ChipotLang: An interpretted langauge for functional systems programming

Benjamin Posnick (bmp53), William Long (wl359)

There are no additional dependencies required to run ChipotLang. We have created a set of 9 sample ChipotLang programs that can be run as a demonstration of our project.

To run the tests, run `make start`. Then, upon being prompted by the main program, enter one of the following... - `test00.guac` A program that calculates the squared length of the hypotenuse of a right triangle when the legs are length 3 and 4, returning true if the value is indeed 25. This program demonstrates that integers and functions are values, functions can be applied to expressions, and if statements behave as one would expect.

- `test01.guac` A program that calculates the 12th Fibonacci number, which is 144. This program demonstrates that recursive programs can be run and that because functions are values in ChipotLang, their bodies are not evaluated until application to an expression. Recursive functions do not require the use of a `rec` keyword as in OCaml. Note: this program takes a few seconds to terminate given that the Fibonacci implementation is not memoized.
- `test02.guac` A program that modifies shared variables using lockall and unlockall. The main thread calls a joinall to ensure that threads t1 and t2 terminate before the main thread terminates. Thread t1 updates l1 and thread t2 updates l2. The expected final value of l1 is [3] and the expected final value of l2 is [4].
- `test03.guac` A program with ten threads that modify a shared variable without locking which makes the final value of the shared variable x non-deterministic. The final value of the shared variable x is printed at the end of the program.
- `test04.guac` A program with ten threads that modify a shared variable with locking which makes the final value of the shared variable x deterministic. The final value of the shared variable x is printed at the end of the program. This value should be 10 as the initial value of x is 0 and each thread increments x by 1.
- `test05.guac` A program that modifies a shared variable with with two threads that both modify the same list using locks. The final value of the list l should be either [2] or [3].
- `test06.guac` A program that runs a variation of the reader-writer problem. We have 5 writers and 5 readers in this program. Each reader prints "Write" along with the value it wrote. Each reader prints "Read" along with the value it just read. The max size of the buffer in this program is 5.
- `test07.guac` A program that swaps the value of two shared variables an odd number of times using locks and threads. The initial value of shared variable x is 0 and y is 1. The program prints the value of x followed by the value of y. Because x and y are swapped an odd number of times, the value of x should be 1 and the value of y should be 0.
- `test08.guac` A program with 25 threads that each append their unique id to a shared log. The final value of the log should be a list containing the numbers 1 - 25. Each thread's write to the log will be persisted because locking is used.
- `test09.guac` A program with 10 threads that each append (with locking) fib (n) to the log where n is between 1 and 5. The final print of this program is a list containing 10 numbers with each of these numbers being one of the first 5 numbers in the Fibonacci sequence.