PL Homework 2 - 2019-1583 김태곤

1. typeless 언어의 장단점

타입이 없으면 코드의 양을 줄이고, 좀 더 빠르게 코드를 완성시킬 수 있다.
타입이 없으니 다시 정의할 필요 없이 기존 코드를 더 빨리 수정할 수 있다.
하지만 type checking이 없어 다양한 데이터 타입을 사용할 때 실수하기 쉬우며
추적하기 힘든 버그가 발생할 수 있다. 또한 다른 사용자가 코드를 보고 이해하기 힘들 수 있고,
호환되는 데이터 타입에서만 연산이 수행되도록 런타임 체크가 많이 일어나므로
정적 타입 언어에 비해 성능이 많이 떨어진다.

2. static scoping을 사용하는 경우 sub1, sub2, sub3 에서 보이는 모든 변수와 해당변수가 선언된 프로그램 (JavaScript)

sub1 : a - sub1, y - sub1, z - sub1, x - main

sub2 : a - sub2, b - sub2, z - sub2, y - sub1, x - main

sub3 : a - sub3, x - sub3, w - sub3, y - main, z - main

3. Python

sub1 : a=7 (sub1), y=9 (sub1), z=11 (sub1), x=1 (main)

sub2 : a=13 (sub2) x=15 (sub2) w=17 (sub2) y=3 (main) z=3 (main)

sub3 : a=19 (sub3) b=21 (sub3) z=23 (sub3) x=15 (sub2) w=17 (sub2) y=3 (main)

4. C

a: (a,b,c) - m $\xrightarrow{fun1}$ a - m, (b,c,d) - 1 $\xrightarrow{fun2}$ a - m, b - 1 (c,d,e) - 2 $\xrightarrow{fun3}$ a - m, b - 1, c - 2, (def) - 3

a - main, b - fun1, c - fun2, (d,e,f) - fun3

b: (a,b,c) - m $\xrightarrow{fun1}$ a - m, (b,c,d) - 1 $\xrightarrow{fun3}$ a - m, (b,c) - 1, (d,e,f) - 3

a - main, (b,c) - fun1, (d,e,f) - fun3

c: (a,b,c) - m $\xrightarrow{fun2}$ (a,b) - m, (c,d,e) - 2 $\xrightarrow{fun3}$ (a,b) - m, c - 2, (d,e,f) - 3 $\xrightarrow{fun1}$ a - m, (b,c,d) - 1, (e,f) - 3

a - main, (b,c,d) - fun1, (e,f) - fun3

d: (a,b,c) - m $\xrightarrow{fun3}$ (a,b,c) - m, (d,e,f) - 3 $\xrightarrow{fun1}$ a - m, (b,c,d) - 1 (e,f) - 3

a - main, (b,c,d) - fun1, (e,f) - fun3