SFWR 2AA4/CS 2ME3 Mid-Term Exam March 2013

Student Name: ______________________________ **Student Number:** ______________

This is a 2 hour open-book exam.

Attempt to answer every question. Total 10 questions. Total possible marks is 50.

Sign the optical scanning sheet in pen. Use a pencil to mark your answers on the optical scanning sheet and make sure you hand in both the question paper and the optical scanning sheet. Make sure your name and student number are on both the question paper and optical scanning sheets. "Bubble-in" your student number. You will not receive a grade if both papers are not handed in.

READ THE QUESTIONS CAREFULLY – PLEASE!

You can write on this question paper. Use it as a working area. It will not be graded! You can also use the rough paper we handed out as a work area. Do not hand in that paper.

Question 1. 3 marks

Consider the following tabular expression and then answer the questions that follow. (m_boolean: bool, m_value: int, f_time: real, k_constant: int, f_result, k_one, k_two and k_three are all of the same type.)

			f_result
$m\_value \geq k\_constant$			k_one
$m\_value < k\_constant$	m_boolean		k_two
	NOT m_boolean	$f\_time \geq 55.0$	k_one
		$0.0 \leq f\_time < 55.0$	k_three

If k_constant = 3 and m_value = 1 and m_boolean = true and f_time = 56 then what is the value of f_result, and is the table complete and disjoint? (f_time is a global time variable. It cannot be negative.)

A) f_result = k_one, and the table is complete and disjoint
[✓ B) f_result = k_two, and the table is complete and disjoint]
C) f_result = k_three, and the table is complete and disjoint
D) f_result not known because the table is not complete, but it is disjoint
E) f_result = k_one, and the table is complete but not disjoint

Question 2. **5 marks**

Consider the following specification:

{n>0 **and for all** i,j (0 ≤ i < n **and** 0 ≤ j < n **and** i ≠ j) **implies** z[i] ≠ z[j]}
myproc(n: IN integer; z[]: UPDATE float array of maximum n elements;
m: OUT integer)
{(t = (∑'z[i])/n (i=0,2,…,n-1)) **and** (m ≤ n) **and**
(**for all** i ((0 ≤ i < n **and** 'z[i] ≤ t) **implies**
(**exists** j (0 ≤ j < m **and** z'[j] = 'z[i]))))
and
(**for all** i ((0 ≤ i < n **and** 'z[i] > t) **implies**
(**not exists** j (0 ≤ j < m **and** z'[j] = 'z[i]))))}

Which of the following segments of code in ***myproc*** most accurately implements this spec?

A)
```
j: integer; a,x: float;
x := 0;
for j := 0 to n - 1 do
begin
   x := x + z[j];
end;
a := x / n;
m := 0;
for j := 0 to n - 1 do
begin
   if float(z[j]) <= a then
   begin
      z[m] := z[j];
      m = m + 1;
   end;
end;
```

B)
```
i,j: integer; x: float;
x := 0;
for i := 0 to n - 1 do
begin
   x := x + z[i];
   x := x / n;
   m := 0;
   for j := 0 to n - 1 do
   begin
      if float(z[j]) <= x then
      begin
         z[m] := z[j];
         m = m + 1;
      end;
   end;
end;
```

C)
```
i,j: integer;
x: float;
x := 0;
for i := 0 to n - 1 do
begin
   x := x + z[i];
end;
x := x / n;
m := 0;
for j := 0 to n - 1 do
begin
   if float(z[j]) <= x then
   begin
      z[m] := z[j];
   end;
   m = m + 1;
end;
```

D)
```
i,j: integer; x,y: float;
x := 0;
for j := 0 to n - 1 do
begin
   x := x + z[j];
end;
y := x / n;
m := 0;
for j := 0 to n - 1 do
begin
   if float(z[j]) >= y then
   begin
      z[m] := z[j];
      m = m + 1;
   end;
end;
```

E) **None of the above is good enough.**

Question 3. 2 marks

Consider the following:

a,b: boolean
c,d,z: integer
e,f,g: char

When a and b are both true and $z = 0$, then $e = g$. When a and b are both true and $z \neq 0$, and if $c > z$ then e = "q", else if $c \leq z$ then e = "0". If a is not true, or b is not true, and if $c > d$ then e = "1", else e ="0".

Which of the following tabular expressions describe the behaviour correctly?

A)

			e
$a \wedge b$	$z = 0$		g
	$z \neq 0$	$c > z$	"q"
		$c \leq z$	"0"
$\neg a \vee \neg b$	$c > d$		"1"
	$c \leq d$		"0"

B)

			e
$a \wedge b$	$z = 0$	$c > z$	g
		$c \leq z$	g
	$z \neq 0$	$c > z$	"q"
		$c \leq z$	"0"
$\neg a \vee \neg b$	$c > d$		"1"
	$c \leq d$		"0"

C)

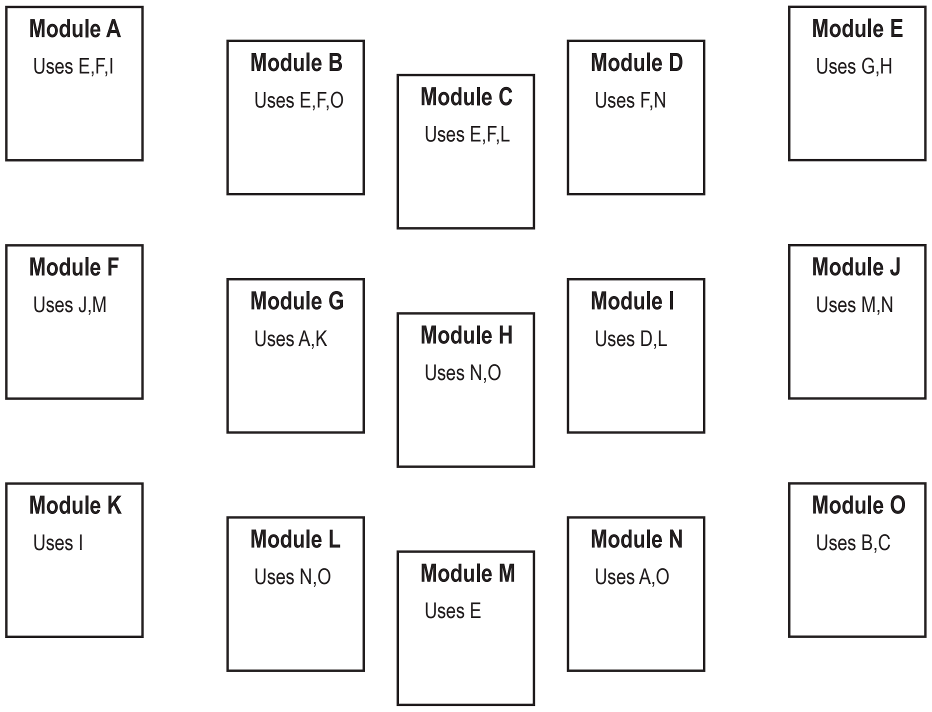
		e
$a \wedge b$	$z = 0$	g
	$c > z$	"q"
	$c \leq z$	"0"
$\neg a \vee \neg b$	$c > d$	"1"
	$c \leq d$	"0"

[✓ **D) Both A and B**]

E) None of the above

Question 4. 7 marks

The following figure describes a design involving 15 modules/classes. The only information provided in the figure is the name of each module and the names of other modules each module "uses".



The following are statements reflecting judgement of the design. Select the statement that you think is most accurate.

A) The design is excellent since cohesion is high and coupling is low.
B) The design is poor because cohesion is low and coupling is high.
[✓ **C) The design is poor since the uses relationship is far from a hierarchy.**]
D) The design is good because any one module uses at most 3 other modules.
E) The design is poor because too many modules use modules E and F.

Question 5. 5 marks

With reference to the design above, the following changes are suggested.

G.method1 will not call A.method3 as it used to. Instead it will call N.method1.
N.method5 will not call A.method3 as it used to. It will also call N.method1.
O.method2 will not call B.method1 or C.method1 as it used to. Instead it will call local methods.
From the choices below, indicate how you think these changes will affect the design.

A) It will have no real impact on the design. It should just be equivalent.
B) This will harm the design since it will result in less cohesion.
[✓ **C) This improves the design since it removes cycles in the uses relationship.**]
D) This improves the design since it will improve cohesion and the uses relationship.
E) None of the above.

Question 6. **5 marks**

Consider the following code written in pseudo-code where : precedes a type declaration, and comments are in curly brackets.

```
public class statistics;

Interface
     newValue(value: float);
     getMin(): float;
     getMax(): float;
     getAve(): float;
     getNumValues(): integer;
     initialize();

Implementation
     aveValue, minValue, maxValue, sumValues: float;
     numValues: integer;

     public newValue(value: float);
     {
          numValues = numValues+1;
          if value < minValue then minValue = value;
          if value > maxValue then maxValue = value;
          sumValues = sumValues+value;
          aveValue = sumValues/numValues;
     }

     public getMin(): float;
     {
          getMin = minValue;
     }

     public getMax(): float;
     {
          getMax = maxValue;
     }

     public getAve(): float;
     {
          getAve = aveValue;
     }

     public getNumValues(): integer;
     {
          getNumValues = numValues;
     }

     public initialize();
     {
          numValues = 0;
          sumValues = 0.0;
          minValue = maxFloat; {assume it is defined}
          maxValue = minFloat; {assume it is defined}
     }
```

Choose the most accurate statement about this design from the selection below.

A) In the method *newValue*, the variables are classified as:
value is an input variable,
aveValue, minValue, maxValue, numValues and sumValues are output variables.
In the method *getMin*, getMin is an output and minValue is an input.
In the method *initialize*, minValue, maxValue, numValues, sumValues, are output variables.

B) In the method *newValue*, the variables are classified as:
value is an input variable,
aveValue, minValue, maxValue, numValues and sumValues are update variables.
In the method *getMin*, getMin is an input and minValue is an output.
In the method *initialize*, minValue, maxValue, numValues, sumValues, are output variables.

C) In the method *newValue*, the variables are classified as:
value is an input variable,
aveValue, minValue, maxValue, numValues and sumValues are update variables.
In the method *getMin*, getMin is an output and minValue is an input.
In the method *initialize*, minValue, maxValue, numValues, sumValues, are output variables.

D) In the method *newValue*, the variables are classified as:
value is an input variable,
aveValue, minValue, maxValue, numValues and sumValues are update variables.
In the method *getMin*, getMin is an output and minValue is an input.
In the method *initialize*, minValue, maxValue, numValues, sumValues, are update variables.

E) None of the above.

Question 7. 5 marks

"Information Hiding" is one of the most important design principles we use. Which of the following statements most accurately describes the principle of *Information Hiding*?

A) Information hiding aids us in producing maintainable designs by instructing us to hide how a module/class is implemented and not expose aspects of the implementation through the class interface.

B) Information hiding aids us in producing more efficient designs by instructing us to hide how a module/class is implemented and not expose aspects of the implementation through the class interface.

C) Information hiding aids us in producing maintainable designs by instructing us to hide each requirement or design decision that is likely to change inside a single module/class, and not expose them on the class interfaces.

D) Information hiding aids us in producing maintainable designs by instructing us to hide how to implement data structures inside a module/class whenever possible.

E) None of the above.

Question 8. **3 marks**

Consider the following Module Interface Specification.

MIS IntegerStack
Uses: None
Exports:
- **Constants**: maxElements
- **Types**: stack_type: integer_array[maxElements]
- **Variables**: stack: stack_type
  stackPointer: integer
- **Methods**:
  - push(value: **integer**): **boolean**
  - pop(): **integer**
  - top(): **integer**
  - isEmpty(): **boolean**
  - isFull(): **boolean**

End MIS IntegerStack

Which of the following statements concerning this MIS is the most accurate?

A) The MIS seems reasonable as far as I can judge. It would be better if the behaviour of each method was specified, and if the method *“push”* did not return a value.
B) The MIS is adequate since each entity has a type specified and judging by the names of the methods, expected functionality is provided.
C) The MIS is very poor since only syntax is provided, no semantics. Also, the internal state of the module is disclosed to users through the interface.
D) The MIS could be improved by specifying behaviour for each method, and the method *“push”* should not return any value.
E) None of the above is true.

Question 9. **10 marks**

You are designing a very simple remote control for a TV set. The control has a <Power> (on/off) button and a channel up <+> and channel down <-> button. The TV can receive 309 channels (channels numbered 1 through 309). When the power is on, and the <+> button is pressed, the channel goes “up” one channel, i.e. channel' = 'channel+1. The channels “wrap-around”. This means that if the <+> button is pressed and if 'channel was already at 309 then channel' = 1. Similarly, pressing the <-> button results in channel' = 'channel-1 and if 'channel was already equal to 1 then channel' = 309. The TV always stores the last channel (LC) that was selected and uses this channel when <Power> causes a change from off to on.

Which of the following tabular expressions describes the correct behaviour of the TV set? We use inputs <Power> (on/off), <+> and <->, and notation LC_{-1} and C_{-1}, to represent the previous value of LC and C (the current channel). The behaviour of C (and LC) is described using a finite state machine model. C=0 represents that the TV is off. Assume that the power is off initially (C_{-1}=0), and LC_{-1}=155 the very first time the TV is powered up.

A)

			LC	C
$C_{-1} = 0$	<Power>		LC_{-1}	LC_{-1}
	<+>	$C_{-1} \geq 309$	C_{-1}	1
		$C_{-1} < 309$	C_{-1}	$C_{-1}+1$
	<->	$C_{-1} > 1$	C_{-1}	$C_{-1}-1$
		$C_{-1} \leq 1$	C_{-1}	309
$C_{-1} > 0$	<Power>		C_{-1}	0
	<+>		C_{-1}	$C_{-1}+1$
	<->		C_{-1}	$C_{-1}-1$

B)

			LC	C
$C_{-1} = 0$	<Power>		LC_{-1}	LC_{-1}
	<+>		NC	NC
	<->		NC	NC
$C_{-1} > 0$	<Power>		C_{-1}	0
	<+>	$C_{-1} \geq 309$	1	1
		$C_{-1} < 309$	$C_{-1}+1$	$C_{-1}+1$
	<->	$C_{-1} > 1$	$C_{-1}-1$	$C_{-1}-1$
		$C_{-1} \leq 1$	309	309

C)

			LC	C
$C_{-1} = 0$	<Power>		LC_{-1}	LC_{-1}
	<+>		LC_{-1}	C_{-1}
	<->		LC_{-1}	C_{-1}
$C_{-1} > 0$	<Power>		LC_{-1}	0
	<+>	$C_{-1} \geq 309$	C_{-1}	1
		$C_{-1} < 309$	C_{-1}	$C_{-1}+1$
	<->	$C_{-1} > 1$	C_{-1}	$C_{-1}-1$
		$C_{-1} \leq 1$	C_{-1}	309

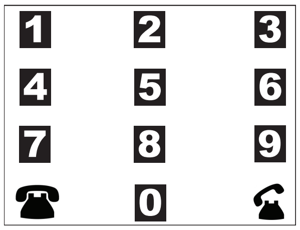
D)

			LC	C
$C_{-1} = 0$	<Power>		LC_{-1}	LC_{-1}
	<+>		NC	NC
	<->		NC	NC
$C_{-1} > 0$	<Power>		LC_{-1}	0
	<+>	$C_{-1} \geq 309$	LC_{-1}	1
		$C_{-1} < 309$	LC_{-1}	$C_{-1}+1$
	<->	$C_{-1} > 1$	LC_{-1}	$C_{-1}-1$
		$C_{-1} \leq 1$	LC_{-1}	309

E) **None of the above.**

Question 10. **5 marks**

Consider the very simple phone dial-pad pictured below.



We can let m_onHook and m_offHook be the monitored variables representing on-hook (no line available) and off-hook (line available), and let m_digit be any one of the numbered digits.

We want to explore possible input sequences in a hotel to dial from one room to another. The hotel specifies that to dial to another room you need to dial the three digit number of the room. Of course you can only dial when the phone is off-hook. Placing the phone on-hook kills the call. Dialing too many digits results in calling the room corresponding to the first three digits dialed (and lots of noise on the line when you press more digits).

Which of the following input sequences is sufficient to define all possible input behaviours? Ignore which actual digit, 0..9, has been pressed. Initially assume the line is not available.

A) **m_offHook**
m_offHook.m_digit.m_digit.m_digit

B) **m_offHook**
m_offHook.m_digit
m_offHook.m_digit.m_digit
m_offHook.m_digit.m_digit.m_digit

[✓ **C)**] **m_onHook**
m_offHook
m_offHook.m_digit
m_offHook.m_digit.m_digit
m_offHook.m_digit.m_digit.m_digit

D) **m_onHook**
m_offHook
m_offHook.m_digit
m_offHook.m_digit.m_onHook
m_offHook.m_digit.m_digit
m_offHook.m_digit.m_digit.m_onHook
m_offHook.m_digit.m_digit.m_digit
m_offHook.m_digit.m_digit.m_digit.m_onHook

E) **None of the above.**