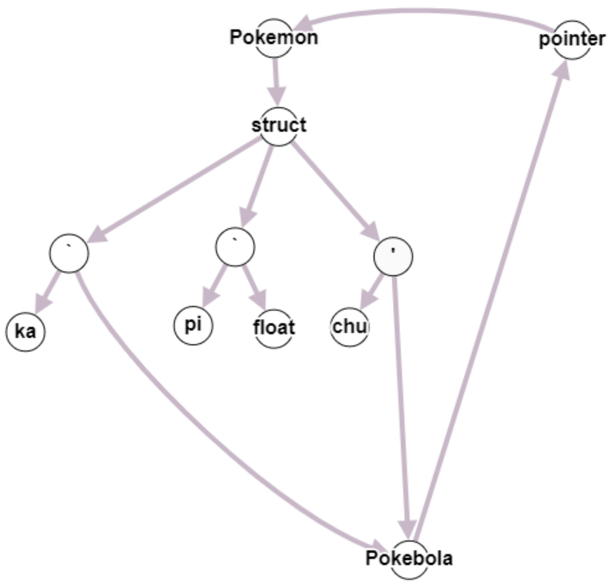
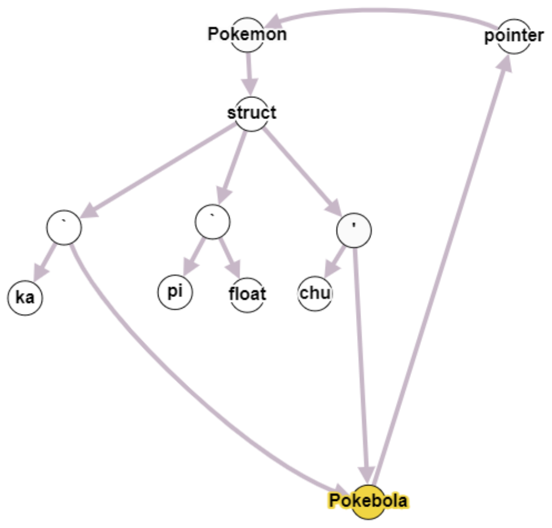
Examen 2

1.a)

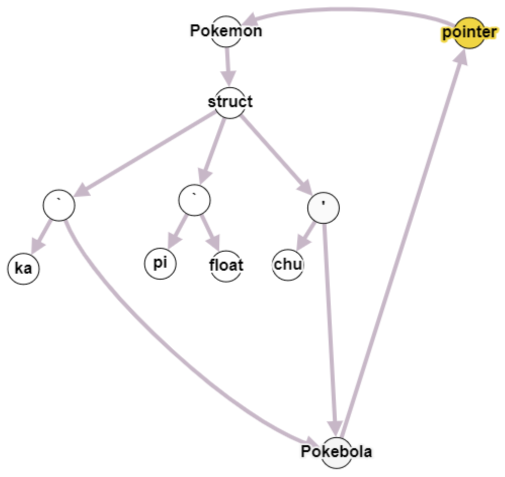


1.b)

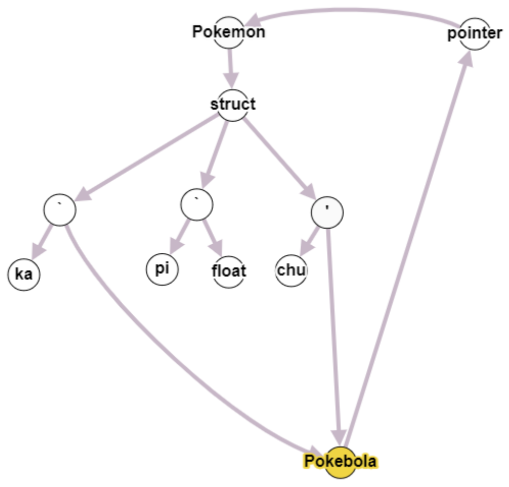
a: pokebola



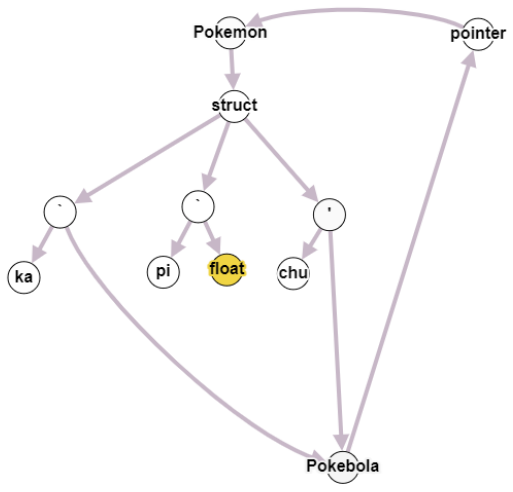
b: *pokemon



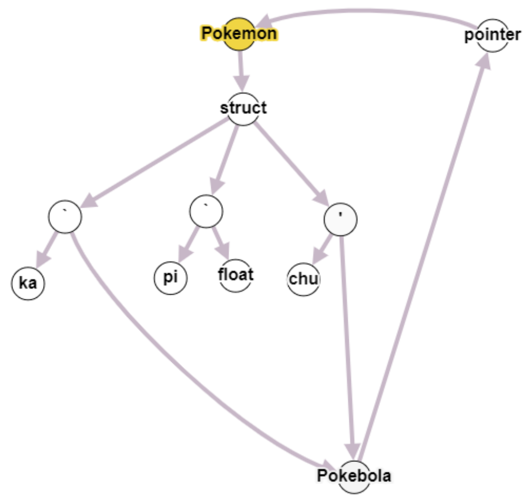
c: pokebola



d: float



e: pokemon



2.a)

```
PUSH 0
LVALUE s
ASSIGN
PUSH 0
LVALUE i
ASSIGN
condition: RVALUE i
PUSH 10
LT
GOTRUE do
GOFALSE end
do: RVALUE s
RVALUE i
RVALUE i
MUL
PUSH 2
DIV
ADD
LVALUE s
ASSIGN
RVALUE i
PUSH 1
ADD
LVALUE i
ASSIGN
GOTO condition
end: EXIT
```

2.b)

```
        s := 0
        i := 0
L:      if i >= 10 goto E
        t1 := i*i
        t2 := t1/2
        s = s + t1
        i = i + 1
        goto L
E:
```

3.a)

```
E -> E1 + E2    {
                        if (E1.type == INT && E2.type == INT) {
                                E.type = INT
                        } else {
                                E.type = ERROR
                        }
                }

| E1 /\ E2      {
                        if (E1.type == BOOL && E2.type == BOOL) {
                                E.type = BOOL
                        } else {
                                E.type = ERROR
                        }
                }

| E1 < E2       {
                        if (E1.type == INT && E2.type == INT) {
                                E.type = BOOL
                        } else {
                                E.type = ERROR
                        }
                }

| E1 ?: E2      {
                        if (E1.type == NULL) {
                                E.type = E2.type
                        } else {
                                E.type = E1.type
                        }
                }
```

```
| E1 !!         {
                     if (E1.type != NULL) {
                          E.type = E1.type
                     } else {
                          E.type = ERROR
                     }
                }

 | ( E1 )       {
                    E.type = E1.type
                }

 | num          {
                    E.type = INT
                }

 | true         {
                    E.type = BOOL
                }

 | false        {
                    E.type = BOOL
                }

 | null         {
                    E.type = NULL
                }
```

3.b) Imagen del repo llamada: derivationTree.png

3.c)

```
S -> repeatWhen E lt S1 gt S2 {
                                 if (E.type != INT || S1.type != VOID || S2.type != VOID) {
                                      S.type = ERROR
                                  }
                                 else {
                                     S.type = VOID
                                 }
                              }
```

4) match(cmap(f,x), if(null(x), [], concat(f(head(x)), cmap(f, tail(x)))))

Expresion	**Tipo**	**Sustitución**
f	γ	
x	ρ	
cmap	β	
cmap(f,x)	ω	β = γ × ρ -> ω
x	ρ	
null	list(α1) → bool	
null(x)	bool	ρ = list(α1)
[]	list(α2)	
x	list(α1)	
head	list(α3) → α3	
head(x)	α3	α1 = α3
f	γ	
f(head(x))	φ	γ = α3 -> φ
x	list(a3)	
tail	list(α4) → list(α4)	
tail(x)	list(α4)	α3 = α4
f	α3 -> φ	
cmap	α3 -> φ × list(α4) -> ω	
cmap(f, tail(x))	ω	
concat	list(α5) × list(α5) → list(α5)	
concat(f(head(x)), cmap(f, tail(x)))	list(α5)	φ = list(α5), ω=list(α5)
if	bool × α6 × α6 → α6	
if(null(x), [], concat(f(head(x)), cmap(f, tail(x))))	list(α2)	α6=list(α2), α5=α6

match(cmap(f,x), if(null(x), [], concat(f(head(x)), cmap(f, tail(x)))))	list(α6)	α7 = list(α6)

S = {

$\beta = \gamma \times \rho \to \omega$

$\rho = list(\alpha 1)$

$\alpha 1 = \alpha 3$

$\gamma = \alpha 3 \to \varphi$

$\alpha 3 = \alpha 4$

$\varphi = list(\alpha 5)$

$\omega = list(\alpha 5)$

$\alpha 6 = list(\alpha 2)$

$\alpha 5 = \alpha 6$

$\alpha 7 = list(\alpha 6)$

}

cmap :: ∀α,β: α -> list(β) × list(α) -> list(β)