# Compiler Work Test

Write a simple lisp/scheme compiler using LLVM. You can write the compiler in any language you want and feel free to be creative as long as it’s possible to compile and run the program.

## *Specification/Features*

1. Function definitions, **(define)**
2. Numbers, floating point
3. Types, number, bool, list
4. Number literals
5. Lists, **(list 1 2 3), (length), (car)** and **(cdr)**
6. Special forms **(if)**
7. Builtin functions, **(println), (random) and (sqrt)**Arithmetic, **+**, **-**, **/** and **\***Comparison, **=, <, >, <=** and **>=**
8. Function calls
9. Bindings **(let (x ..))**

## **Bonus** Features

1. (lambda) and closures
2. Use SSA internally

## Non Requirements

1. Garbage collector
2. Type safety (behaviour is undefined when **car** is applied on empty list or **length** on a number and so on)
3. Tail call optimization

## Hints

1. Generate llvm-bitcode as text and compile with clang
2. Heap allocate all values (including numbers and booleans) using malloc
3. Write builtin functions in C
4. Use clang to generate llvm-bitcode for some examples to see what code you need to generate

## Requirements

1. At least the example program should compile and run

### Example Program

|  |
| --- |
| (define (sum-list lst)  (if (= (length lst) 0)  0  (+ (car lst) (sum-list (cdr lst)))))  (println (sum-list (list 1 2 3)))  (define (throw)  (let ((x (random))  (y (random)))  (<= 1.0 (sqrt (+ (\* x x) (\* y y))))))  (define (inc x)  (+ x 1))  (define (dec x)  (- x 1))  (define (pi-monte n in)  (if (= n 0)  in  (if (throw)  (pi-monte (dec n) in)  (pi-monte (dec n) (inc in) ))))    (println (\* 4 (pi-monte 100000 0))) |

When run you should get a result similar to

|  |
| --- |
| 6  314728 |