15-150 Fall 2014
Homework 01

Out: Monday 25th August, 2014
Due: Monday 1st September, 2014 at 23:59 AST

1 Introduction

Welcome to 15-150! This assignment will give you a taste of the course and help you get familiar with the ideas we discussed this week. It will also give you a chance to get used to the infrastructure we'll use for the rest of the term—don't wait to ask questions.

1.1 Getting the Homework Assignment

The starter files for the homework assignment have been distributed through Autolab at

`https://autolab.cs.cmu.edu`

Select the page for the course, click on "hw01", and then "Download handout". Uncompressing the downloaded file will create the directory `hw01-handout` containing the starter files for the assignment. The directory where you will be doing your work is called `hw/01` (see below): copy the starter files there.[1]

1.2 Submitting the Homework Assignment

Submissions will be handled through Autolab, at

`https://autolab.cs.cmu.edu`

In preparation for submission, your `hw/01` directory will need to contain the file `hw01.pdf` with your written solutions and the files `pascal.sml` with your code. The starter code for this assignment contains incomplete versions of these files. You will need to complete them with the solution to the various programming tasks in the homework.

To submit your solutions, run

[1] The download and working directory have different names so that you don't accidentally overwrite your work were the starter files to be updated. **Do not do your work in the download directory!**

```
make
```

from the `hw/01` directory on any Unix machine (this include Linux and Mac). This will produce a file `hw01-handin.tgz`, containing the files to be handed in for this homework assignment. Open the Autolab web site, find the page for this assignment, and submit your `hw01-handin.tgz` file via the "Handin your work" link. If you are working on AFS, you can alternatively run

```
make submit
```

from the `hw/01` directory. That will create `hw01-handin.tgz`, containing the files to be handed in for this homework assignment and directly submit this to Autolab.

All submission will go through Autolab's "autograder". The autograder simply runs a series of tests against the reference solution. Each module has an associated number of points equal to the number of functions you need to complete. For each such function, you get 1.0 points if your code passes all tests, and 0.0 if it fails at least one test. Click on the cumulative number for a module for details. Obtaining the maximum of a module does not guarantee full credit in a task, and neither does a 0.0 translate into no points for it. In fact, the course staff will be running additional tests, reading all code, and taking into account other aspects of the submitted code such as structured comments and tests (see below), style and elegance.

To promote good programming habits, your are limited to a maximum of 5 submissions for this homework. Use them judiciously! In particular, make sure your code compiles cleanly before submitting it. Also, make sure your own test suite is sufficiently broad.

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

The SML files `pascal.sml` 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.

1.3 Due Date

This assignment is due on Monday 1st September, 2014 at 23:59 AST. Remember that there are no late days for this course and you will get 2 bonus points for every 12 hours that you submit early.

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 About page of the course Web site.

2.1 Piazza

We are using a Web-based discussion platform 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 points) You should have been signed up for Piazza. There is an announcement there that tells you a 'magic number'. What is the number?

2.2 Integrity Policy

Task 2.2 (5 points) Read 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. Kemal and Davide are discussing Problem 4 over Skype. Meanwhile, Davide is writing up the solution to that problem.

2. Thierry and Saquib 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 to 6pm. Then each individually writes up his solution.

3. Khaled and Christos write out a solution to Problem 2 on a whiteboard in the ARC. Then they erase the whiteboard and run to the computer cluster. Sitting at opposite sides of the room, each student types up the solution.

4. Iliano is working on a problem alone on a whiteboard in the CS corridor. He accidentally forgets to erase his solution and goes home to write it up. Later, Kemal walks by, reads it, waits 4 hours, and then writes up his solution. Is Iliano in violation of the policy? Is Kemal?

5. Christos is working on a tricky question shortly before the deadline and just can't figure it out. To get a hint, he looks at the staff solution to the problem from when his friend Majd took it last semester.

Task 2.3 (3 points) The course relies on a software service called MOSS to check for plagiarized code. What does it do exactly? A quick online search will bring up examples of MOSS output, possibly for languages other than SML. Take a screen shot of one of these inputs together with something that uniquely identifies you and include it into your solution. If a student is pressed for time, what is likely to be least time consuming: write his/her own code or "borrow" code from a friend and modify it in such a way that MOSS won't flag them as similar?

Task 2.4 (4 points) Majd has two midterms the same week that homework 6 is due. He is running out of time and fears he will get a bad grade. Out of desperation, he gets a hold of Nina's solution, makes a few cosmetic changes and submits it as his own. He gets caught and bears the penalty for violations of academic integrity detailed on the class web site. Other than that, Majd was a pretty good student: he earned 94.9% on all other homeworks (a solid *A* for the course).

1. What will his final percentage grade be given the penalty? What do you think it is as a letter grade?
2. What would his final percentage grade have been if he had not submitted homework 6 (thereby earning 0% for it)? What letter grade would that be?
3. In truth, he had finished a third of the homework before cheating. What would his final percentage grade be if he had submitted that? What letter grade is that?
4. Apart from the final grade in the course, what are the consequences of Majd's action?

2.3 Bonuses

Task 2.5 (2 points) Go to the class web page, read carefully the part that explains how the grades are calculated, and answer the following questions:

1. Assume you get 100% on each homework assignment in the course. How much extra credit will you receive if you submit every assignment three full days before the deadline. Compute it as a percentage of the final grade for the course, whose maximum is 100%.
2. What if you consistently get 80% in each homework assignments.

Justify your answers

3 Get SML running

Task 3.1 (5 points) Get access to an SML/NJ prompt by either installing SML/NJ locally on your machine or ssh'ing somewhere that has it installed already. Take a screen shot of the prompt running with something that *uniquely identifies you* in it.

Task 3.2 (Extra Credit). Install some other implementation of SML on your machine and take a screen shot of that. Again, make sure that it shows something that uniquely identifies you.

4 Interpreting error messages

In this part of the homework, you will be working with the file `errors.sml` that you downloaded as part of the startup code

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

```
use "errors.sml";
```

at the SML prompt. The file contains errors for you to correct. The next six tasks will guide you through the process of correcting these errors. Remember to fix only one error at a time and evaluate the `errors.sml` file after fixing each error.[2]

Task 4.1 (2 points) What error message do you see when you evaluate the `errors.sml` file without modifying it? What caused this error? How can it be fixed?

Task 4.2 (2 points) What is the next batch of errors? What caused these errors? How can they be fixed?

Task 4.3 (2 points) What error do you see after that? What caused it? How can it be fixed?

Task 4.4 (2 points) What is the next error? What caused it? How can it be fixed?

Task 4.5 (2 points) After this, you should see two errors. What are they? What caused them? How can they be fixed?

Task 4.6 (4 points) Once you have fixed these errors, you will get a bunch more errors. Two simple fixes are sufficient to make them go away. What are these errors? What caused them? What are the fixes?

When you correct these last errors and evaluate the file there should be no more error messages. Congratulations! This is your first taste of debugging SML code.

You do not need to submit the error-free code you have produced.

[2]The semicolons at the end of the declarations in `errors.sml` are not necessary. We included them to make this task easier for you.

5 Specs and Functions

Consider the following function:

```
(* binary : int -> int *)
fun binary (n: int): int =
   if n < 2
     then n
     else (n mod 2) + 10 * binary (n div 2)
```

A specification for this function has the form

```
(* binary: int -> int
 * REQUIRES:  . . .
 * ENSURES: . . .
 *)
```

The function *satisfies* this spec if for all values `n` of type `int` that satisfy the assumption from the `REQUIRES` condition, `binary(n)` evaluates to a value that satisfies the `ENSURES` condition.

For each of the following specifications, say whether or not this function satisfies the specification. If not, give a counter-example to illustrate what goes wrong.

Task 5.1 (2 points)

```
(* binary: int -> int
 * REQUIRES: true
 * ENSURES:  binary(n) evaluates to either 0 or 1
 *)
```

Task 5.2 (2 points)

```
(* binary: int -> int
 * REQUIRES: n>0
 * ENSURES:  binary(n) evaluates to a number
 *)
```

Task 5.3 (2 points)

```
(* binary: int -> int
 * REQUIRES: n>=0
 * ENSURES:  binary(n) evaluates to a non-zero number
 *)
```


Task 5.4 (2 points)

```
(* binary: int -> int
 * REQUIRES: n>=0
 * ENSURES:  binary(n) evaluates to a non-negative number
 *)
```

Task 5.5 (2 points)

```
(* binary: int -> int
 * REQUIRES: n>=0
 * ENSURES:  binary(n) evaluates to a positive number
 *)
```

Task 5.6 (2 points) Which *one* of these specifications should you choose? Say why, briefly.

6 Inductive Definitions

In this section, you will practice working with inductive definitions and proving properties by induction.

6.1 Proof Structure

Remember that every proof by induction is structured as follows:

1. The specific *technique* being employed and on what.
2. The *structure* of the proof (number of cases and what they are).
3. For each base case:
   - The statement specialized to this case ("*To show*").
   - The proof of this case.
4. For each inductive case:
   - The statement specialized to this case ("*To show*").
   - The induction hypothesis or hypotheses (*IH*).
   - The proof of this case.

Following this methodology, students have historically submitted proofs that contained fewer errors and were more likely to be correct than otherwise.

6.2 Warmup

Task 6.1 (5 points)

Using mathematical induction on n, prove the following identity.

Property 1. *For all natural numbers n,*

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

One way to make sense of this property is to think in terms of the binary representation of numbers: the number represented by the summation consists of $n + 1$ ones (and no zeros) since 2^i contributes a one in position i exactly — for example, for $n = 3$, we obtain $\sum_{i=0}^{3} 2^i = 1111$ in binary (15 in decimal). Now, adding 1 to this number changes all these ones to zeros and adds a one in front of it — for example $1111 + 1 = 10000$ in binary (16 in decimal). A one followed by zeros only is a power of two.

Now something similar happens in other representations. Take the familiar decimal system for example: $999 + 1 = 1000$. Now, $1000 = 10^3$ while $999 = 9 \times \sum_{i=0}^{2} 10^i$. What is 9? It is one less than the base of our representation — 10 here. This suggest the following general property.

Property 2. *For all natural numbers $b > 1$ and n,*

$$(b-1)\sum_{i=0}^{n} b^i = b^{n+1} - 1$$

Task 6.2 (1 points) Explain how property 1 is a special case of property 2.

Task 6.3 (Extra Credit).Prove the Property 2.

6.3 Pascal's Triangle

Pascal's triangle is a geometric arrangement of the binomial coefficients in a triangle.[3] Building Pascal's triangle is fairly easy. We do so from the top to the bottom. A row always starts with 1 and ends with 1. To construct the elements in between, add the number directly above to the left with the number directly above to the right to find the new value as shown in Figure 1.

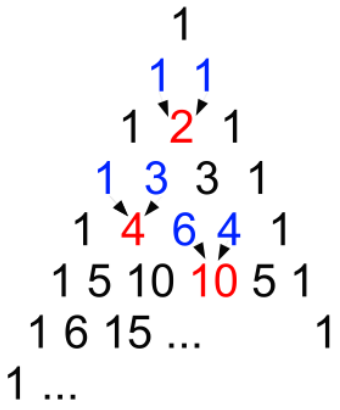

Figure 1: Building Pascal's Triangle

Each number in Pascal's triangle represents a binomial coefficient — $\binom{n}{k}$ is precisely the number at the intersection of row n and diagonal k in Pascal's triangle. We start counting rows and diagonals at 0. For instance, the coefficient $\binom{4}{2}$ is 6 as found at the intersection of row $n = 4$ and diagonal $k = 2$ as shown in Figure 2.

[3]Recall that the binomial coefficients are the number of ways that k objects can be chosen from among n objects, regardless of order. It is written $\binom{n}{k}$ and pronounced "n choose k".

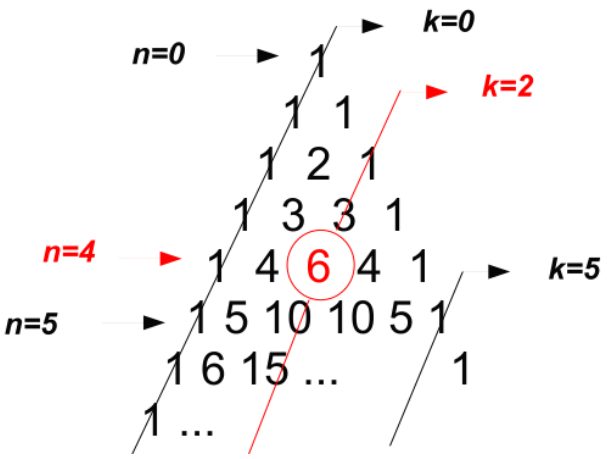

Figure 2: The value of $\binom{4}{2}$ in Pascal's Triangle

The binomial coefficients $\binom{n}{k}$ are also used in the expansion of the binomial expressions $(x+y)^n$. For an example, let's expand $(x+y)^2$:

$$\begin{array}{rcccccc} (x+y)^2 & = & \binom{2}{0}x^2 & + & \binom{2}{1}x^1y^1 & + & \binom{2}{2}x^0y^2 \\ & = & 1x^2y^0 & + & 2x^1y^1 & + & 1x^0y^2 \\ & = & x^2 & + & 2xy & + & y^2 \end{array}$$

Notice the coefficients used to expand the binomial of type $(x+y)^n$ for $n=2$ are the numbers in row 2 of Pascal's triangle as shown in Figure 3.

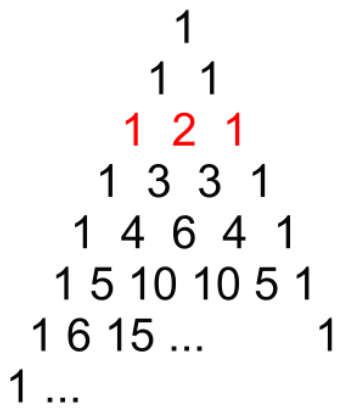

Figure 3: Binomial Coefficients in Pascal's Triangle

In general, a binomial like $(x+y)^n$ raised to any non-negative integer power n is expanded as

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k}y^k$$

and the corresponding binomial coefficients can be found at row n in Pascal's triangle.

Task 6.4 (6 points) Give an inductive definition of the binomial coefficient $\binom{n}{k}$ **based on the way Pascal's triangle is built**.

Task 6.5 (3 points) Justify (without proving it) that the following property holds based on the work that you have done so far

Property 3. *For all natural number n and all k such that $1 \leq k < n$,*

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Task 6.6 (10 points) Using property 3 and mathematical induction on n, prove the following property:

Property 4. *For all n and k such that $0 \leq k \leq n$,*

$$\sum_{j=k}^{n} \binom{j}{k} = \binom{n+1}{k+1}$$

Task 6.7 (3 points) Based on what you have done so far, give a one-line proof of the following property:

Property 5. *For all natural number $n > 0$,*

$$\sum_{k=0}^{n} \binom{n}{k} = 1 + \sum_{i=0}^{n-1} 2^i$$

6.4 Coding it up

In this and future assignments, we will be grading your programs on more than just their input-output behavior. It's not enough to have programs that happen to work: they need to clearly state what they do, have some empirical evidence that they work as advertised, and be easy for other people to read and reason about.

You must use the following five step methodology for writing functions, for *every* function you write in this assignment. Recall the five-step methodology:

1. In the first line of comments, write the name and a call template for the function.
2. In the second line of comments, specify via a `REQUIRES` clause any assumptions about the arguments passed to the function.

3. In the third line of comments, specify via an `ENSURES` clause what the function computes (what it returns).

4. Implement the function (include type annotations for the arguments and result of the function).

5. Provide test cases, generally in the format
   `val <return value> = <function> <argument value>`.

For example, here is the factorial function presented in lecture written according to the five-step methodology:

```
(*  factorial(n) = f
 *  REQUIRES:  n >= 1
 *  ENSURES: f is n!
 *)
fun factorial (1 :int): int = 1
  | factorial n = n * factorial (n-1)

(* Tests: *)
val 1 = factorial 0
val 720 = factorial 6
```

Task 6.8 (5 points) Implement the inductive definition for $\binom{n}{k}$ you gave in Task 7.4 as an SML function called `pascal` with type `pascal: int * int -> int`. For example

```
pascal(0,0)  ↪  1
pascal(4,2)  ↪  6
pascal(5,0)  ↪  1
```

You can find some starter code in file `pascal.sml`.

Task 6.9 (5 points) Using `pascal`, define the SML function `exp: int -> int` so that `exp n` evaluates to 2^n. You are not allowed to use exponentiation in your definition. You may use an auxiliary function if you need it.

7 Parallel Computing

A lot of this course has to do with parallel computation. The following exercises are a preview of things to come.

7.1 Work, Span and all that

The *computation tree* for an expression is a structure that reflects the order in which its sub-expressions can be evaluated. Each non-leaf node in the tree 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 points) Draw the computation tree for the following expression:

```
(7 * 2) < (13 - (12 div 5))
```

We define the *work* of a computation tree to be the total number of non-leaf nodes (i.e., the number of nodes labelled 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 points) 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 starting at) 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 takes 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 6 (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 points) Use Brent's Theorem to find a lower bound on the number of steps required to evaluate the computation tree for `(7 * 2) < (13 - (12 div 5))` on a machine with $P = 2$ processors.

Task 7.4 (2 points) 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.

7.2 Doing Laundry

Next, consider the task of doing n loads of laundry in a laundromat with multiple washing machines and dryers. The way you do laundry is to put a load in the washer, wait for it to be done and then transfer that load into the dryer. Once the dryer completes its cycle, that load of laundry is clean. Assume that each load takes the same time (1 hour) to wash and the same time (1 hour) to dry.

Task 7.5 (3 points) What is the work for this task? Justify your answer briefly.

Task 7.6 (3 points) If you had an unbounded number of washing machines and dryers, how quickly could you finish the task? What is the span of this task? Justify your answer briefly.

