15-150 Assignment 01

Karan Sikka
ksikka@cmu.edu
Section G
1/20/12

1: Task 2.1

The magic number is 2701.

2: Task 2.2

1. This is not permitted behavior. Eric must wait 4 hours before typing up his solution.
2. This is permitted.
3. This is not permitted behavior. They must wait 4 hours before typing up his solution.
4. Laura is in violation of the policy. She was supposed to erase the board. Rob is not in violation of the policy. He could not help but read the board.

3: Task 3.1

`(intToString 3) ^ (intToString 6)` evaluates to a value of type String.
Parentheses take, precedence in SML so `intToString 3` and `intToString 6` are evaluated first. `intToString` is of type `int -> string`. Both 3 and 6 are of type int, so `intToString` is well-typed, and they return strings. Finally, the ^ (concatenate) function takes two strings as inputs, and outputs a string. The expression is well-typed since `intToString 3` and `intToString 6` evaluate to strings. Therefore, the final resulting value will be a value of type string.

4: Task 3.2

The function `intToString` takes an int as its input. However, “1” is a string. Since intToString is passed a string instead of an int, it is not well-typed.

5: Task 3.3

The expression `fact 4` is evaluated first, since it is inside parentheses. We know from the problem that `fact 4` evaluates to 24. Making the proper substitution, SML will then evaluate `intToString 24`. Using the reasoning from a previous example, this expression is well-typed and will output “24”.

6: Task 3.4

1. well-typed expressions
This expression is syntactically correct. It is also well-typed, since div is passed two ints, and f is passed an int. However it is not evaluable because it attempts 5 div 0, which is undefined, and

therefore throws an error.

2. value
4 evaluates to itself, 4.

3. evaluable expressions
1+1 is well-typed because the + function takes in two ints. It correctly evaluates to a value of 2. Since it does not evaluate to something simpler than itself, 1+1 is not a value.

4. syntactically correct expressions
This expression is not well typed because the + function is passed an int and a string, while + is only defined for two ints. However, it is syntactically correct.

7: Task 4.1

Work = 3
Span = 2
Lower Bound = 2

8: Task 4.2

Let the processors be named p1 and p2.
1: p1 will compute $3 + 19$ and p2 will compute $15 + 150$
2: p1 will compute the top node, $22 + 165$, while p2 will idle

9: Task 4.3

The task of making hamburgers is a process involving bulk operations on collections of objects. For each hamburger made, the work is proportional to the number of components of the burger, ie. buns, meat, tomatoes, pickles, onions, ketchup, mustard, etc, because each of these have to be placed into the burger. If tasks could be parallelized, each component could be fetched at the same time. Then, two of components would be placed together. Then, two of these stacks would be placed together. And so on. This is an analogous algorithm to summing in parallel. Therefore the work is related to n and the span is related to $log(n)$.

10: Task 5.1

"Error: syntax error: inserting BAR"
This error was caused because the case syntax was lacking a bar before the underscore. It was fixed by inserting one there.

11: Task 5.2

"Error: unbound variable or constructor: y"

The error is that SML doesn't recognize the variable y. This is probably a typo. The error was fixed by replacing y with n.

12: Task 5.3

"Error: unbound variable or constructor: factorial"
The error is a misspelling in the function name. Factorial was changed to fact.

13: Task 5.4

"Error: types of rules don't agree [tycon mismatch]" "Error: right-hand-side of clause doesn't agree with function result type [tycon mismatch]" The error is that 1.0 is not considered an int by SML. Changing it to 1 fixes this error.