HW 5
Functional Programming and SML

Brandon DeLuca and Weili Zhong

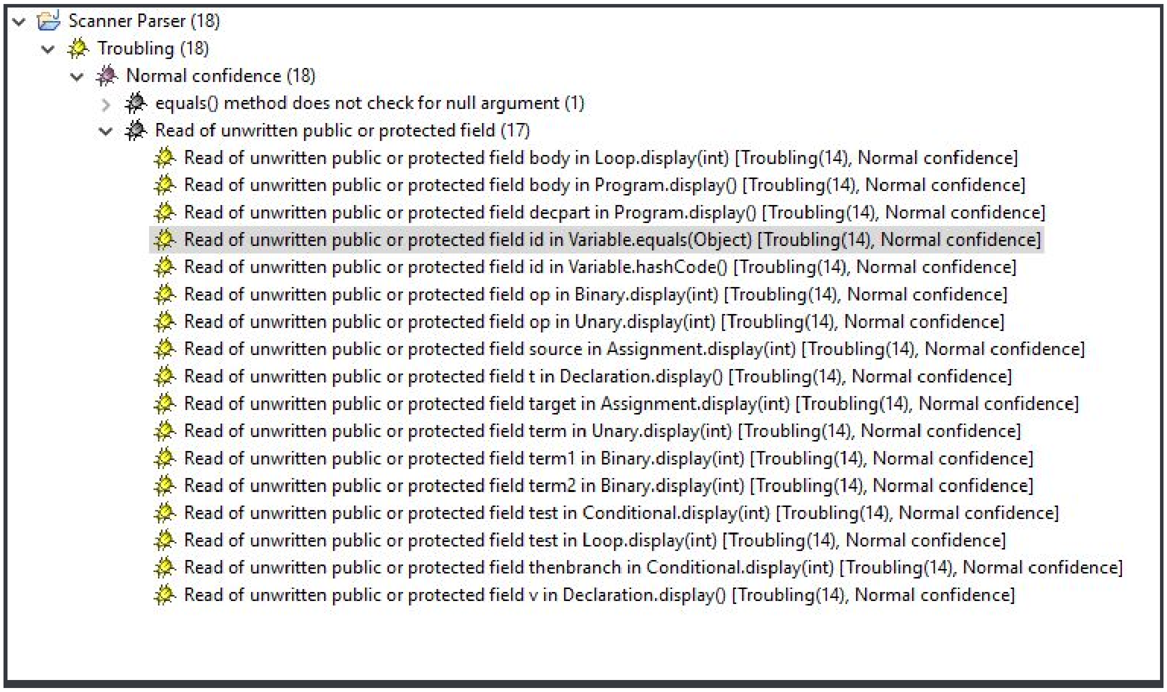
12/17/18

Problem 1: FindBug

Use FindBug on the scanner and parser code you wrote and provide the results in a readable way.

http://findbugs.sourceforge.net

FindBugs (as a project) is dead, but development continues under a new name: SpotBugs.

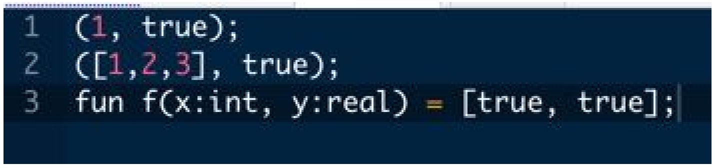


```
v Scanner Parser (18)
  v Troubling (18)
    v Normal confidence (18)
      > equals() method does not check for null argument (1)
      v Read of unwritten public or protected field (17)
          Read of unwritten public or protected field body in Loop.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field body in Program.display() [Troubling(14), Normal confidence]
          Read of unwritten public or protected field decpart in Program.display() [Troubling(14), Normal confidence]
          Read of unwritten public or protected field id in Variable.equals(Object) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field id in Variable.hashCode() [Troubling(14), Normal confidence]
          Read of unwritten public or protected field op in Binary.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field op in Unary.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field source in Assignment.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field t in Declaration.display() [Troubling(14), Normal confidence]
          Read of unwritten public or protected field target in Assignment.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field term in Unary.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field term1 in Binary.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field term2 in Binary.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field test in Conditional.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field test in Loop.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field thenbranch in Conditional.display(int) [Troubling(14), Normal confidence]
          Read of unwritten public or protected field v in Declaration.display() [Troubling(14), Normal confidence]
```

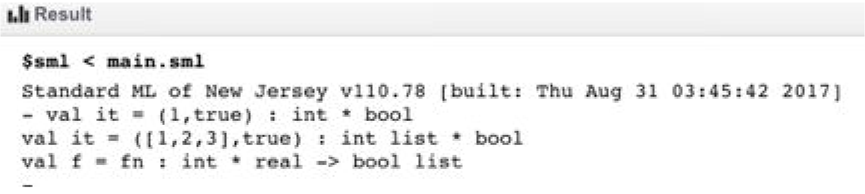
Problem 2: Types

1. What are the types of the following expressions?
   - [(1,5), (2,3), (5,6)]; → (int * int) list
   - fun f(x:real) = true; → real → bool
   - map f; → real list → bool list

2. Provide expressions of the following types:
   - int * bool → (1, true);
   - int list * bool→ ([1,2], true)
   - int * real --> bool list → fun f(x:int, y:real) = [true];



```
(1, true);
([1,2,3], true);
fun f(x:int, y:real) = [true, true];
```



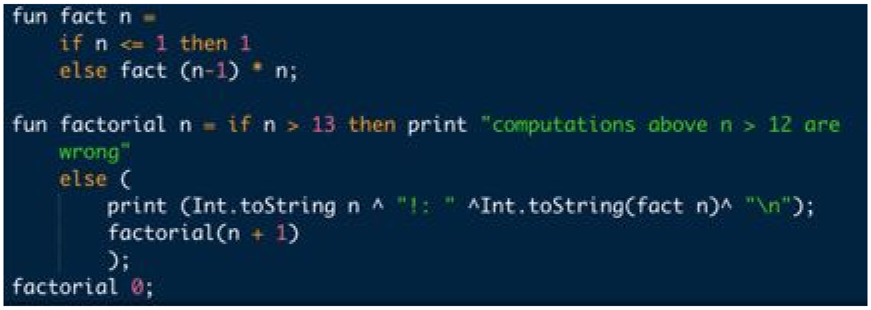
```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- val it = (1,true) : int * bool
val it = ([1,2,3],true) : int list * bool
val f = fn : int * real -> bool list
-
```

Problem 3: Factorial and Fibonacci

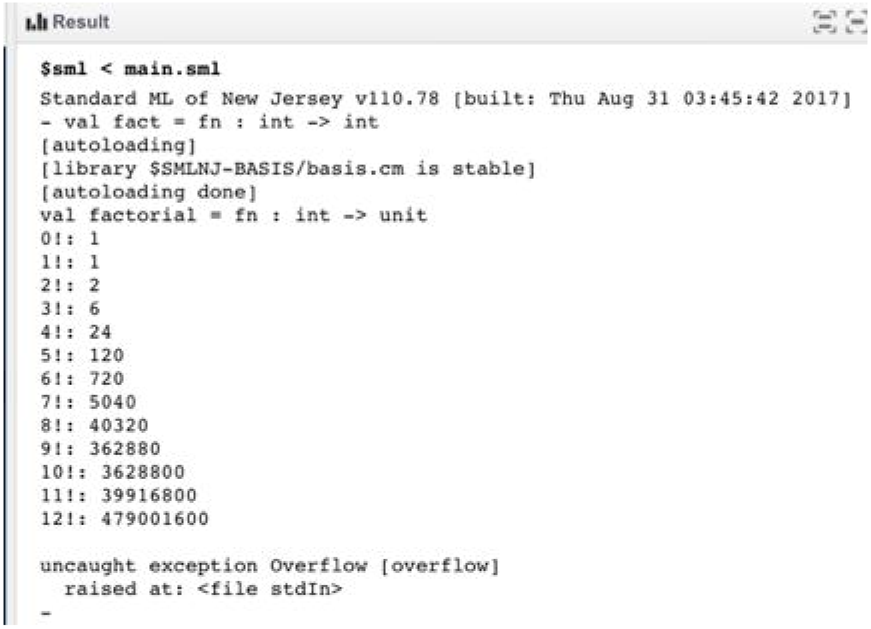
Write the factorial and fibonnaci functions using exceptions in SML.

FACTORIAL:

```
fun factaux n =
    if n <= 1 then 1
    else factaux (n-1) * n;

fun factorial n = if n > 12 then print "computations of n > 12 are inaccurate"
    else (
        print (Int.toString n ^ "! : " ^ Int.toString(factaux n) ^ "\n");
        factorial(n + 1)
        );
factorial 0;
```



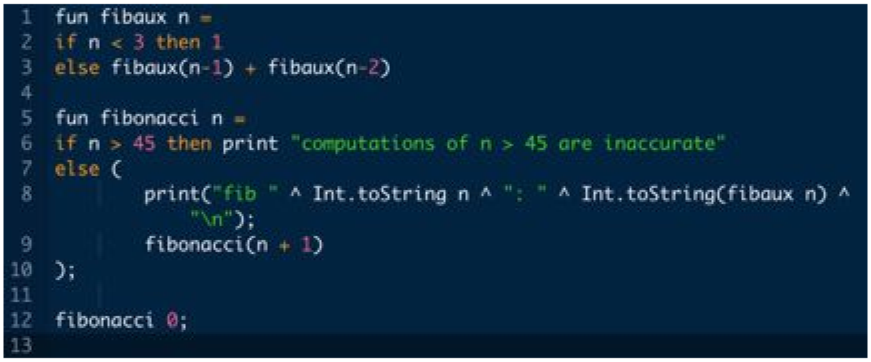
```
fun fact n =
    if n <= 1 then 1
    else fact (n-1) * n;

fun factorial n = if n > 13 then print "computations above n > 12 are wrong"
    else (
        print (Int.toString n ^ "!: " ^Int.toString(fact n)^ "\n");
        factorial(n + 1)
        );
factorial 0;
```



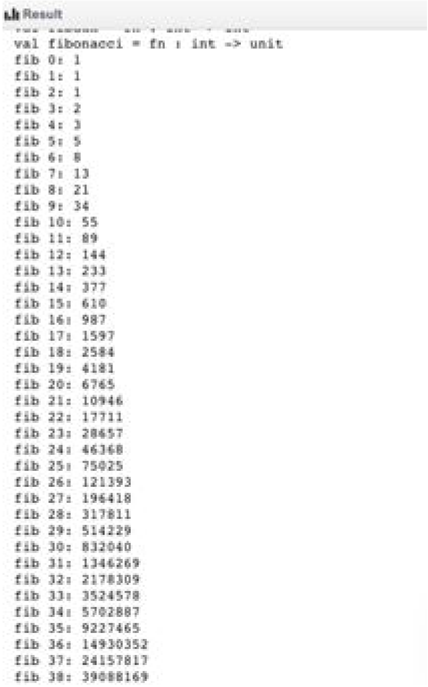
```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- val fact = fn : int -> int
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val factorial = fn : int -> unit
0!: 1
1!: 1
2!: 2
3!: 6
4!: 24
5!: 120
6!: 720
7!: 5040
8!: 40320
9!: 362880
10!: 3628800
11!: 39916800
12!: 479001600

uncaught exception Overflow [overflow]
  raised at: <file stdIn>
-
```

FIBONACCI:

```
fun fibaux n =
        if n < 3 then 1
        else fibaux(n-1) + fibaux(n-2)

fun fibonacci n =
        if n > 45 then print "computations of n > 45 are inaccurate"
        else (
              print("fib " ^ Int.toString n ^ ": " ^ Int.toString(fibaux n) ^ "\n");
              fibonacci(n + 1)
        );

fibonacci 0;
```



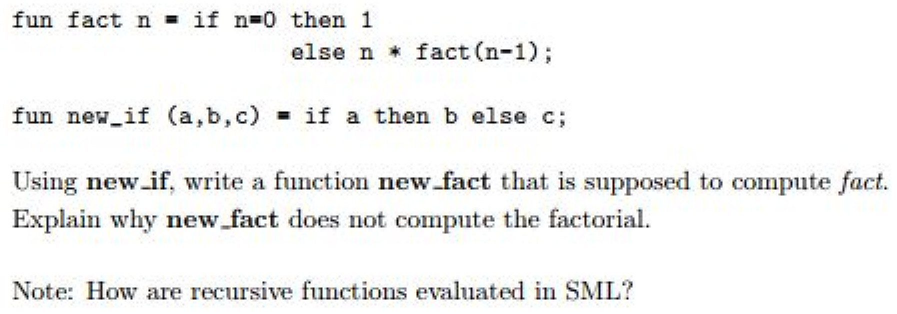
```
fun fibaux n =
if n < 3 then 1
else fibaux(n-1) + fibaux(n-2)

fun fibonacci n =
if n > 45 then print "computations of n > 45 are inaccurate"
else (
        print("fib " ^ Int.toString n ^ ": " ^ Int.toString(fibaux n) ^
            "\n");
        fibonacci(n + 1)
);

fibonacci 0;
```



```
val fibonacci = fn : int -> unit
fib 0: 1
fib 1: 1
fib 2: 1
fib 3: 2
fib 4: 3
fib 5: 5
fib 6: 8
fib 7: 13
fib 8: 21
fib 9: 34
fib 10: 55
fib 11: 89
fib 12: 144
fib 13: 233
fib 14: 377
fib 15: 610
fib 16: 987
fib 17: 1597
fib 18: 2584
fib 19: 4181
fib 20: 6765
fib 21: 10946
fib 22: 17711
fib 23: 28657
fib 24: 46368
fib 25: 75025
fib 26: 121393
fib 27: 196418
fib 28: 317811
fib 29: 514229
fib 30: 832040
fib 31: 1346269
fib 32: 2178309
fib 33: 3524578
fib 34: 5702887
fib 35: 9227465
fib 36: 14930352
fib 37: 24157817
fib 38: 39088169
```

Provide the SML code. Show evidence of testing of your functions.

Problem 4: Evaluation of Functions



```
fun fact n = if n=0 then 1
                    else n * fact(n-1);

fun new_if (a,b,c) = if a then b else c;
```

Using **new_if**, write a function **new_fact** that is supposed to compute *fact*. Explain why **new_fact** does not compute the factorial.

Note: How are recursive functions evaluated in SML?

```
fun new_if(a,b,c) = if a then b else c;
        val new_if = fn: bool * ‘a * ‘a -> ‘a

fun new_fact(n) =  new_if(n = 0, 1, n * new_fact(n-1));
        val new_fact = fn : int -> int
```

SML gives new_fact(n) with an innermost evaluation, therefore not allowing the evaluation to ever terminate, as compared to an outer most evaluation. Thus the reaching the base case when n = 0 would end the function and give the result.

Problem 5: map

What is the type of map?

It is the curried version of apply

The type is:

(‘a -> ‘b) -> ‘a list -> ‘b list

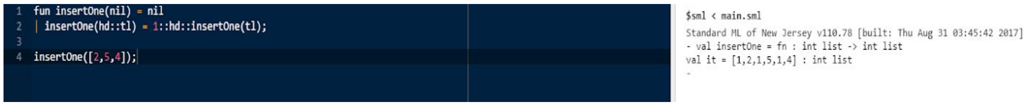
Use map to multiply by 3 each element of a list of ints.

```
map (fn x => x * 3) [1,2,3];
        -val it = [3,6,9] : int list
```

Problem 6: insertOne

Write a function that inserts 1 in front of each element of a list. It also adds 1 at the beginning of the list. For example, insertOne[5,3,4] returns [1,5,1,3,1,4]. insertOne[] returns [].

```
fun insertOne(nil) = nil
| insertOne(hd::tl) = 1::hd::insertOne(tl);

insertOne([3,5,4]);
```



```
fun insertOne(nil) = nil
| insertOne(hd::tl) = 1::hd::insertOne(tl);

insertOne([2,5,4]);
```

```
$sml < main.sml
Standard ML of New Jersey v110.78 [built: Thu Aug 31 03:45:42 2017]
- val insertOne = fn : int list -> int list
val it = [1,2,1,5,1,4] : int list
-
```

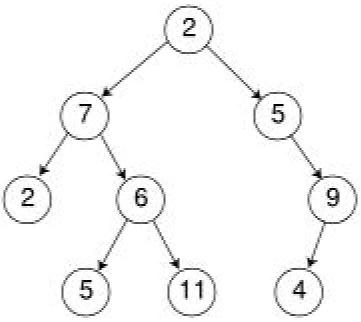
Provide the SML code. Show evidence of testing of your function.

Problem 7: SML Datatypes

This is a datatype describing the tree data structure.

datatype 'a Tree = Empty | Node of ‘a * 'a tree * 'a tree;

Using the tree datatype construct the binary tree below.



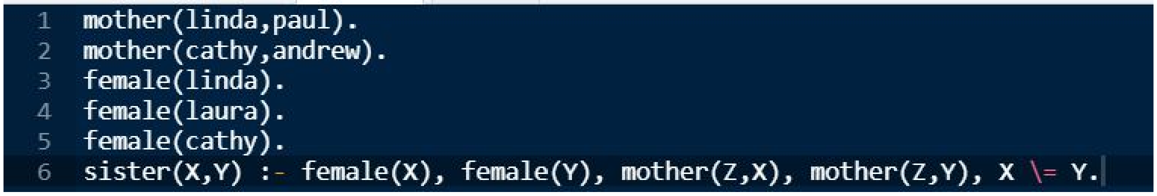
node(2,node(7,node(2,empty,empty),node(6,node(5,empty,empty),node (11,empty,empty))),node(5,empty,node(9,node(4,empty,empty),empty)));

Problem 8: Prolog

1. Let us consider the following set of facts that describe the mother predicate.

mother(linda, paul).
mother(cathy, andrew).
mother(cathy, laura)

- Define a predicate female(X) which holds iff X is a female
- Define a predicate sister(X,Y) which holds iff X and Y are sisters
- Implement female and sister in PROLOG
- Provide screenshots



```
mother(linda,paul).
mother(cathy,andrew).
female(linda).
female(laura).
female(cathy).
sister(X,Y) :- female(X), female(Y), mother(Z,X), mother(Z,Y), X \= Y.
```

2. Implement the function g such that g(x) = x+5.

```
g(X,Y) :-
        integer(X),
        Y is X+5.
```

4 ?- g(5,Y).
Y = 10.

5 ?- g(10,Y).
Y = 15.

6 ?- g(12,Y).
Y = 17