RAPPORT : William Chidiac

Fonctionnalités Implémentées :

- Les 3 interpréteurs, Ocaml, Rust, Ocaml-cps, ont tous été complètements implémentés et passent tous les tests.

Organisation :

- Les trois interpréteurs implémentent principalement les mêmes fonctionnalités du langage Lua (sauf pour la partie coroutine de l'interpréteur Ocaml-cps). Il suffisait donc d'implémenter l'interpréteur Rust et d'utiliser la même structure de code dans les autres, en s'adaptant bien sûr aux structures de données qui diffèrent et à la signature des fonctions… De plus il a plus facile de traduire du Rust en Ocaml que le contraire, vu que dans ce sens-là, il n'y a pas à faire attention au life-times.

- En ce qui concerne la partie des coroutines dédier a l'interpréteurs Ocaml-cps, je n'avais compris qu'il fallait rajouter un paramètre supplémentaire, qui représenterai les coroutines courante, dans les fonctions d'interprétation. En premier temps, j'ai donc essayé d'implémenter les coroutines sans rajouter ce paramètre-là, ce qui m'a causé beaucoup de complications. Après en avoir discuté avec mes camarades, j'ai compris qu'il fallait rajouter le paramètre de coroutine courante ce qui a facilité l'implémentation.

Avis et Conclusion :

La programmation en style « passage de continuation » est une implémentation que j'ai trouvé très astucieuse qui m'a beaucoup fait pense a de la programmation monadique. De plus la découverte des coroutines m'a beaucoup intéressé, et m'a aidé à faire un rapprochement certains concept de programmation concurrente vu lors du dernier cour.