Tick exercise 1

MPhil ACS & Part III course,
Functional Programming: Implementation, Specication and Verification,
Michaelmas term, 2013.

Deadline: **4pm, 28 Oct 2013**
Assessment: marked pass or fail, this exercise is 10 % of the final course mark
Return solutions to: Kate Cisek, FS05

The exercise:

1. Give *brief* answers to each of the following.
   (a) How are lists represented in Lisp?
   (b) Explain what is meant by "functions as first-class values".
   (c) What is characteristic of *pure* functional programming languages?
2. Write a *big-step* operational semantics for the language given on Slide 41 of Lecture 3 (i.e. from *Small-step operational semantics and SML*). Use the same `val` and `exp` datatypes.
3. Consider *decompiation into logic* as presented in Lecture 4.

assembly code:		explanation of instructions:
`LOOP:`	`cmp r8,0`	compares `r8` with zero
	`je EXIT`	jumps to `EXIT`, if `r8` was zero
	`add r10,r11`	assigns: `r10 := r10 + r11`
	`add r11,r10`	assigns: `r11 := r11 + r10`
	`sub r8,2`	assigns: `r8 := r8 - 2`
	`jmp LOOP`	jumps to `LOOP`
`EXIT:`		

   (a) Write down the definition of a function that decompilation could extract from the code above.
   (b) *Briefly and informally* explain how the result of decompilation could be used to show that the assembly code above computes the Fibonacci numbers. [Hint: each iteration of the loop computes the next two Fibonacci numbers, if `r10` and `r11` are initialised to 0 and 1, respectively.]