COMP 212 : Functional Programming, Spring 2015

Written Homework 01

Name: _____

Wes Email: _____

Question	Points	Score
1	4	
2	14	
3	11	
4	6	
Total:	35	

You *must* print this PDF and write your answers *neatly* by hand. You can hand the assignment in during lab or to Dan Licata's office (Exley 633).

1. **Collaboration Policy**

Read the collaboration policy is on the course website. Then, for each of the following situations, decide whether or not the students' actions are permitted by the policy. Explain your answers.

(1) (a) Eric and Amy are discussing Problem 3 over Skype. Meanwhile, Eric is writing up his solution to that problem.

Solution:

(1) (b) Brandon and Abby eat lunch (at noon) while talking about their homework, and by the end of lunch, they have covered their napkins with notes and solutions. They throw out all of the napkins and go to class from 1pm-5pm. Then, each individually writes up his or her solution.

Solution:

(1) (c) Esha and Tyler write out a solution to Problem 4 on a whiteboard. Then, they erase the whiteboard and run to the computer cluster. Sitting at opposite sides of the room, each student types up the solution.

Solution:

(1) (d) Laura is working on a problem alone on a whiteboard. She accidentally forgets to erase her solution and goes home to write it up. Later, Rob walks by, reads it, waits 4 hours, and then writes up his solution. Is Laura in violation of the policy? Is Rob?

Solution:

2. **Type Checking and Evaluation**

In this section we will explore the step-by-step reasoning of type checking and evaluation to better understand when an SML expression is well-typed, what its type is, how it will evaluate, and what its value will be. We will also do some basic analysis of the number of steps in the evaluation of an expression.

We will start with an example. Consider the expression `intToString 7` and assume that we know `intToString : int -> string`. To determine the type of this expression, we first note that the expression `7` has the type `int`. Now, using this and the fact that `intToString` has the type `int -> string` we conclude that the application of these two expressions has the type `string` since the first expression has a function type and the type of the second expression matches the type of the argument for this function.

(2) (a) Determine the type of the expression:

```
(intToString 3) ^ (intToString 6)
```

Describe your reasoning in the same manner as the example.

Solution:

(2) (b) Explain why the expression, `intToString "1"`, is not well-typed.

Solution:

(2) (c) We will now look at an example of reasoning about evaluation. Consider the expression `(intToString 7) ^ "1"` and assume that we know the application, `(intToString 7)`, evaluates to the value `"7"`. Note that the value `"1"` evaluates to itself. Using these two facts, we conclude that the whole expression evaluates to `"71"` since the `^` operator evaluates its two subexpressions and then evaluates to the concatenation of the two strings that result from these evaluations.

Determine the value that results from the evaluation of the expression:

```
intToString (4 + 4)
```

Describe your reasoning in the same manner as the example.

Solution:

(d) Recall, from Lecture 2, the following categories of expressions, each of which is strictly larger than the next:

- syntactically correct expressions
- well-typed expressions
- valuable expressions
- values

For each of the following expressions, state the **most specific** set that the expression belongs to. Briefly explain why your answer is correct.

(2) i. `f (5 div (1 - 1))` where

```
fun f (x : int) : int = 47
```

Solution:

(2) ii. `6`

Solution:

(2) iii. `1+1`

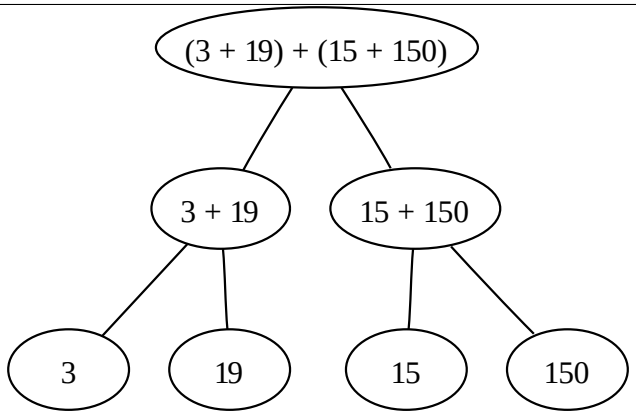
Solution:

(2) iv. `(intToString 5) + 1`

Solution:

3. **Parallel Computing** We can draw an expression, such as `(3 + 19) + (15 + 150)`, as a tree, such as where the leaves are values and the internal nodes are operations. The *work*, W, of the expression tree is the total number of internal nodes, while the *span*, S, is the number of edges along the longest path from the root to a leaf.

   Regardless of the number of processors used to evaluate a compound expression in parallel, the number of steps required to evaluate the final result must be at least as great as the span because evaluating a parent requires first evaluating its children (a *data dependency*). Also, if each of P processors performs one evaluation step in parallel during



each *time cycle*, it would require at least W/P time cycles to perform all W operations. These observations give the intuition behind *Brent's Theorem*:

Theorem 1 (Brent's Principle)**.** *If an expression, e has work W and span S, then evaluating e on a P-processor machine takes time proportional to* $\max(W/P, S)$.

(3) (a) What are the work and span for the tree in Figure 3? Use Brent's Principle to determine a lower bound on the number of steps required to evaluate this arithmetic expression on a machine with $P = 2$ processors.

Solution:

(3) (b) Describe one possible assignment of the nodes in the tree to the two processors to achieve this lower bound. In particular, for each time step, state what node each processor is evaluating. If a processor is idle during a time step state this.

Solution:

(5) (c) In the first lecture, we acted out the process of counting the number of students in the class who had heard of functional programming before this class. Following this model, describe a process involving bulk operations on collections of objects or people that occurs in your life. Analyze the work (*i.e.*, how many operations it takes overall to do the task) and the span (*i.e.*, how long it takes if you could parallelize arbitrarily much).

Solution:

4. **Interpreting Error Messages**

Download the file `hw01.sml` from the assignments page.

You can evaluate the SML declarations in this file using the command

```
use "hw01.sml";
```

at the SML REPL prompt (start smlnj in the same directory as where you saved the

file). Unfortunately, the file has some errors that must be corrected. The next tasks will guide you through the process of correcting these errors.

(2) (a) What is the first error message do you see when you evaluate the unmodified `hw01.sml` file? What caused this error and how can it be fixed?[1]

Solution:

Correct this one error in the `hw01.sml` file and load it again.

(2) (b) Now what is the first of the remaining errors? What caused this error?

Solution:

Again, correct just this one error in the `hw01.sml` file and reload it.

(2) (c) What is the final error message? What does this error message mean? How can you fix it?

[1] *Hint:* Compare the syntax of `double` and `intToString`. What is different?

Solution:

When you correct this final error and evaluate the file there should be no more error messages.