`S`. From its pre- and postcondition, we can see that it "rotates" the values of the three variables `q`, `r` and `s`. But variable `t` does not appear in the postcondition: its name "`t`" stands for "temp". The three assertions *aaa*, *bbb* and *ccc* are for you to fill in, as your answer to this part of the question.

```
{q==Q and r==R and s==S} #    ← precondition
{aaa}
t= q
{bbb}
q= r
{ccc}                                              (1)
r= s
{ddd}
s= t
{q==R and r==S and s==Q} #    ← postcondition
```

In the assertions, we are using Python syntax: thus `and` for logical conjunction $\wedge$ and == for equality $=$, as in Python-style conditionals.

(a) Use the Hoare-Logic assignment axiom's *textual substitution* to figure out and write down what Assertion *ddd* is. (You do not have to give your working.) Do the substitution *exactly*, and do not modify or re-arrange *ddd* in any other way.

Hint: Your answer for *ddd* should have exactly the same number of characters as the postcondition. (1 mark)

(b) Do the same for *ccc* as you did for *ddd* in a. (1 mark)

(c) Do the same for *bbb*. (1 mark)

(d) Do the same for *aaa*. (1 mark)

(e) Is *aaa* textually the same as the precondition? (Yes or no.) (1 mark)

(f) Is *aaa* equivalent to the precondition? (Yes or no.) (1 mark)

(g) Is *aaa* implied by the precondition? (Yes or no.) (1 mark)

(h) Does the precondition imply *aaa*? (Yes or no.) (1 mark)

Question 9 (17 marks)

Suppose that the date today is 1 January 2021, so that "1 day ago" would be 31 December 2020, and "366 days ago" would be 1 January 2020 (because 2020 was a leap year). Assume that you have a function `DiY` such that that `DiY(y)` is the number of days in Year `y`.

The program in Fig. 1 below,[1] reads a natural number `G` (for "a`G`o") and sets variables `d` (for "day") and `y` (for "year") so that "`G` days ago" (from 1 January 2021) is the `d`-th day of Year `y`, where 1 January in any year is Day 1 (in that year). Thus for example with input `G==365` initially, the program should establish `y==2020` and `d==2` finally.

At (2) just below it's shown how the program was written — first the invariant

`G+DiY(y)>g` and "Day `1` in Year `y` is `g` days ago."

was chosen. The program text is in blue, and assertions are in black. (Remember there's Fig. 1 if you want to see just program text on its own.)

```
{G>=0 and "Today is 1 January 2021."} #     ← precondition
{aaa and "Today is 1 January 2021."}
g= 0
{bbb and "Day 1 in Year 2021 is g days ago."}
y= 2021
{ccc and ddd}
while G>g:
    {eee and fff and ggg}
    {G>g and "Day 1 in Year y-1 is g+DiY(y-1) days ago."}
    y= HHH                                                        (2)
    {G>g and "Day 1 in Year y is g+DiY(y) days ago."}
    g= III
    {jjj and kkk}

{lll and mmm and nnn}
{1<=g+1-G<=DiY(y) and "Day g+1-G of Year y is G days ago."}
d= OOO
{1<=d<=DiY(y) and "Day d of Year y is G days ago."} #     ← postcondition
```

This first part of this question asks you to fill in some assertions (lower-case triple letters like *xxx*) in Program (2).

[1]. . . once the missing `HHH`, `III` and `OOO` are supplied. . .

(a) Which of the assertions *aaa–ggg* and *jjj–nnn* should be the first conjunct `G+DiY(y)>g` of the invariant? Just give their names. (2 marks)

Hint: There are four occurrences of `G+DiY(y)>g` among *aaa–ggg* and *jjj–nnn* altogether. Do not worry about which of several you pick on a single line: but (obviously) don't use any of them more than once.

(b) Which of the assertions *aaa–ggg* and *jjj–nnn* should be the second conjunct "Day `1` in Year `y` is `g` days ago." of the invariant? (2 marks)

(c) Which assertion should be the loop guard `G>g`? (1 mark)

(d) Which assertion should be negation of the loop guard? (1 mark)

(e) Fill in the remaining (two) assertions that you haven't already filled in with your answers to the questions a–d just above. Make sure they're valid with respect to the assertions already there and any program code that is already completed. (Some program code is yet to be filled in.) (2 marks)

For the assignment statements, you are asked to fill in the right-hand sides where they are missing (upper-case letters like `XXX`).

(f) What should the expression `HHH` be?

Hint: Use the assertions on either side of it. (2 marks)

(g) What should the expression `III` be? (2 marks)

(h) What should the expression `OOO` be? (2 marks)

Now we have to do some arithmetic. Answer this question in no more than two sentences:

(i) When one (or several) assertions appear on successive lines, the earlier assertions taken together (i.e. their conjunction) must imply the later ones.

Explain why the assertion {*lll* `and` *mmm* `and` *nnn*} that you filled in earlier implies the assertion {"Day `g+1-G` of Year `y` is `G` days ago."} on the following line. (2 marks)

Finally

(j) Give in no more than two sentences the reason this program is guaranteed to terminate. You do not have to mention the "variant", but your answer must be crystal clear. (1 mark)

If you do mention the variant, you might manage to give a one-sentence answer, such as "The variant is ..."

```
def DiY(y):
    r= 365
    if y%4==0:   r= 366
    if y%100==0: r= 365
    if y%400==0: r= 366
    return r

from sys import stdin
G= int(stdin.readline())

g= 0
y= 2021
while G>g:
    y= HHH
    g= III
d= OOO

print("Day %d of Year %d was %d days ago."%(d,y,G))
```

Today is assumed to be 1 January 2021, and `G` is a number of days "ago".

Figure 1: "Days ago" calculator (incomplete)

Dont forget to submit your work and include the statement given on the front page.

— END OF EXAM —