Assignment #3

Name: _____ ID: _____

This assignment has **4** questions, for a total of **32** marks.

Recall the following acronyms: SOS (structural operational semantics), COS (contextual operational semantics), SM (small step), BG (big step), CBV (call by value), CBN (call by name).

Question 1: **Polymorphic behaviour** ..... 6 marks

Prove that for any closed term f of type $\forall\alpha.\forall\beta.\alpha \to (\alpha \uplus \beta)$ and for any closed types τ_1, τ_2 value $v : \tau_1$, we have $f\ \tau_1\ \tau_2\ v \rightsquigarrow^* inl\ v$.

Question 2: **Free theorems** ..... 6 marks

Consider type $\tau = \forall\alpha.\forall\beta.\beta \to \alpha$. Is it inhabited? If yes, show a term that inhabits τ and prove that such a term is in the term relation for τ. If not, prove that there is no term in the term relation for τ.

Question 3: **A Register Machine Language** ..... 14 marks

Formalise RML, a register machine language (conceptually, this language is an assembly language without the memory). Statically, RML programs are codebases, and codebases are lists that map memory addresses to instructions. Memory addresses are natural numbers; the list of instructions is described below, alongside their semantics.

When executed, RLM programs are run on a machine with:

- a registers file comprising 15 registers, where a register maps a register name to a value, and values are natural numbers; [2]
- a flags file comprising a 'zero' and a 'sign' flag, where a flag maps a flag name 'zero' or 'sign' to a boolean value; [2]
- a text memory containing the codebase; [2]
- a program counter, which is a natural number indicating which instruction is being executed. [1]

Instructions include:

- load a constant natural number to a register; [1]
- adding two registers together (and storing the result in the first register); [1]
- subtracting two registers together (and storing the result in the first register, and setting the 'sign' flag if the second operand is larger than the first); [1]
- comparing two registers and setting the 'zero' flag if they are the same; [1]
- jumping unconditionally to an address read by a register; [1]
- jumping to an address read by a register if the 'zero' flag is set; [1]
- jumping to an address read by a register if the 'sign' flag is set; [1]

Note that the instructions format must be generic enough to allow any register to be used in their concrete form.

When an instruction is executed, the program counter moves to the following instruction (unless a jump instruction set it to a certain address).

Question 4: **From the Register Machine to an Assembly Language** ..... 6 marks

Add a memory to the RLM language, turning it into ASM. The memory is a mapping from addresses to values. [2]

Add memory-manipulating instructions to ASM:

- read the content of a memory address into a register. Take the address where to read from, from another register; [2]
- write the content of a register to the memory address contained in another register. [2]