15-150 Assignment 01

Jack Kasbeer
jkasbeer@andrew.cmu.edu
Section K
September 8, 2015

2: Course Resources & Policy

1. Magical number: 2933
2. Ethics Committment: handed in @ OH
3. Student's actions permitted?
   (a) Yes, Shaurya & Ethan's actions are permitted because they're simply discussing an example from class versus graded material.
   (b) No, Jackie & Jun's actions are not permitted, albeit their effort to burn the napkins was admirable. Regardless, students are not permitted to discuss homework solutions with each other, whether or not that discussion has a record of occurring.
   (c) The course policies do not specify exactly whether or not Mihir's actions were permissible, but they do say that the work must be entirely your own. To be safe, Mihir should email the 15-150 staff before proceeding, but for the question I would answer "yes", as my intuition would say that using a library book is not cheating.
   (d) Ashwin is not in violation of course policies unless he intentionally left his work there for Tianyu. On the other hand, Tianyu is definitely in violation of course policy as that was not her own work that she used for the homework. But the situation also doesn't specify whether or not the problem Ashwin was working on was in fact graded material... In this case, neither would be in violation.
   (e) Sri's actions are not permitted; the course policies specify that you cannot use previous semester work in your own work.

3: Types

1. `(intToString 7) ^ (intToString 9) : string`
   intToString converts both 7 and 9 to strings (as mentioned above), so the `^` operation on two strings has the type `string * string : string`, hence the expression is of type string. Now, applying parts i-iii from the handout to both `intToString 7` & `intToString 9`, we start with both of these expressions being type `string`. Continuing...
   (i) (HAT) `^ : string * string -> string`
   (ii) `intToString 7 : string` by handout (i-iii)
   (iii) `intToString 9 : string` by handout (i-iii)
   (iv) `(intToString 7) ^ (intToString 9) : string` by (HAT)

2. `intToString 2.0` is not well-typed because `intToString` requires an argument of type `int`, and 2.0 is of type `real`.

4: Evaluation

1. Informally: `intToString (fact 4)`
   We are told that `fact 4` evaluates to 24, so the expression above thus evaluates to `intToString 24`. Since this argument is of type `int`, the expression is well-typed and will evaluate to `"24"`.

2. Formally: `intToString (fact 4)`
   (i) `fact 4` $\Longrightarrow$ `24`

   (ii) `intToString (fact 4)` $\Longrightarrow$ `intToString 24`

   (iii) `intToString 24` $\Longrightarrow$ `"24"`

5: Interpreting Error Messages

1. Error message:
   `code/hw01-fix.sml:6.3 Error: syntax error: inserting  VAL`
   This error was caused by the second case of the function `isEmpty` where a preceding | is missing (bad syntax). I fixed it by simply adding a | on the line of the case of a non-empty `int list`.

2. First error in the set:
   ```
   code/hw01-fix.sml:11.44-11.50 Error: operator and operand don't agree [overload conflict]
   operator domain: real * real
   operand:         real * [int ty]
   in expression:   t - 32
   ```
   This error is caused by `t - 32` because the types of `t` and `32` don't match (`t : real` and `32 : int`). I fixed it by simply changing `32` to `32.0`.

3. Next error message:
   `code/hw01-fix.sml:18.23-18.29 Error: unbound variable or constructor: iseven`
   This error is caused by the function `iseven` because this isn't a defined function. It can be easily fixed by capitalizing the E, changing the function call to `isEven`.

4. First error of the next set:
   `code/hw01-fix.sml:23.5-23.11 Error: can't find function arguments in clause`
   This error is caused by the lack of an argument in the function's definition. This set of errors was resolved by changing the definition to `fun double (x : int) : int = 2 * x`, adding the argument `x` after the name of the function (`double`).

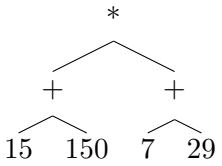
5. Final error: `code/hw01-fix.sml:30.48-30.59 Error: operator and operand don't agree [tycon mis`
   ```
   operator domain: real * real
   operand:         real * int
   in expression:   0.5 * b
   ```
   This error is caused by a type mismatch between b, h, and 0.5. We cannot multiply these numbers because b & h are of type `int` while 0.5 is of type `real`. I fixed the error by changing the function to take a tuple of two numbers of type `real` instead of type `int`. There are no more errors!

6: Specs & Functions

1. Does not satisfy the specification
2. Satisfies the specification
3. Satisfies the specification
4. Satisfies the specification
5. The specification from Task 6.3 gives the most information about the behavior of `decimal` because in addition to specifying that n shouldn't be negative, but may be zero, it communicates that an empty list will not be returned by the function.

7: Parallel Computing

1. Computation tree



2. The work of the above tree = 3; the span = 2.
3. The lower bound on the number of steps needed to evaluate this computation tree is 2 (Brent's Theorem).
4. If we assign one processor to the left addition (L), and one to the right (R), then at $time - step = 1$, $L = 15 + 150 = 165$ & $R = 7 + 29 = 36$. Then at $time - step = 2$, $L = R = 165 * 36 = 5940$. Hence, the evaluation has been done in two steps when executing in parallel with two processors.
5. If we have one digger, one sower, and one sprinkler, we can plant n trees in $time = n * (3t)$ minutes (since we have 3 processes that take t minutes each, and n trees).
6. With an infinite number of diggers, sowers, and sprinklers, it would take $3t$ minutes to plant n apple trees. With an unlimited number of resources, we can plant every single apple tree at the same time, and since each "planting" involves three dependent tasks that all take time t, it takes $t + t + t = 3t$ minutes.