Name: ....
Student Number ....

Software Engineering 2F04

DAY CLASS — Dr. Mark Lawford
DURATION OF EXAMINATION: 3 Hours
McMaster University Final Examination — December 2000

THIS EXAMINATION PAPER INCLUDES 6 PAGES AND 4 QUESTIONS. YOU ARE RESPONSIBLE FOR ENSURING THAT YOUR COPY OF THE PAPER IS COMPLETE. BRING ANY DISCREPANCY TO THE ATTENTION OF YOUR INVIGILATOR.

Special Instructions: The use of calculators, notes, and text books is not permitted during this exam. Answer all questions in the provided answer booklets. Fill in your name and student number and sign each booklet you use. This paper must be returned with your answers.

Useful Factoids:

Rules Governing Equality

Ia : $\vdash (\forall x)(x = x)$

Ib : $\vdash (\forall x)[x = t \rightarrow (\phi \leftrightarrow \phi[x, t|x])]$

Ic : $\vdash (\forall x)(\forall y)(x = y \rightarrow y = x)$

Id : $\vdash (\forall x)(\forall y)(\forall z)(x = y \wedge y = z \rightarrow x = z)$

1. Propositional & Predicate Logic (30 marks)

a) (6 marks) Show the following useful tautologies:

i)

$$(P \rightarrow (Q \rightarrow R)) \Leftrightarrow (P \wedge Q \rightarrow R)$$

ii)

$$(P \vee Q \rightarrow R) \Leftrightarrow (P \rightarrow R) \wedge (Q \rightarrow R)$$

b) (5 marks) Consider the following predicate logic formula:

$$\phi : (\forall x)(\forall y)(\forall z)(Rxy \wedge Ryz \rightarrow Rxz)$$

i) Find an interpretation structure $\mathbf{S_1}$ such that $\mathbf{S_1} \models \phi$.
ii) Find an interpretation structure $\mathbf{S_2}$ such that $\mathbf{S_2} \models \neg\phi$.
iii) Is ϕ a logical theorem?

Continued on page 2

c) (6 marks) Determine if the following set of premises is consistent or inconsistent. Justify your answer with a formal proof or interpretation structure as need be.

$$\Gamma := \{(\forall x)[Px \to (\exists y)Qxy], \neg[(\exists x)\neg Px \vee (\exists y)Qyy]\}$$

d) (7 marks) Determine if the following argument is valid or invalid. Justify your answer with a formal proof or interpretation structure as need be.

Premises: $(\forall x)(Rax \to a = x \vee a = b), (\exists x)Rax, Sa \wedge \neg Sb$

Conclusion: Raa

e) (6 marks) Determine if the following set of premises is consistent or inconsistent. Justify your answer.

$$\Gamma := \{(\exists x)(\forall y)(x = y), (\exists x)(\exists y)x \neq y\}$$

2. Partial Functions and PVS Typechecking (20 marks)

a) (6 marks) Consider the following formula know as the "axiom of reflection" in traditional logical systems:

$$(\forall x)(\forall y)(x = y \to f(x) = f(y))$$

i) Show that this formal is a logical theorem of our (Rubin) traditional logic system by formally proving:

$$\vdash (\forall x)(\forall y)(x = y \to f(x) = f(y))$$

ii) When considered in a traditional analysis ("Parnas") style logic system, this formula is not a logical theorem. i.e.,

$$\not\vdash (\forall x)(\forall y)(x = y \to f(x) = f(y))$$

Supply an concrete interpretation for f using the universe of real numbers and explain why this is the case.

b) (10 marks) Create the "best" PVS definitions for the following functions:

i) $f(x) = x^2 + 4x - 5$ [Hint: use a little calculus to figure out the minimum of $f(x)$.]

ii) For $f(x)$ defined as above:

$$g(x) = \frac{1}{f(x) - 9}$$

iii) For $f(x)$ defined as above:

$$h(x, y) = \frac{1}{y - f(x) - 9}$$

c) (4 marks) Let $\phi[t|x]$ be a valid substitution. Explain why

$$\Gamma, (\forall x)\phi \vdash \psi \text{ iff } \Gamma, (\forall x)\phi, \phi[t|x] \vdash \psi$$

Continued on page 3

3. Predicate Logic & Mathematical Induction (20 marks)

a) (5 marks) Assuming that our universe of interpretation is the natural numbers, write down the formal predicate logic equation that represents the statement:

For every n the quantity $11^n - 4^n$ *is divisible by* 7.

b) (10 marks) Use mathematical induction to informally prove the statement from part (a).

c) (5 marks) Let us consider why rule MI is a valid rule of inference. In this problem you will show how the antecedents of rule MI can be used to write down a proof of $\phi[2|n]$ and $\phi[a|n]$ in general. Let $\Gamma := \{\phi[0|n], (\forall m)(\phi[m|n] \rightarrow \phi[m+1|n])\}$.

i) Formally show that $\Gamma \vdash \phi[2|n]$.

ii) What lines would you add to the proof in (i) to show $\Gamma \vdash \phi[3|n]$?

iii) Given some $a \in \mathbb{N}$, how many lines would it take to show $\Gamma \vdash \phi[a|n]$?

4. Software Verification with PVS (30 marks)

In this problem we will add some additional functionality to the simplified pressure sensor trip example from Assignment 5. Recall that the pressure trip monitors a pressure sensor and is then "tripped" when the sensor value exceeds a normal operating setpoint. The pressure sensor trip makes use of deadbands to eliminate sensor chatter. An updated version of the specification and the actual implementation for the sensor trip are give in Figure 1 by `f_PressTrip` and `PTRIP`, respectively. In the function definitions, `f_PressTripS1` and `PREV` play corresponding roles as the arguments for the previous value of the state variable computed by the function.

The definitions of `f_PressTrip` and `PTRIP` have been modified so that the theorem `Sentrip1` at the bottom of the specification in Figure 1 is now easily proved. Thus we conclude that the implementation `PTRIP` will produce the correct output that is equivalent to the specification `f_PressTrip` output for all possible inputs.

As the software project's formal verification expert, you have been assigned to check the implementation of some new functionality that has been added to the pressure trip module. The module is now suppose to also implement a trip status indicator that is used to flag when pressure sensor trip has occurred. Once every 3 seconds the Trip Computer transmits the status indicator flag to the operator's display computer. The transmitted indicator value depends upon the history of the pressure sensor trip in the previous 3 seconds. If there was a sensor trip at any time during the last 3 seconds, the transmitted indicator value is $TRUE$, otherwise, it is $FALSE$.

The specification of the trip status indicator function is given by the vertical condition table `f_PressStatus` shown in Figure 2. When the condition on the left is $TRUE$, the value on the right is returned. The interpretation of this table is that if the current value of `f_PressTrip` is tripped (i.e., there is a sensor trip) then the status indicator `f_PressStatus` is set to $TRUE$. When there is not a sensor trip, if it is time to transmit (i.e., variable `Transmit` is $TRUE$ when the current time is a multiple of 3 seconds) then `f_PressStatus` is "cleared" by setting it to $FALSE$. Otherwise `f_PressStatus` is left at its previous value `f_PressStatusS1`.

Figure 2 also contains the formatted PVS for this version of the implementation. To efficiently meet all the specifications for the pressure trip module, the developers have decided to partially compute the status output at the same time that `PTRIP` is computed. This is done by using a table of the same structure as `PTRIP` in the implementation function `STATUS`. Here `PREV` is the previous value of `STATUS`.

Continued on page 4

sentrip : THEORY
BEGIN

Trip : TYPE = {Tripped, NotTripped}

AItype : TYPE = {i : nat | $0 \leq$ i $\wedge$ i ≤ 5000}

f_PressTrip(Pressure : real, f_PressTripS1 : Trip) : Trip = TABLE

Pressure < 2400	2400 ≤ Pressure ∧ Pressure < 2450	Pressure ≥ 2450
NotTripped	f_PressTripS1	Tripped

ENDTABLE

PTRIP(PRES : AItype, PREV : bool) : bool = TABLE

PRES < 2400	2400 ≤ PRES ∧ PRES < 2450	PRES ≥ 2450
FALSE	PREV	TRUE

ENDTABLE

Trip2bool(TripVal : Trip) : bool = TABLE

TripVal = Tripped	TripVal = NotTripped
TRUE	FALSE

ENDTABLE

bool2Trip(BoolVal : bool) : Trip = TABLE

BoolVal = TRUE	BoolVal = FALSE
Tripped	NotTripped

ENDTABLE

real2AItype(x : real) : AItype = TABLE

$x \leq 0$	$0 < x \wedge x < 5000$	$x \geq 5000$
0	floor(x)	5000

ENDTABLE

Sentrip1 : THEOREM
(∀ (Pressure : real, f_PressTripS1 : Trip) :
f_PressTrip(Pressure, f_PressTripS1) =
bool2Trip(PTRIP(real2AItype(Pressure), Trip2bool(f_PressTripS1))))

END sentrip

Figure 1: Formatted PVS specification for the fixed pressure sensor trip example

Continued on page 5

f_PressStatus(f_PressTrip : Trip, f_PressStatusS1, Transmit : bool) : bool = TABLE

f_PressTrip = Tripped	TRUE
$\neg$(f_PressTrip = Tripped) $\wedge$ Transmit	FALSE
$\neg$(f_PressTrip = Tripped) $\wedge$ $\neg$Transmit	f_PressStatusS1

ENDTABLE

STATUS(PRES : AItype, PREV : bool) : bool = TABLE

PRES < 2400	2400 ≤ PRES ∧ PRES < 2450	PRES ≥ 2450
PREV	PREV	TRUE

ENDTABLE

Status1 : THEOREM
($\forall$ (Pressure : real, f_PressTripS1 : Trip, f_PressStatusS1, Transmit : bool) :
f_PressStatus(f_PressTrip(Pressure, f_PressTripS1), f_PressStatusS1, Transmit) =
IF$\neg$(Transmit) THEN STATUS(real2AIType(Pressure), f_PressStatusS1)
ELSE FALSE
ENDIF)

Figure 2: Formatted PVS input for the trip status indicator block comparison

In addition to the tabular function, the implementation contains the main program thread that provides the function call sequence for the main program loop. The computation of the implementation status output is finished in in the main program thread by a conditional statement that checks the `Transmit` value to determine when it is time to transmit the status, in which case it resets the indicator value to *FALSE* (i.e., it sets the value the `STATUS` to to *FALSE* once it is transmitted). This part of the status indicator computation is modeled by the IF-THEN-ELSE statement that is part of the block comparison theorem.

The definitions from Figure 2 are appended to the specification in Figure 1. Attempting to prove the block comparison theorem `Status1` results in several unprovable sequents, including the one below:

```
{-1}    Transmit!1
{-2}    f_PressTripS1!1 = Tripped
  |-------
{1}    Pressure!1 < 2400
```

a) (5 marks) Write down the characteristic equation for the unprovable sequent.

b) (5 marks) State a new theorem called `Status2` that could be used to prove that the implementation does not meet the specification. This would provide confirmation that the unprovable sequent for theorem `Status1` results from inconsistencies between the specification and implementation and not from a poor choice of PVS prover commands by the verifier. This is an example of refutation theorem proving where a software engineer tries to prove that the implementation is *NOT* equivalent to the specification.

c) (5 marks) Find all the values of `Transmit!1`, `Pressure!1` and `f_PressTripS1!1` that provide counter examples for the characteristic equation you found in (a).

Continued on page 6

d) (5 marks) Pick specific values for `Transmit!1`, `Pressure!1`, `f_PressStatusS1!1` and `f_PressTripS1!1` that provide a counter example and confirm that it provides a counter example to theorem `Status1` by evaluating

```
f_PressStatus(f_PressTrip(Pressure!1,f_PressTripS1!1),f_PressStatusS1!1,Transmit!1)
```

and

```
IF NOT(Transmit) THEN STATUS(real2AItype(Pressure),f_PressStatusS1) ELSE FALSE
```

and comparing the results.

e) (5 marks) How could the implementation be altered to fix this problem? (HINT: Think about how you could use the current value of `PTRIP` in the main program loop.)

f) (5 marks) Assume that initially `f_PressTripS1= NotTripped` and that the other previous state values are initially $FALSE$. The update functions `f_PressTrip, PTRIP, f_PressStatus` and `STATUS` are called once every second and the value of transmit is $TRUE$ every 3 seconds. Sketch a sequence of inputs for `Pressure` and the resulting sequence of outputs produced by the specification and implementation that results in specification and implementation having different status outputs for an extended period of time.

"Why did I choose software? ...I hate everything about it!" - 2F04 student

The End