2)

Class StaticChecker:

def visitProgram(self, ctx, o):

o = [{}]

Self.visit(ctx.exp, o)

def visitAssoc(self, ctx, o):

if ctx.key in o[0].keys():

return False

if type(ctx.value) is type(IntLit()):

return True

else:

o = [{}] + o

self.visit(ctx.value, o)

o.pop()

return True

def visitArray(self, ctx, o):

for assoc in ctx.assocs:

error = self.visit(assoc, o)

if not error:

raise DuplicateKey(ctx)

def visitIntLit(self, ctx, o):

pass

3)

iconst\_1

istore 2

Label1:

iload 2

iload 4

if\_cmpge Label2

iload 2

iload 4

irem

if\_ne Label3

goto Label4

Label3:

iload 6

iload 2

iadd

istore 3

Label4:

iload 2

iconst\_1

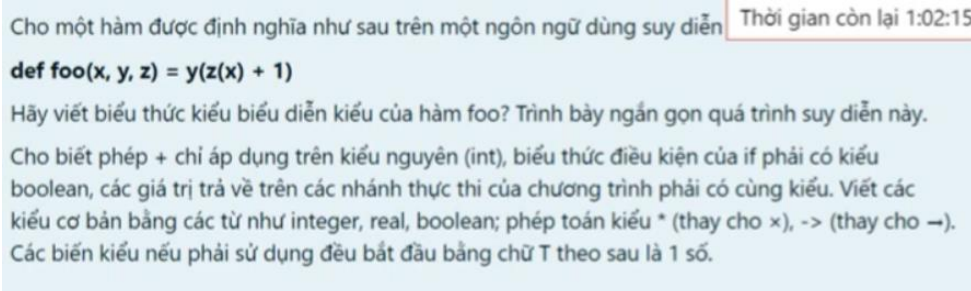
iadd

istore 2

goto Label1

Label2:

4)



Đặt kiểu hàm foo là T1 -> T2 với T1 là kiểu input, T2 là kiểu output

Hàm foo có 3 parameter nên T1 = T3\*T4\*T5 với T3 là kiểu x, T4 là kiểu của y, T5 là kiểu của z

Đặt kiểu của hàm y là T6 -> T7, suy ra T4 = T6 -> T7

Đặt kiểu của hàm z là T8 -> T9, suy ra T5 = T8 -> T9

Vì z(x) nên T8 = T3

Vì phép + chỉ áp dụng trên số nguyên nên Z(x) trả về kiểu nguyên, suy ra T9 = Int

Suy kiểu của hàm z là T3 -> Int

Vì z(x) + 1 có kiểu int nên T6 = Int

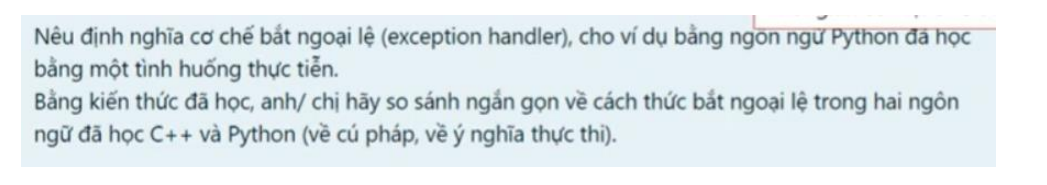
Từ trên, ta suy ra kiểu của x là T3, kiểu của y là Int -> T7, kiểu của z là T3 -> Int

Vì y(z(x) + 1) trả về kiểu T7, suy ra T2 = T7

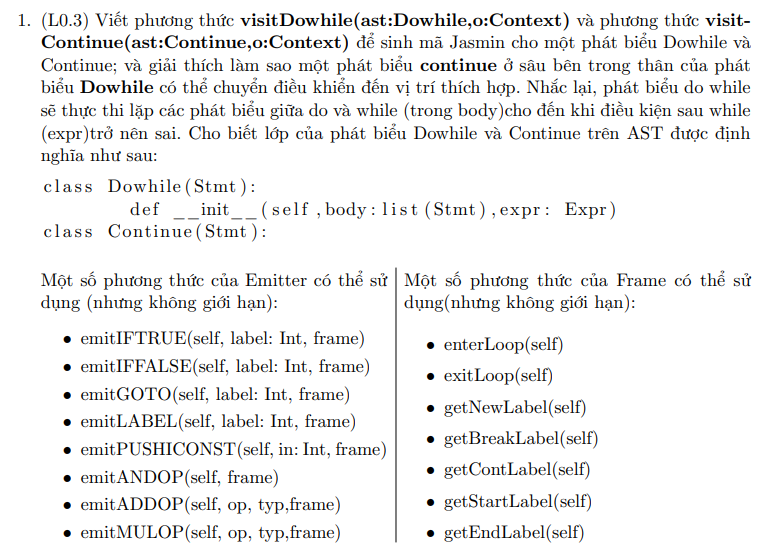
Ta suy ra, T1 = T3 \* (Int -> T2) \* (T3 -> Int)

Suy ra , hàm foo có kiểu là (T1 \* (Int -> T2) \* (T1 -> Int)) -> T2

5)



1)



def visitDowhile(self, ast, o):

o.frame.enterLoop()

start = o.frame.getContinueLabel()

end = o.frame.getBreakLabel()

o.emitLabel(start, o.frame)

for i in ast.body:

self.visit(i, o)

self.visit(ctx.expr, o)

self.emit.emitIFTRUE(start, o.frame)

self.emit.emitLabel(end, o.frame)

def visitContinue(self, ast, o):

start = o.frame.getContinueLabel()

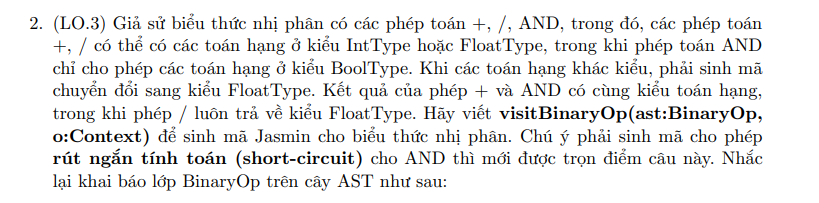
self.emit.emitGOTO(start, o.frame)

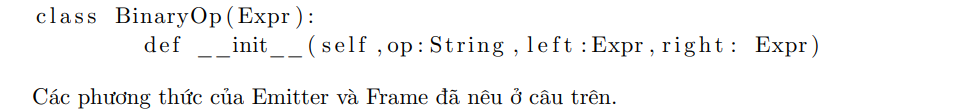
def visitBreak(self, ast, o):

end = o.frame.getBreakLabel()

o.emitGOTO(end, o.frame)

2)





def visitBinaryOp(self, ast, o):

if ast.op == “+”:

val1, typ1 = self.visit(ast.left, o)

val2, typ2 = self.visit(ast.right, o)

rt = typ1

if typ1 != typ2:

rt = FloatType()

if typ1 is IntType():

val1 = val1 + self.emit.emitI2F(o.frame)

if typ2 is IntType():

val2 = val2 + self.emit.emitI2F(o.frame)

return val1 + val2 + self.emit.emitADDOP(“+”, rt, o.frame), rt

elif ast.op == “/”:

val1, typ1 = self.visit(ast.left, o)

val2, typ2 = self.visit(ast.right, o)

rt = FloatType()

if typ1 != typ2:

if typ1 is IntType():

val1 = val1 + self.emit.emitI2F(o.frame)

if typ2 is IntType():

val2 = val2 + self.emit.emitI2F(o.frame)

return val1 + val2 + self.emit.emitMULOP(“/”, rt, o.frame), rt

else: # “AND”

val1, typ1 = self.visit(ast.left, o)

first = o.frame.getNewLabel()

second = o.frame.getNewLabel()

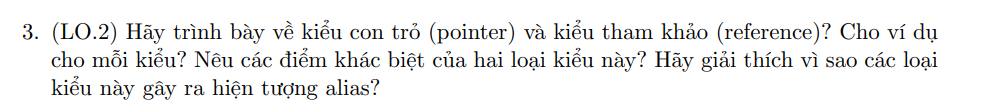
val1 = val1 + self.emit.emitIFFALSE(first, o.frame)

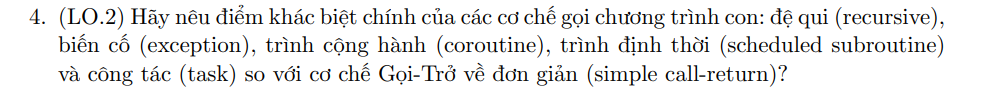
val2, typ2 = self.visit(ast.right, o)

val2 = val2 + self.emit.emitANDOP(o.frame) + self.emit.emitGOTO(second, o.frame) + self.emit.emitLABEL(first, o.frame)

val0 = self.emit.emitPUSHCONST(0, o.frame) + self.emit.emitLABEL(second, o.frame)

return val1 + val2 + val0, BoolType()





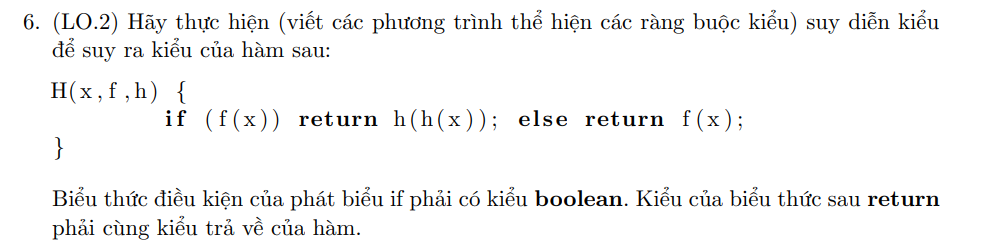
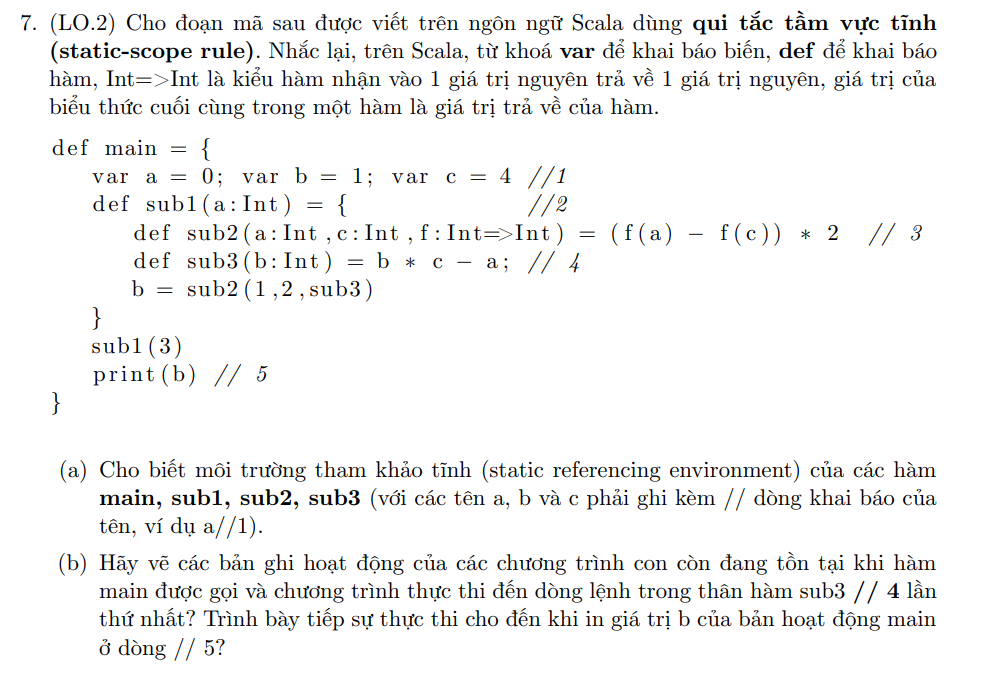
Simple call-return không thể gọi đệ quy với chương trình con đệ quy

Simple call-return cần explicit call-site, exception có thể không cần explicit call-site, cần sự kiện nào xảy ra cần chương trình con cụ thể

Simple call-return so với coroutines, coroutine có thể trả về quyền kiểm soát cho chương trình con khác mà không cần đến kết thúc chương trình

Simple call-return so với task, task cho phép nhiều chương trình con chạy cùng lúc với nhau, nhưng đòi hỏi các cơ chế cao để đảm bảo không bị deadlock, race condition và liên lạc giữa các chương trình con

Simple call-return so với trình định thời, trình định thời cần có bộ định thời để biết thời gian cũng như độ ưu tiên của chương trình

 a)

b)

sub1(3)

a//2 = 3

sub2(1,2,sub3)

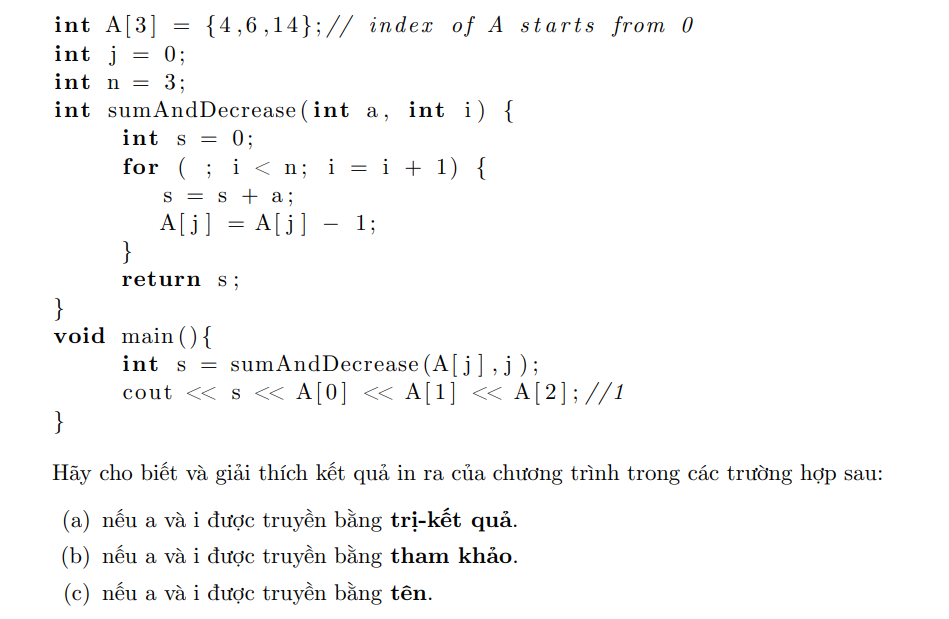
a//2 = 3

(sub3(1) – sub3(3)) \* 2

1 \* 4 – 3

3 \* 4 – 3

-16



1. Trị + Kết quả

s = 4 + 4 + 4

4, 5, 13

1. Tham khảo

s = 4 + 3 + 3

3, 5, 13

1. Name

s = 4 + 6 + 14

3, 5, 13

a=new ob(2)

foo(a):

a.set(3)

foo\_2(a):

a=new Ob(4)

foo(a)

foo\_2(a)

print(a.eval())

a = 3 => pass by value

a = 4 => pass by reference

void main()

{

int x = 5;

foo (x,x);

print (x);

}

void foo (int a, int b)

{

a = 2 \* b + 1;

b = a - 1;

a = 3 \* a - b;

}

Pass by value: Không thay đổi x = 5

Pass by reference:

X = 2 \* x + 1 = 11

X = 11 – 1 = 10

X = 3 \* 10 – 10 = 20

Pass by value – result:

a = 2 \* 5 + 1

b = 11 – 1

a = 3\*11 – 10

x = 10 or 23 ?

// Gán giá trị val1 vào id

// EnterLoop -> Lấy 2 label Continue + Break

. Label: Continue

// Lấy id với val2

// emitREOP(“<”, IntType(), o.frame)

// emitIFFALSE(Break, o.frame)

// self.visit(stmt, o.frame) Cái body

// Lấy giá trị id và val3, thực hiện phép cộng

// Gán lại vào id

// Goto Continue

.Label: Break