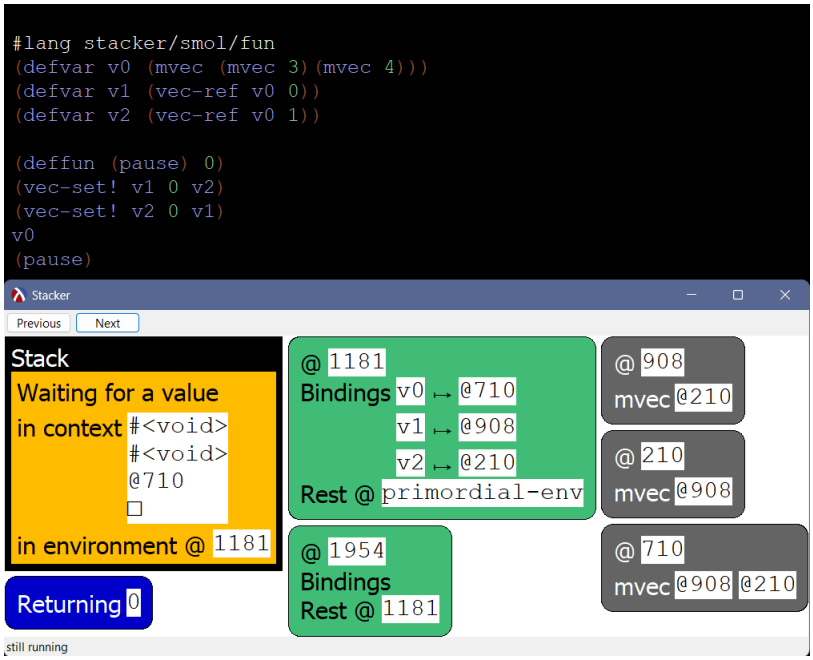
Tarea-01 Lenguajes De Programación

Mario Alejandro Castro Lerma

Septiembre 17 2023

1 De configuración a programa

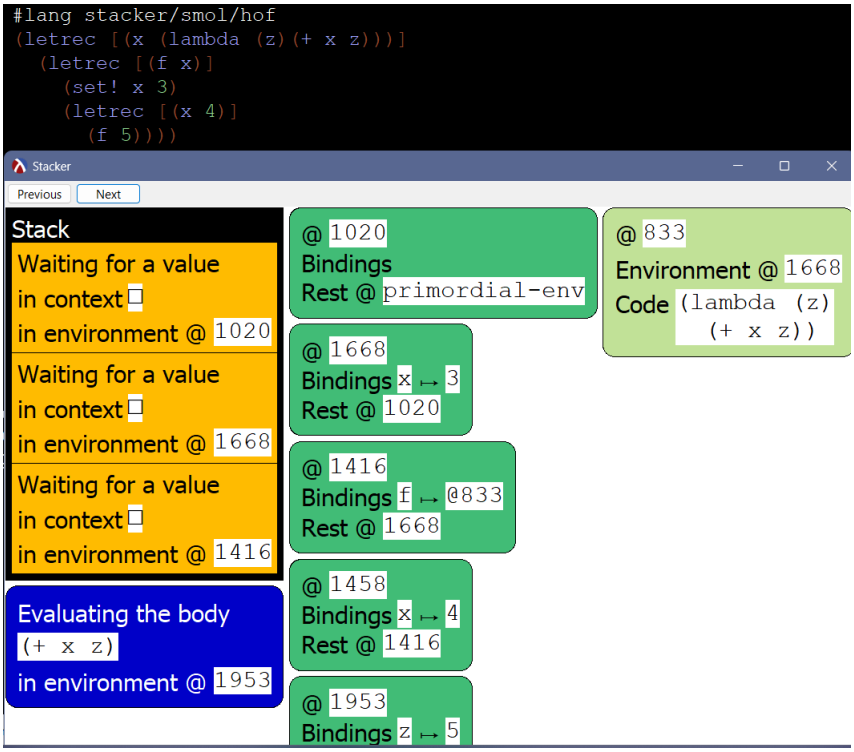
Primer problema:



Codigo:
(defvar v0 (mvec (mvec 3)(mvec 4)))
(defvar v1 (vec-ref v0 0))
(defvar v2 (vec-ref v0 1))
(deffun (pause) 0)
(vec-set! v1 0 v2)
(vec-set! v2 0 v1)
v0
(pause)

Mi razonamiento es primero observar que variables están definidas y en que bloques, primero defino las que están en el bloque superior las cuales en este caso son v0, v1, v2.
Una vez definidas las variables del bloque superior me fijo en que contienen estas, en esta caso tienen direcciones de memoria y estas direcciones apuntan a vectores, por lo tanto podemos saber que estas variables contienen vectores y que v0 contiene 2 vectores. Entonces, antes de empezar a asignar las direcciones de los vectores y demás, también podemos ver que existe otro bloque en el programa, en este caso es una función común sin definiciones dentro, en este caso es el pause que se nos pidió agregar.
Ahora empezamos con las asignaciones, inicialmente intente asignar directamente las direcciones y utilizar multiples vec-set! sin embargo estas daban error, despues intente definir las variables sin vectores y solamente usar set! para asignarles despues vectores pero esto tampoco tuvo exito, despues unos compañeros me dieron la idea de utilizar vec-ref, lo cual no habia pensado en utilizar, con esto ya solo fue necesario probar multiples casos hasta llegar a la conclusion.

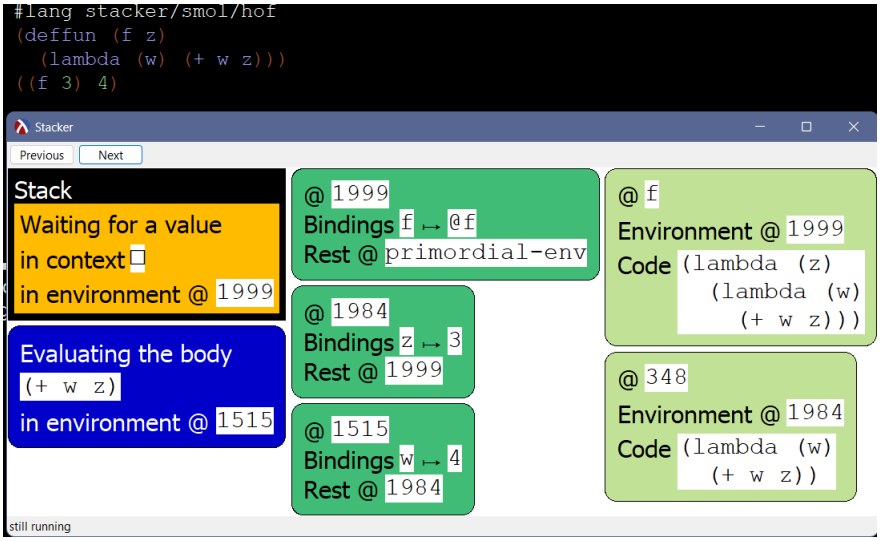
Segundo problema:



Codigo:
lang stacker/smol/hof
(letrec [(x (lambda (z)(+ x z)))]
(letrec [(f x)]
(set! x 3)
(letrec [(x 4)]
(f 5))))
Esta fue la función que más trabajo me costo, mi razonamiento fue primero ver que habían múltiples bloques donde habían múltiples variables definidas, esto me asusto inicialmente pero una vez observando bien se puede ver que habían varios entornos con sus propias variables definidas y estas no eran funciones por que en hof las funciones aparecen como lambdas del lado derecho, por esto supuse que eran let, además de que los vimos en el transcurso de la semana.
Cuando iba a utilizar let un amigo me dijo que iba a necesitar letrec así que con esto empecé a experimentar definiendo primero las variables directamente a como estaban en el ejercicio, después me di cuenta que había un problema y era que lambda estaba definida en un bloque superior a f, sin embargo f era de un bloque inferior y esto me detuvo unas horas.
Finalmente iba a rendirme pero antes de esto pregunte a algunos amigos y ellos me dijeron que revisara las notas de la clase, una vez habiendo hecho esto, me di

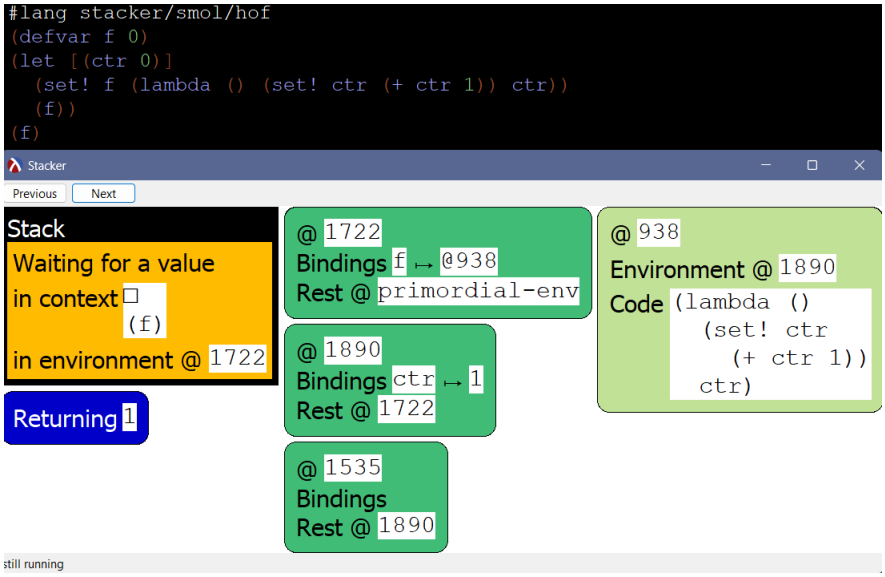
cuenta que había olvidado por completo la función set! y lo que hacia especiales a los letrec y es que las variables definidas dentro puedan referenciar a si mismas, por lo tanto con esto puede terminar la función.

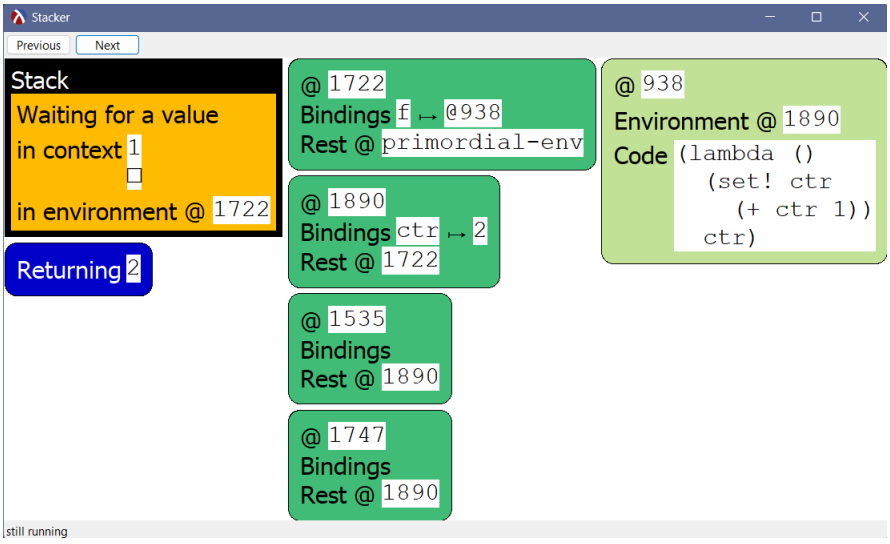
Tercer problema:



Código:
lang stacker/smol/hof
(deffun (f z)
(lambda (w) (+ w z)))
((f 3) 4)
En este problema mi razonamiento fue ver incialmente donde estaban definidas las variables, en esta caso tenemos una funcion en el bloque superior con el nombre f, el cual dentor tiene una funcion lambda donde suma w x.
El problema aqui era que yo no recordaba que lambda recordaba los valores dados anteriormente, de forma que estuve un tiempo sufriendo hasta que tuve que irme a las notas. Una vez recordando esto, pude terminar el problema.

Cuarto problema:





Código:
lang stacker/smol/hof
(defvar f 0)
(let [(ctr 0)]
(set! f (lambda () (set! ctr (+ ctr 1)) ctr))
(f))
(f)

Mi razonamiento en este problema fue primero definir una variable f y esta tuviera una lambda dentro. El problema es que lambda estaba definida en otro bloque, así que desde aquí supuse que necesitaba un set!.
Luego vi que en el nuevo ambiente era necesario definir una variable llamada ctr, este ambiente decidí hacerlo con set!, en este definí ctr y dentro cambie f por la dirección de lambda, con esto ya cumplía la mitad del problema, ahora solo era definir que hacia lambda.
Podemos ver que ctr aumenta 1 así que decidí simplemente hacer eso y regresar el valor de ctr, como esto sucedía 2 veces entonces llame a la f 2 veces, sin embargo una tenia que ser dentro de let por que si no set! regresaba null.

2 Objetos

OBJETOS BÁSICOS
1. Describe qué hace el programa
Inicialmente recibe un valor init el cual es guardado en lambda y la direccion de lambda es regresada.
Despues la función lambda recibe en una variable m 1 de 3 cosas: "inc" el cual suma 1 a el valor "init", "get" el cual regresa el valor de "init" y si no se recibe ninguna de estas 2 entonces regresa -1730.
2. ¿Qué clases hemos codificado?
"o-state-2"
3. ¿Qué objetos hemos codificado?
"init"
4. ¿Qué atributos hemos codificado?
"init" y "m"
5. ¿Qué métodos hemos codificado?
"inc" y "get"
6. Describe cómo se implementaron las clases.
Se definio una funcion llamada o-state-2 la cual contiene los objetos, metodos y atributos.
7. Describe cómo se implementaron los objetos.
Después de recibir un valor inicial, regresa la dirección de lambda la cual contiene lo necesario para operar con el objeto.
8. Describe cómo se implementaron los atributos.
Son las variables que se utilizan dentor del objeto
9. Describe cómo se implementaron los métodos.
Utilizando ifs para verificar la llamada de un metodo.

ATRIBUTOS ESTÁTICOS
1. Describe qué hace el programa
Inicia un contador desde 0, cada vez que se llame con "inc" aumenta la cantidad dada inicialmente "amount" en 1, a su vez, cada vez que sea llame cualquier funcion del objeto, se aumenta en 1 el contador "counter"
2. ¿Cuántos objetos son creados?
2
3. ¿Cuál es el valor del amount de o-1 al final de la ejecución del programa?
5
4. ¿Cuál es el valor del amount de o-2 al final de la ejecución del programa?
8
5. ¿Cuál es el valor del counter de o-1 al final de la ejecución del programa?
0
6. ¿Cuál es el valor del counter de o-2 al final de la ejecución del programa?
2