Programming Language Concepts, CS2104
Tutorial 7 (15 October 2011)
(All students must prepare/attempt in advance.)

Exercise 1

Translate the following C program into VAL.

```
int pascal(int n, int k) {
    if ( n == 0 || n == k ) return 1 ;
    return pascal(n-1,k) + pascal(n-1,k-1) ;
}
```

Add VAL code that would invoke `pascal(10,5)`, so as to be able to test your program.

Exercise 2

Translate the above C function into Oz.

Exercise 3

The following Oz expression:

```
{NewArray LowIndex HighIndex Initializer}
```

evaluates to a mutable array with indices between `LowIndex` and `HighIndex`, and initialized with the value `Initializer`. Moreover, the call

```
{OS.rand}
```

returns a random integer.

Write an Oz program that creates an array of random size (make it less than 100, so we can test it in class), initializes it with random integers, sorts the array, and then prints the array on the screen (remember, there are no iterative loops in Oz, only recursion).

Exercise 4

Consider the following mystery function written in C.

```
int mystery ( int n ) {
  int aux (int n, int (*k)(int)) {
    int k1 ( int ret ) {
      int k2 ( int ret2 ) {
        return k(ret+ret2) ;
      }
      return aux(n-2,k2) ;
    }
    if ( n <= 1 ) return k(n) ;
    else return aux(n-1,k1) ;
  }
  int identity(int x) { return x ; }
  return aux(n,identity) ;
}
```

What does this function do? Notice that the function is tail-recursive, because all the function calls occur in return statements. Can you derive a systematic translation scheme for converting general recursion into tail recursion?

If you have uncovered the systematic translation scheme, apply it to the `pascal` function given in Exercise 1.