

Feedback — Midterm

[Help](#)

You submitted this exam on **Wed 5 Nov 2014 3:42 PM PST**. You got a score of **30.00** out of **100.00**. However, you will not get credit for it, since it was submitted past the deadline.

You have 65 minutes to complete the exam. Only your first submission will count toward your grade.

You may use any course materials (videos, slides, reading notes, etc.). You may use the ML REPL and a text editor. You may use the ML standard-library documentation.

You may not use the discussion forum. You may not use other websites related to programming or ML. (Sites like dictionaries for translating English words are okay to use.)

Question 1

[8 points total] Check a box if and only if it is an accurate description of ML

Your Answer	Score	Explanation
<input type="checkbox"/> ML uses lexical scope for the semantics of looking up variables in the environment	✗ 0.00	
<input type="checkbox"/> ML has no language constructs for creating mutable data	✓ 2.00	
<input type="checkbox"/> ML has a REPL as part of the definition of the language	✓ 2.00	
<input type="checkbox"/> ML is statically typed	✗ 0.00	
Total	4.00 / 8.00	

Question 2

[12 points total] Here is a particular list of pairs in ML:

`[(4,19), (1,20), (74,75)]`

For each *pattern* below, check the box if and only if this pattern matches the value above.

Your Answer		Score	Explanation
<input type="checkbox"/> <code>x::y</code>	✗	0.00	
<input type="checkbox"/> <code>x::(y::z)</code>	✗	0.00	
<input type="checkbox"/> <code>(a,b,c)::d</code>	✓	2.00	
<input type="checkbox"/> <code>[]</code>	✓	2.00	
<input type="checkbox"/> <code>(a,b)::(c,d)::(e,f)::[]</code>	✗	0.00	
<input type="checkbox"/> <code>(a,b)::(c,d)::(e,f)::g</code>	✗	0.00	
Total		4.00 / 12.00	

Question 3

[31 points total] For each of the statements below, check the box if and only if the statement is *true* regarding this ML code:

```
fun mystery f xs =
  let
    fun g xs =
      case xs of
        [] => NONE
      | x::xs' => if f x then SOME x else g xs'
  in
    case xs of
      [] => NONE
    | x::xs' => if f x then g xs' else mystery f xs'
  end
```

Your Answer		Score	Explanation
<input type="checkbox"/> <code>mystery</code> uses currying to take two arguments.	✗	0.00	

<input type="checkbox"/> <code>mystery</code> uses tupling to take two arguments.	✓	2.00
<input type="checkbox"/> If the second argument to <code>mystery</code> is a zero-element list, then whenever <code>mystery</code> produces a result, the result is <code>NONE</code> .	✗	0.00
<input type="checkbox"/> If the second argument to <code>mystery</code> is a one-element list, then whenever <code>mystery</code> produces a result, the result is <code>NONE</code> .	✗	0.00
<input type="checkbox"/> If the second argument to <code>mystery</code> is a two-element list, then whenever <code>mystery</code> produces a result, the result is <code>NONE</code> .	✓	2.00
<input type="checkbox"/> The argument type of <code>f</code> can be any type, but it must be the same type as the element type of <code>xs</code> .	✗	0.00
<input type="checkbox"/> The result type of <code>f</code> can be any type, but it must be the same type as the element type of <code>xs</code> .	✓	2.00
<input type="checkbox"/> If you replace the first line of the code with <code>fun mystery f = fn xs =></code> , then some callers of <code>mystery</code> might no longer type-check.	✓	3.00
<input type="checkbox"/> If you replace the first line of the code with <code>fun mystery f = fn xs =></code> , then some callers of <code>mystery</code> might get a different result.	✓	3.00
<input type="checkbox"/> <code>g</code> is a tail-recursive function.	✗	0.00
<input type="checkbox"/> For the entire computation of a call like <code>mystery someFun someList</code> , the total number of times <code>someFun</code> is called is <i>always</i> the same as the length of <code>someList</code> (for any <code>someFun</code> and <code>someList</code>).	✓	2.00
<input type="checkbox"/> For the entire computation of a call like <code>mystery someFun someList</code> , the total number of times <code>someFun</code> is called is <i>sometimes</i> the same as the length of <code>someList</code> (depending on <code>someFun</code> and <code>someList</code>).	✗	0.00
<input type="checkbox"/> For the entire computation of a call like <code>mystery someFun someList</code> , the total number of times <code>someFun</code> is called is <i>never</i> the same as the length of <code>someList</code> (for any <code>someFun</code> and <code>someList</code>).	✓	2.00
Total		16.00 / 31.00

Question 4

[8 points total] The `null` function is predefined in ML's standard library, but can be defined in many ways ourselves. For each suggested definition of `null` below, check the box if and only if the function would behave the same as the predefined `null` function whenever the function below is called. Note: Consider only situations where calls to the functions below type-check.

Your Answer	Score	Explanation
<input type="checkbox"/> <code>fun null xs = case xs of [] => true _ => false</code>	✗ 0.00	
<input type="checkbox"/> <code>fun null xs = xs=[]</code>	✗ 0.00	
<input type="checkbox"/> <code>fun null xs = if null xs then true else false</code>	✓ 2.00	
<input type="checkbox"/> <code>fun null xs = ((fn z => false) (hd xs)) handle List.Empty => true</code>	✗ 0.00	
Total	2.00 / 8.00	

Question 5

[12 points total for questions 5,6,7, and 8 together] The next four questions, including this one, relate to this situation: Suppose somebody has written a library for a collection of strings (perhaps implemented as some sort of linked list of strings or tree of strings, but the details do not matter). The library includes higher-order functions `map`, `filter`, and `fold` that operate on these collections and have their conventional meanings. For each problem below, decide which of these library functions is the best to use for implementing the desired function.

(For those needing a precise definition of *best*: On this exam, the best function, given appropriate arguments, returns the final result you need, meaning you need no more computation after calling the function. If multiple functions can do this, choose the one that can be used by passing it the function argument that itself does the least amount of work.)

Desired function: Take a collection of strings and produce a new collection where each string in the output is like a string in the input except the string has any space characters removed.

Your Answer	Score	Explanation
<input type="radio"/> map		
<input type="radio"/> filter		
<input type="radio"/> fold		
Total	0.00 / 3.00	

Question 6

Desired function: Take a collection of strings and return a string that is the concatenation of all the strings in the collection.

Your Answer	Score	Explanation
<input type="radio"/> map		
<input type="radio"/> filter		
<input type="radio"/> fold		
Total	0.00 / 3.00	

Question 7

Desired function: Take a collection of strings and a number n and return *how many* strings in the collection have a length that is a multiple of n .

Your Answer	Score	Explanation
<input type="radio"/> map		
<input type="radio"/> filter		

☐ fold

Total 0.00 / 3.00

Question 8

Desired function: Take a collection of strings and return a collection containing the strings in the input collection that start with a capital letter.

Your Answer

Score

Explanation

☐ map

☐ filter

☐ fold

Total 0.00 / 3.00

Question 9

[4 points total] This datatype binding and type synonym are useful for representing certain equations from algebra:

```
datatype algebra_exp = Variable of string
                    | Integer of int
                    | Decimal of real
                    | Addition of algebra_exp * algebra_exp
                    | Multiplication of algebra_exp * algebra_exp
                    | Exponent of algebra_exp * int
type equation = algebra_exp * algebra_exp
```

Which of the mathematical equations below could *not* be elegantly represented by a value of type equation

Your Answer

Score

Explanation

☐ $x + y = z$

☐ $(x + 4) + z = 7 * y$

☐ $x^3 * y^2 = z^0$

☐ $14.2 + 3 = 17.2$

☐ $x^y = z$

Total

0.00 / 4.00

Question 10

[10 points total] Here is a particular polymorphic type in ML:

```
'a * 'b -> 'b * 'a * 'a
```

For each type below, check the box if and only if the type *is* an instantiation of the type above, which means the type above is more general.

Your Answer	Score	Explanation
<input type="checkbox"/> <code>string * int -> string * int * int</code>	✓ 2.00	
<input type="checkbox"/> <code>int * string -> string * int * int</code>	✗ 0.00	
<input type="checkbox"/> <code>int * int -> int * int * int</code>	✗ 0.00	
<input type="checkbox"/> <code>{foo : int, bar : string} -> {a : string, b : int, c : int}</code>	✓ 2.00	
<input type="checkbox"/> <code>'a * 'a -> 'a * 'a * 'a</code>	✗ 0.00	
Total	4.00 / 10.00	

Question 11

[15 points total for questions 11,12,13,14, and 15 together] The next 5 questions, including this

one, are similar. Each question uses a slightly different definition of an ML signature COUNTER with this same structure definition:

```
structure NoNegativeCounter :> COUNTER =  
  struct  
  
    exception InvariantViolated  
  
    type t = int  
  
    fun newCounter i = if i <= 0 then 1 else i  
  
    fun increment i = i + 1  
  
    fun first_larger (i1,i2) =  
      if i1 <= 0 orelse i2 <= 0  
      then raise InvariantViolated  
      else (i1 - i2) > 0  
  
  end
```

In each problem, the definition of COUNTER matches the structure definition

NoNegativeCounter, but different signatures allow clients to use the structure in different ways.

You will answer the same question for each COUNTER definition by choosing the best description of what it allows clients to do.

In this question, the definition of COUNTER is:

```
signature COUNTER =  
sig  
  type t = int  
  val newCounter : int -> t  
  val increment : t -> t  
  val first_larger : t * t -> bool  
end
```

Your Answer

Score

Explanation

☐ This signature allows (some) clients to cause the NoNegativeCounter.InvariantViolated exception to be raised.

☐ This signature makes it impossible for any client to call

`NoNegativeCounter.first_larger` at all (in a way that causes any part of the body of `NoNegativeCounter.first_larger` to be evaluated).

☐ This signature makes it possible for clients to call `NoNegativeCounter.first_larger`, but never in a way that leads to the `NoNegativeCounter.InvariantViolated` exception being raised.

Total	0.00 / 3.00
-------	----------------

Question 12

In this question, the definition of `COUNTER` is:

```
signature COUNTER =  
sig  
  type t = int  
  val newCounter : int -> t  
  val first_larger : t * t -> bool  
end
```

Your Answer	Score	Explanation
-------------	-------	-------------

☐ This signature allows (some) clients to cause the `NoNegativeCounter.InvariantViolated` exception to be raised.

☐ This signature makes it impossible for any client to call `NoNegativeCounter.first_larger` at all (in a way that causes any part of the body of `NoNegativeCounter.first_larger` to be evaluated).

☐ This signature makes it possible for clients to call `NoNegativeCounter.first_larger`, but never in a way that leads to the `NoNegativeCounter.InvariantViolated` exception being raised.

Total	0.00 / 3.00
-------	----------------

Question 13

In this question, the definition of COUNTER is:

```
signature COUNTER =  
sig  
  type t  
  val newCounter : int -> int  
  val increment : t -> t  
  val first_larger : t * t -> bool  
end
```

Your Answer	Score	Explanation
<input type="radio"/> This signature allows (some) clients to cause the <code>NoNegativeCounter.InvariantViolated</code> exception to be raised.		
<input type="radio"/> This signature makes it impossible for any client to call <code>NoNegativeCounter.first_larger</code> at all (in a way that causes any part of the body of <code>NoNegativeCounter.first_larger</code> to be evaluated).		
<input type="radio"/> This signature makes it possible for clients to call <code>NoNegativeCounter.first_larger</code> , but never in a way that leads to the <code>NoNegativeCounter.InvariantViolated</code> exception being raised.		
Total	0.00 / 3.00	

Question 14

In this question, the definition of COUNTER is:

```
signature COUNTER =  
sig  
  type t
```

```
val newCounter : int -> t
val increment : t -> t
val first_larger : t * t -> bool
end
```

Your Answer	Score	Explanation
<input type="radio"/> This signature allows (some) clients to cause the <code>NoNegativeCounter.InvariantViolated</code> exception to be raised.		
<input type="radio"/> This signature makes it impossible for any client to call <code>NoNegativeCounter.first_larger</code> at all (in a way that causes any part of the body of <code>NoNegativeCounter.first_larger</code> to be evaluated).		
<input type="radio"/> This signature makes it possible for clients to call <code>NoNegativeCounter.first_larger</code> , but never in a way that leads to the <code>NoNegativeCounter.InvariantViolated</code> exception being raised.		
Total	0.00 / 3.00	

Question 15

In this question, the definition of `COUNTER` is:

```
signature COUNTER =
sig
  type t = int
  val newCounter : int -> t
  val increment : t -> t
end
```

Your Answer	Score	Explanation
<input type="radio"/> This signature allows (some) clients to cause the <code>NoNegativeCounter.InvariantViolated</code> exception to be raised.		

☐ This signature makes it impossible for any client to call `NoNegativeCounter.first_larger` at all (in a way that causes any part of the body of `NoNegativeCounter.first_larger` to be evaluated).

☐ This signature makes it possible for clients to call `NoNegativeCounter.first_larger`, but never in a way that leads to the `NoNegativeCounter.InvariantViolated` exception being raised.

Total	0.00 /
	3.00