

15-150 Fall 2015

Homework 01

Out: Wednesday, 2 September 2015
Due: Tuesday, 8 September 2015 at 23:59 EST

1 Introduction

Welcome to 15-150! This assignment introduces the course infrastructure and the SML runtime system, then asks some simple questions related to the first week of lectures and lab.

1.1 Getting The Assignment

The starter files for the homework assignment have been distributed through our `git` repository. To learn how to use it, read the documentation at

<http://www.cs.cmu.edu/~15150/resources/git.pdf>

If you still need help, ask a TA promptly and get started on the non-code questions.

In the first lab, you set up a clone of this repository in your AFS space. To get the files for this homework, log in to one of the UNIX servers via SSH or sit down at a cluster machine, change into your clone of the repository, and run

```
git pull
```

This should add a directory for Homework 1 to your copy of the repository, containing a copy of this PDF and some starter code in subdirectories. If this does not work for you, contact course staff immediately.

1.2 Submission

Submissions will be handled through Autolab, at

<https://autolab.cs.cmu.edu>

In preparation for submission, your `hw/01` directory should contain a file named exactly `hw01.pdf` containing your written solutions to the homework.

To submit your solutions, run `make` from the `hw/01` directory (that contains a `code` folder and a file `hw01.pdf`). This should produce a file `hw01.tar`, containing the files that should be handed in for this homework assignment. Open the Autolab web site, find the page for this assignment, and submit your `hw01.tar` file via the “Handin your work” link.

The Autolab handin script does some basic checks on your submission: making sure that the file names are correct; making sure that no files are missing; making sure that your code compiles cleanly. Note that the handin script is *not* a grading script—a timely submission that passes the handin script will be graded, but will not necessarily receive full credit. You can view the results of the handin script by clicking the number (usually either 0.0 or 1.0) corresponding to the “check” section of your latest handin on the “Handin History” page. If this number is 0.0, your submission failed the check script; if it is 1.0, it passed.

Remember that your written solutions must be submitted in PDF format—we do not accept MS Word files or other formats.

Your `hw01-fix.sml` file must contain all the code that you want to have graded for this assignment, and must compile cleanly. If you have a function that happens to be named the same as one of the required functions but does not have the required type, it will not be graded.

Warning : On this homework and all future homeworks, we reserve the right to refuse to grade or to otherwise penalize submissions that do not follow specified formatting or the instructions in the handout. If your code does not compile you may receive a zero on those sections of the homework. Please contact course staff if you have any questions. If you attempt to contact us close to the deadline, please be aware that we may not be able to respond before the deadline. If you cannot access the Autolab site, notify the course staff immediately.

1.3 Due Date

This assignment is due on Tuesday, 8 September 2015 at 23:59 EST. Remember that you may use a maximum of one late day per assignment, and that you are allowed a total of three late days for the semester.

2 Course Resources and Policy

Please make sure you have access to the various course resources. We will post important information often. You can find more information about these resources in the Tools page of the course's Web site.

We are using Web-based discussion software called Piazza for the class. You are encouraged to post questions, but please do not post anything that gives away answers or violates the academic integrity policy. If you think that your question might give away answers, you can make it a *private* question, visible only to the course staff.

Task 2.1 (1 pts). You should have received an e-mail message with instructions on signing up for Piazza. Activate your account. There is announcement there that tells you a 'magic number'. What is the number?

Task 2.2 (0 pts). Read the course and CMU policies on academic integrity on the course website at <http://www.cs.cmu.edu/~15150/policy.html>. Upon reading the policies, fill out the 15-150 Ethics Commitment handed out in lab (also attached to the end of this assignment). You must hand in a physical, hand signed copy of your completed commitment. If you did not already hand it in during lab on Wednesday September 2nd, **you must turn it in at the beginning of class on Tuesday September 8th.**

You will not receive credit on any assignment until you have completed and turned in the 15-150 Ethics Commitment.

Task 2.3 (5 pts). Refer to the collaboration policy on the course website. For each of the following situations, decide whether or not the students' actions are permitted by the policy. Explain your answers.

1. Shaurya and Ethan sit next to each other in lecture. During lecture Shaurya is confused by one of the examples that is covered. He asks Ethan about it after class, so Ethan explains it to him.
2. Jackie and Jun 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 burn the napkins and go to class from 1pm-6pm. Afterwards, they each individually writes up her solution.
3. Mihir is stuck on a homework problem and is not sure what to do. He begins to read a functional programming textbook he found at the library for help. In the book, there is an example problem that solves the problem Mihir is stuck on. He uses that example to construct his answer to the homework problem.

4. Ashwin is working on a problem alone on a whiteboard in Gates. He accidentally forgets to erase his solution and goes home to write it up. Later, Tianyu walks by, reads it, waits 4 hours, and then writes up his solution. Is Ashwin in violation of the policy? Is Tianyu?
5. Sri is working late on a tricky question and just can't figure it out. To get a hint, she looks at the staff solution to the problem from when her friend Richard took it last semester.

3 Types

In this section we will explore the step-by-step reasoning of type checking to better understand when an SML expression is well-typed and, if so, what its type is.

An application expression `e1 e2` has type `t2` if `e1` has type `t1 -> t2` and `e2` has type `t1`. In an arrow type like `t1 -> t2`, `t1` is the *argument type* and `t2` is the *result type*. Therefore, this application is well-typed if the function expression `e1` has an arrow type, and the argument expression `e2` has the correct argument type. The application `e1 e2`, then, has the corresponding result type.

Using the notation from class for type bindings, we write `e : t` to mean that `e` has type `t`. We can summarize the above *typing rule* as follows:

(APP) If `e1 : t1 -> t2` and `e2 : t1`, then `(e1 e2) : t2`.

For example, suppose `intToString` has type `int -> string`. Consider the application expression `intToString 7`. We already said that `intToString` has type `int -> string`, an arrow type with argument type `int` and result type `string`. Clearly `7` has type `int`. Since this is the correct argument type, the application `intToString 7` has the corresponding result type (`string`).

We can summarize this informal discussion as follows:

- (i) `intToString : int -> string`
- (ii) `7 : int`
- (iii) `(intToString 7) : string` by (APP)

Task 3.1 (3 pts). The infix operator `^` has type `string * string -> string`. An expression of the form `e1 ^ e2` has type `string` if `e1` has type `string` and `e2` has type `string`.

Determine the type of the expression:

`(intToString 7) ^ (intToString 9)`

Describe your reasoning in the same manner as above, first informally using English(!), then summarize using the more formal notation. If part of your reasoning exactly corresponds to that found in the example feel free to cite the correspondence rather than copying everything.

Task 3.2 (2 pts). Explain why the expression `intToString 2.0` is not well-typed.

4 Evaluation

A well-typed expression can be evaluated. If its evaluation terminates, the result is a *value*. If the expression is already a value (such as an integer numeral, or a function), it is not evaluated further. In an expression like `e1 ^ e2`, the infix concatenation operator `^` evaluates its two subexpressions (`e1` and `e2`) from left to right, then returns the string obtained by concatenating the two strings that result from these evaluations.

Here is an example: Consider the expression `(intToString 7) ^ "1"`. Assume that the application `(intToString 7)` evaluates to the value `"7"`. The expression `"1"` is already a value. So the expression `(intToString 7) ^ "1"` evaluates to `"71"`, the string built by concatenating `"7"` and `"1"`.

Using the notation from class, we write $e \implies e'$ when `e` reduces to `e'` (when an expression “reduces to” a value we may also say “evaluates to”). We can summarize the relevant facts about evaluation in this example as:

- (i) `(intToString 7) \implies "7"`
- (ii) `"1" \implies "1"`
- (iii) `(intToString 7) ^ "1" \implies "7" ^ "1"`
- (iv) `"7" ^ "1" \implies "71"`

Now we ask you to perform a similar analysis on another example. Assume that the expression `fact 4` evaluates to 24, and that `intToString` function has the usual behavior, e.g., `intToString 42` evaluates to `"42"`.

Task 4.1 (2 pts). Determine the value that results from the following expression:

`intToString (fact 4)`

Explain your reasoning informally in the same manner as above.

Task 4.2 (3 pts). Now use the \implies notation from class, as above, to express the key evaluation facts in your analysis.

5 Interpreting Error Messages

Download the file `hw01-fix.sml` from the `git` repository as described in Section 1.1.

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

```
use "hw01-fix.sml";
```

at the SML REPL prompt. Unfortunately, the file has some errors that must be corrected. The next five tasks will guide you through the process of correcting these errors.

Task 5.1 (2 pts). What error message do you see when you evaluate the unmodified `hw01-fix.sml` file? What caused this error? How can it be fixed?¹

Correct this one error in the file `hw01-fix.sml` and evaluate the file again using the same command as before.

Task 5.2 (2 pts). With the first error corrected, you will encounter a set of errors. What is the first error in this set? What caused this error? How do you fix it?²

Now also correct this second error in the file `hw01-fix.sml` and evaluate the file again.

Task 5.3 (2 pts). What is the first error message you see after correcting these first two bugs? What does this error message mean? How do you fix this error?

Fix this third error as well, and evaluate the file again.

Task 5.4 (2 pts). There is now yet another set of errors. What is the first error message? What does this error message mean? How do you fix this error?

Once again, fix the error, then re-evaluate the file.

Task 5.5 (2 pts). There should be one more error message. What is it? What caused it? How do you fix this error?

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

¹*Hint:* Compare the syntax between `isEmpty` and `isEven`

²*Hint:* Think about types.

6 Specs and Functions

Consider the following function:

```
(* decimal : int -> int list *)
fun decimal (n:int):int list =
  if n<10 then [n] else (n mod 10) :: decimal(n div 10);
```

A specification for this function has typical form

```
(* decimal : int -> int list
 * REQUIRES: . . .
 * ENSURES: . . .
 *)
```

The function *satisfies* this spec if for all values `n` of type `int` that satisfy the assumption from the requires-condition, `decimal n` evaluates as described in the ensures-condition and if the function itself makes no recursive calls that violate the requires-condition.

For each of the following specifications, say whether or not this function satisfies the specification. If not, give an example to illustrate what goes wrong. A (decimal) digit is an integer value in the range 0 - 9.

Task 6.1 (2 pts).

```
(* decimal: int -> int list
 * REQUIRES: true
 * ENSURES: decimal(n) evaluates to a non-empty list of digits
 *)
```

Task 6.2 (2 pts).

```
(* decimal: int -> int list
 * REQUIRES: n>0
 * ENSURES: decimal(n) evaluates to a non-empty list of digits
 *)
```

Task 6.3 (2 pts).

```
(* decimal: int -> int list
 * REQUIRES: n>=0
 * ENSURES: decimal(n) evaluates to a non-empty list of digits
 *)
```


Task 6.4 (2 pts).

```
(* decimal: int -> int list
  * REQUIRES:  n>=0
  * ENSURES: decimal(n) evaluates to a list of digits
  *)
```

Task 6.5 (2 pts). Which *one* of these specifications gives the *most* information about the applicative behavior of the function `decimal`? Say why, briefly.

7 Parallel Computing

In lab we showed how to draw a computation tree for an expression, whose structure reflects the order in which its sub-expressions can be evaluated. Each non-leaf node is labeled with an operator, and its children are sub-trees representing the sub-expressions to be combined by that operator. Leaf nodes are labeled with values, such as integer numerals.

Task 7.1 (2 pts). Draw the computation tree for the following expression:

$$(15 + 150) * (7 + 29)$$

We define the *work* of a computation tree to be the total number of non-leaf nodes (i.e., the number of nodes labeled with operations). The *span* of a computation tree is the number of edges along the longest path from the root to a leaf.

Task 7.2 (2 pts). What are the work and span for the above computation tree?

Suppose we have an expression whose computation tree has work W and span S . No matter how many processors are usable for parallel evaluation, the number of steps required to evaluate the expression must be at least S , because to evaluate (the expression represented by) a node we must first evaluate its children (its immediate sub-expressions), because the value at the node depends on the values of these sub-expressions; this is a *data dependency*. Also note that 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 of the W operations required to fully evaluate the expression. These observations give the intuition behind *Brent's Theorem*:

Theorem 1 (Brent's Theorem) *If an expression e evaluates to a value with work W and span S , then evaluating e on a P -processor machine requires at least $\max(W/P, S)$ steps.*

Task 7.3 (2 pts). Use Brent's Theorem to find a lower bound on the number of steps required to evaluate the computation tree for $(15 + 150) * (7 + 29)$ on a machine with $P = 2$ processors.

Task 7.4 (2 pts). Describe a possible assignment of the nodes in this computation tree to two processors that achieves this lower bound. In particular, for each time step, say what node each processor is evaluating. If a processor is idle during a time step say so.

Consider the task of planting n apple trees in a garden using diggers, sowers, and sprinklers. For each apple tree, we must first take t minutes to dig a hole for it with a

digger, then take t minutes to sow the seed using a sower, and then finally take t minutes to sprinkle water on it.

Task 7.5 (3 pts). How much *time* would it take to complete the task with one digger, one sower, and one sprinkler? Justify your answer briefly.

Task 7.6 (3 pts). If you had an infinite number of diggers, sowers, and sprinklers, how much *time* would it take to finish the task? Justify your answer briefly.

This assignment has a total of 50 points.

15-150 Ethics Commitment

Fall 2015

We understand that most of you would never consider cheating in any form. There is, however, a small minority of students for whom this is not the case. In the past, when we have caught students cheating they have often claimed that they did not understand the rules and penalties. For this reason we require that each student read, fill out and return this document *as part of Homework 1*.

Please read the 15-150 course policy and the CMU University Policy on cheating and plagiarism (both available online). Then read every line of this document and write your initials in each blank to indicate that you have read the line and agree to what it says. If you have any questions, ask a TA before finishing.

- _____ I have read, and will keep in mind, the 15-150 course policy document and the university policy on cheating.
- _____ I promise to conduct myself honestly and responsibly, within the spirit of these policies. I will not attempt to bend the rules.
- _____ If I am unsure about whether a potential action is ethically acceptable, I will consult the course staff beforehand.
- _____ I will not work with other students on the solutions to homework, other than to discuss conceptual issues not directly relevant to solution details.
If I need homework help I will ask one of the TAs or the professor.
- _____ I understand that it is cheating to hand in work produced with the help of another student, or to use solutions from a prior semester, and I can be penalized for it.
- _____ I understand that it is cheating to allow another person access to my work, and I can be penalized for it.
- _____ I will not write down a homework solution and let anyone else copy it.
- _____ I will not read or copy solutions written by another student.
- _____ I will erase any whiteboard that I write work on after using it.
- _____ I will not discuss any program or proof as I am writing it up.
- _____ I will not consult books for help with homework problems.
(Looking up conceptual material may be OK.)

- _____ I will not ask for help from students who previously took this course¹. (Again, conceptual help may be OK.)
- _____ I will not search the Internet looking for solutions. (Again, conceptual help may be OK.)
- _____ I understand that posting for help on mathoverflow or reddit (or any other similar website) is cheating, and I can be penalized for it.
- _____ I will not attempt to hack the 15-150 infrastructure in any way.
- _____ I will not copy from another student during an exam.
- _____ I will not allow another student to see my work during an exam.
- _____ If I am taking a make-up exam, I will not talk about the exam to anyone who took it before me or will take it later.
- _____ If I take the exam on time, I will not discuss its contents with anyone who has not taken the exam yet.
- _____ I understand that the penalty for cheating on *any* part of an assignment is at least a zero on the entire assignment, and may lead to failure of the class and a disciplinary report. Multiple occurrences of ethical violations can lead to suspension or expulsion from the university.
- _____ I understand that if, in a moment of weakness, I do something that I shouldn't have, the best course of action is to talk to one of the instructors immediately. If you come to us before we come to you, the punishment may be more lenient.

We're not trying to play games with you about what's allowed or disallowed, so don't try to read between the lines or twist the words in an attempt to justify questionable decisions. If you began this class yourself in a prior semester and are retaking it, do not copy old solutions (whether your own or obtained from TAs or others). Each person's work must be entirely developed from scratch, this semester.

The spirit of our policy rules should be clear. We expect you to operate with integrity, honesty and a genuine desire to learn. The best way to work is *entirely by yourself*, except for asking general questions about the programming language, about functional programming, or about proof styles; never work on homework solutions in a group; don't share your notes or your computer with others; and plan your schedule so you have plenty of time to hand in early.

¹The only exception to this rule: you are allowed to talk to walk-in tutors arranged by Academic Development, some of whom may be former 150 students; but do not ask them for detailed help on homework solutions.