

Homework Assignment #1

**Due: Thursday, 4 October, 2007, by 6pm**

**No submissions will be accepted after 6pm Thursday, 4 October**

---

**Silent Policy:** *A silent policy will take effect 24 hours before this assignment is due. This means that no question about this assignment will be answered, whether it is asked on the newsgroup, by email, or in person.*

**Handing in this Assignment**

*What to hand in on paper:* Please use an **unsealed** envelope, attaching the cover page provided here to the front. Note that without a properly completed and **signed** cover page, your assignment will not be marked. Put inside:

1. A printout of your code. **Write your student number on the first page of this printout.**
2. For each procedure that you write, you must describe the testing strategy that you used. This should include a table listing the test cases that you designed for your procedure, what output your procedure returned, and an explanation of why the test case and output are significant in verifying the correctness of the procedure. Read the following for a discussion of good testing practices:

<http://www.cs.toronto.edu/~gpenn/csc324/software.testing.pdf>

You must hand in this part of the assignment in your tutorial on the due date.

*What to hand in electronically:* In addition to your paper submission, you must submit your code electronically. Use this command: `submit -c csc324h -a A1 a1.sml`, from the directory where `a1.sml` lives. Type **man submit** for more information. You can also use the CDF secure website: <https://www.cdf.utoronto.ca/students>.

*Warning:* marks will be deducted for incorrect submission. Note that if the code submitted electronically differs from the code submitted on paper, we will only mark the electronically submitted version (if, in such a case, you put comments, etc. only on paper, we will mark the question as if no comments, etc. were provided).

Since we will test your code electronically, you must:

- *make certain that your code runs on CDF,*
- use the exact function names and argument(s) (including the order of arguments) specified,
- use the exact file name specified (`a1.sml`),
- not load any other files from your submitted file, and
- not display anything but the function output (no text messages to the user, fancy formatting, etc. — just what is in the assignment handout).

**Bulletin Board** Important corrections (hopefully few or none) and clarifications to the assignment will be posted on the bulletin board, linked from the course home page.

---

Homework Assignment #1  
**Cover Sheet**

---

Last Name:

First Name:

CDF login:

Email:

I have read, understood, and agree to the policies described in the Course Information handout, including the policy on collaboration.

Signature: \_\_\_\_\_

In this assignment you'll build a function, `allcombo3: 'a list * 'a list * 'a list -> ('a * 'a * 'a) list`, that builds a list of all of the triples that can be formed in order from the elements of three lists, e.g.:

```
- allcombo3 ([1,2],[3,4],[5,6]);
val it = [(1,3,5),(1,3,6),(1,4,5),(1,4,6),(2,3,5),(2,3,6),(2,4,5),(2,4,6)]
: (int * int * int) list
```

You're going to do it with higher-order functions, however. We've already seen one higher-order function in class, `map: ('a -> 'b) -> 'a list -> 'b list`, which takes a function and a list as arguments, applies the function to every element on the list, and returns a list of the results. You'll need this one below. You'll also need another one, called `reduce: ('a -> 'b -> 'b) * 'a list * 'b -> 'b`:

```
fun reduce(F, [], z) = z
  | reduce(F, (x::xs), z) = F(x)(reduce(F, xs, z));
```

`reduce` also takes a function and a list as arguments (in a tuple, however), and uses the function to synthesize a single value from all of the elements in the list. There's also a third argument that tells us what value to return for empty lists, e.g.:

```
- fun add(x)(y) = x+y;
val add = fn : int -> int -> int
- reduce(add, [1,3,5,7], 0);
val it = 16 : int
```

Here, we use `reduce` to add together all of the numbers in a list. We need an explicit `add` function because ML can't parse infix operators like `+` without any arguments. `0` is the most natural choice for the third argument because `0` added to any element yields that same element. You'll also need these functions for this assignment:

```
fun append(x)(y) = x@y;
fun cons(a)(l) = a::l;
fun tuple3(one::two::three::[]) = (one,two,three);
```

All of these functions are defined in the file `aifuns.sml`, which is available from the course web-page. Don't change this file or submit it — your code must work with the original.

**Question 1.** (2 marks) Submit this homework according to all of the guidelines.

**Question 2.** (5 marks) Define a function `distribute: 'a -> 'a list list -> 'a list list`, which returns all of the lists formed by distributing a single element over each list in a list of lists: lists, e.g.:

```
- distribute 1 [[2,3],[3,5],[4,9]];
val it = [[1,2,3],[1,3,5],[1,4,9]] : int list list
```

**Hint:** Using the functions above, you can define this with a single call to `map`.

**Question 3.** (10 marks) Define a function `eltcombos: 'a list -> 'a list list -> 'a list list`, which distributes a whole list of elements over a list of lists, e.g.:

```
- eltcombos [1,2] [[3,4],[5,6]];
val it = [[1,3,4],[1,5,6],[2,3,4],[2,5,6]] : int list list
```

**Hint:** Using the functions above, including `distribute`, you can define this with a single call to `reduce`. You'll probably need `map`, too.

**Question 4.** (10 marks) Define a function, `allcombos`: `'a list list -> 'a list list`, that builds a list of all of the lists that can be formed in order from the elements of three lists, e.g.:

```
- allcombos [[1,2],[3,4],[5,6]];
val it = [[1,3,5],[1,3,6],[1,4,5],[1,4,6],[2,3,5],[2,3,6],[2,4,5],[2,4,6]]
: int list list
```

**Hint:** Using the functions above, including `eltcombos`, you can define this with a single call to `reduce`.

**Question 5.** (5 marks) Now define `allcombo3` using `allcombos`. `allcombo3` can be defined by a single call to `map`.