COMP301: Problem Set 10

Problem 1.

- **Call By Value:** Value of `a` doesn't change. Evaluates to `7`.
- **Call By Reference:** Content of `a` gets modified. Evaluates to `5`.

Problem 2.1.

```
let a = 12 in let f = proc(x) begin x; x; x end in (f begin set a = -(a, -13); a end)
```

Initial Environment: `ρ = [i = 0, v = 1, x = 2]`

Call By Name

After `x` is evaluated

```
Env => [x = 5][a = 3]ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (num-val 38)
4: (procedure x <<begin x; x; x end>> [a = 3]ρ)
5: (thunk <<begin set a = -(a, -13); a end>> [f = 4][a = 3]ρ)
```

After `a` is evaluated

```
Env => [f = 4][a = 3]ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (num-val 12)
4: (procedure x <<begin x; x; x end>> [a = 3]ρ)
5: (thunk <<begin set a = -(a, -13); a end>> [f = 4][a = 3]ρ)
```

Call By Reference

After `x` is evaluated

```
Env => [x = 5][a = 3]ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (num-val 25)
4: (procedure x <<begin x; x; x end>> [a = 3]ρ)
5: (num-val 25)
```

After `a` is evaluated

```
Env => [f = 4][a = 3]ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (num-val 25)
4: (procedure x <<begin x; x; x end>> [a = 3]ρ)
5: (thunk <<begin set a = -(a, -13); a end>> [f = 4][a = 3]ρ)
```

Problem 2.2.

```
letrec infinite-loop(c) = (infinite-loop -(c, 1)) in let f = proc(x) begin -(2, 1); 10 end in (f (infinite-loop 99))
```

Initial Environment: `ρ = [i = 0, v = 1, x = 2]`

If we ran the above program in IREF, the program would enter an infinite loop. When calling either by **call by name** or **call by need** the environment and state of the store are the same. *The thunk stays frozen and never gets thawed.*

When calling `f`

```
Env => [f = 3](extend-env-rec infinite-loop c <<(infinite-loop -(c, 1))>>)ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (procedure x <<begin -(2, 1); 10 end>> (extend-env-rec infinite-loop c <<(infinite-loop -(c, 1))>>)ρ)
```

Inside the body of `f`

```
Env => [x = 4](extend-env-rec infinite-loop c <<(infinite-loop -(c, 1))>>)ρ

Store
0: (num-val 1)
1: (num-val 5)
2: (num-val 10)
3: (procedure x <<begin -(2, 1); 10 end>> (extend-env-rec infinite-loop c <<(infinite-loop -(c, 1))>>)ρ)
4: (thunk <<(infinite-loop 99)>> [f = 3](extend-env-rec infinite-loop c <<(infinite-loop -(c, 1))>>)ρ)
```