Question 1:

Assume that

* class Id in AST is declared with field name in str type.
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Object is passed to the parameter o of visitId has 3 fields:
  + Field frame keeps object Frame.
  + Field sym of the argument keeps a list of Symbol which has three fields: name (str type), mtype (Type type) and value (Val type). The Val type has two concrete classes: Index with field value in int type and CName with field value in str type. An Index object keeps the index of the variable while a CName keeps the name of the class name (used for global variable). The first element of sym contains the identifier which belongs to the innermost referencing environment while the last element of sym contains one in the outermost referencing environment (global).
  + Field isLeft in boolean type indicates the identifier in the left (isLeft true) or in the right (isLeft false).
* The method visitId must return a pair of jasmin code to read or write value of the identifier and the type of the identifier (one object of a subclass of class Type)

Based on the above assumption, write method visitId(self,ctx,o) of visitor CodeGeneration? Your code is at line 230.

**Answer**:

def visitId(self, ctx, o):

for i in o.sym:

if i.name == ctx.name:

if o.isLeft == False:

if isinstance(i.value.value, int):

return self.emit.emitREADVAR(i.name, i.mtype, i.value.value, o.frame), i.mtype

return self.emit.emitGETSTATIC(i.value.value + '/' + i.name, i.mtype, o.frame), i.mtype

else:

if isinstance(i.value.value, int):

return self.emit.emitWRITEVAR(i.name, i.mtype, i.value.value, o.frame), i.mtype

return self.emit.emitPUTSTATIC(i.value.value + '/' + i.name, i.mtype, o.frame), i.mtype

Question 2:

Assume that

* class Assign(Stmt) in AST is declared with field lhs and rhs in Expr type. The types of the left hand side and right hand side are the same.
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Object is passed to the parameter o of visitId has 2 fields:
  + Field frame keeps object Frame.
  + Field sym of the argument keeps a list of Symbol which has three fields: name (str type), mtype (Type type) and value (Val type). The Val type has two concrete classes: Index with field value in int type and CName with field value in str type. An Index object keeps the index of the variable while a CName keeps the name of the class name (used for global variable). The first element of sym contains the identifier which belongs to the innermost referencing environment while the last element of sym contains one in the outermost referencing environment (global).
* When visiting the expression in the left hand side or the right hand side of the assignment statement, object Access must be passed to parameter o where Access has 3 fields:
  + frame and sym are similar to the object passed to parameter o
  + Field isLeft in boolean type indicates the identifier in the left (isLeft true) or in the right (isLeft false).
* The method visitAssign must print out the code of the assignment statement (use method printout of Emitter)

Based on the above assumption, write method visitAssign(self,ctx,o) of visitor CodeGeneration? Your code is at line 230.

**Answer**:

def visitAssign(self, ctx, o):

left\_access = Access(o.frame, o.sym, True)

right\_access = Access(o.frame, o.sym, False)

rhs = ctx.rhs.accept(self, right\_access)

lhs = ctx.lhs.accept(self, left\_access)

return self.emit.printout(rhs[0] + lhs[0])

Question 3:

Assume that

* class If(Stmt) in AST is declared with fields expr in Expr type; tstmt and estmt in Stmt type. In case the if statement has no else, the estmt gets None value.
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Object is passed to the parameter o of visitId has 2 fields:
  + Field frame keeps object Frame.
  + Field sym of the argument keeps a list of Symbol which has three fields: name (str type), mtype (Type type) and value (Val type). The Val type has two concrete classes: Index with field value in int type and CName with field value in str type. An Index object keeps the index of the variable while a CName keeps the name of the class name (used for global variable). The first element of sym contains the identifier which belongs to the innermost referencing environment while the last element of sym contains one in the outermost referencing environment (global).
* When visiting the expression of the if statement, object Access must be passed to parameter o where Access has 3 fields:
  + frame and sym are similar to the object passed to parameter o
  + Field isLeft in boolean type indicates the identifier in the left (isLeft true) or in the right (isLeft false).
* The method visitIf must print out the code of the if statement (use method printout of Emitter)

Based on the above assumption, write method visitIf(self,ctx,o) of visitor CodeGeneration? Your code is at line 230.

**Answer**:

def visitIf(self,ctx,o):

if\_exprCode, if\_exprType = ctx.expr.accept(self, Access(o.frame, o.sym, False))

if ctx.estmt:

labelElse = o.frame.getNewLabel()

labelExit = o.frame.getNewLabel()

self.emit.printout(if\_exprCode)

self.emit.printout(self.emit.emitIFFALSE(labelElse, o.frame))

ctx.tstmt.accept(self, o)

self.emit.printout(self.emit.emitGOTO(labelExit, o.frame))

self.emit.printout(self.emit.emitLABEL(labelElse, o.frame))

ctx.estmt.accept(self, o)

self.emit.printout(self.emit.emitLABEL(labelExit, o.frame))

else:

labelExit = o.frame.getNewLabel()

self.emit.printout(if\_exprCode)

self.emit.printout(self.emit.emitIFFALSE(labelExit, o.frame))

ctx.tstmt.accept(self, o)

self.emit.printout(self.emit.emitLABEL(labelExit, o.frame))

Question 4:

Assume that

* class While(Stmt) in AST is declared with fields expr in Expr type; stmt in Stmt type. \
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Object is passed to the parameter o of visitId has 2 fields:
  + Field frame keeps object Frame.
  + Field sym of the argument keeps a list of Symbol which has three fields: name (str type), mtype (Type type) and value (Val type). The Val type has two concrete classes: Index with field value in int type and CName with field value in str type. An Index object keeps the index of the variable while a CName keeps the name of the class name (used for global variable). The first element of sym contains the identifier which belongs to the innermost referencing environment while the last element of sym contains one in the outermost referencing environment (global).
* When visiting the expression of the if statement, object Access must be passed to parameter o where Access has 3 fields:
  + frame and sym are similar to the object passed to parameter o
  + Field isLeft in boolean type indicates the identifier in the left (isLeft true) or in the right (isLeft false).
* The method visitWhile must print out the code of the while statement (use method printout of Emitter).Note generating labels for break and continue statements inside the while statement

Based on the above assumption, write method visitWhile(self,ctx,o) of visitor CodeGeneration? Your code is at line 265.

**Answer**:

def visitWhile(self,ctx,o):

o.frame.enterLoop()

labelExit = o.frame.getBreakLabel()

self.emit.printout(self.emit.emitLABEL(o.frame.getContinueLabel(), o.frame))

if\_exprCode, if\_exprType = ctx.expr.accept(self, Access(o.frame, o.sym, False))

self.emit.printout(if\_exprCode)

self.emit.printout(self.emit.emitIFFALSE(labelExit, o.frame))

ctx.stmt.accept(self, o)

self.emit.printout(self.emit.emitGOTO(o.frame.getContinueLabel(), o.frame))

self.emit.printout(self.emit.emitLABEL(labelExit, o.frame))

o.frame.exitLoop()

Question 5:

Assume that

* class FuncDecl in AST is declared with field name in str type, param in List[VarDecl], returnType in Type type and body in Tuple[List[Type],List[Stmt]] where VarDecl has 2 fields: name in str type and typ in Type type.
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Parameter o of method visitFuncDecl is passed an object whose field sym keeps a list of Symbol corresponding to the declarations visited.
* When visiting the declarations and statements in parameter and body, the method visitFuncDecl must pass object SubBody which has two fields: frame containing the object Frame of this function declaration and sym containing the list of Symbol corresponding to the declarations from global (in o.sym) and the declaraions in this function in reverse order.
* The method visitFuncDecl must print out a directive declarations (use method printout(str) of Emitter), visit declarations and statement code and returns an object of Symbol which has field name in str type, mtype in Type type and value in Val type. The mtype field must contain a MType object and the value field must contain a CName object whose value contains self.className.

Based on the above assumption, write method visitFuncDecl(self,ctx,o) of visitor CodeGeneration? Your code is at line 85.

Remind that class Type has subclasses: IntType, FloatType, VoidType, StringType, ArrayType, MType which has field partype in List[Type] and field rettype in Type.

**Answer**:

def visitFuncDecl(self,ctx,o):

paraType = list(map(lambda x: x.typ, ctx.param))

funcType = MType(paraType, ctx.returnType)

self.emit.printout(self.emit.emitMETHOD(ctx.name, funcType, True))

message = SubBody(Frame(ctx.name, ctx.returnType), o.sym.copy())

message.frame.enterScope(True)

paraSymbolList = list(map(lambda x: x.accept(self, message), ctx.param))

message.sym[:0] = paraSymbolList[::-1]

varSymbolList = list(map(lambda x: x.accept(self, message), ctx.body[0]))

message.sym[:0] = varSymbolList[::-1]

self.emit.printout(self.emit.emitLABEL(message.frame.getStartLabel(), message.frame))

list(map(lambda b: b.accept(self, message), ctx.body[1]))

self.emit.printout(self.emit.emitLABEL(message.frame.getEndLabel(), message.frame))

self.emit.printout(self.emit.emitENDMETHOD(message.frame))

return Symbol(ctx.name, funcType, CName(self.className))

Question 6:

Assume that

* class BinExpr in AST is declared with field op in str type, e1 and e2 in Expr type. op can be '&&' or '||' which can accept their operands in BoolType and the result type is also BoolType.
* The visitor CodeGeneration has field emit keeping an object of Emitter
* Object Frame is kept in field frame of the argument passed to parameter o of visitBinExpr
* The method visitBinExpr must return a pair of jasmin code of a binary expression and the type of the result (BoolType). Note that the boolean expression must be short-circuit evaluated.

Based on the above assumption, write method visitBinExpr(self,ctx,o) of visitor CodeGeneration? Your code is at line 160.

Remind that class Type has subclasses: IntType, FloatType, VoidType, BoolType, StringType, ArrayType, MType.

**Answer**:

def visitBinExpr(self, ctx, o):

ast = ctx

frame = o.frame

(code, \_) = ast.e1.accept(self, o)

code += self.emit.emitDUP(frame)

if ast.op == "&&":

false\_label = frame.getNewLabel()

code += self.emit.emitIFFALSE(false\_label, frame)

(\_code, \_) = ast.e2.accept(self, o)

code += \_code

code += self.emit.emitANDOP(frame)

end\_bin = frame.getNewLabel()

code += self.emit.emitGOTO(end\_bin, frame)

code += self.emit.emitLABEL(false\_label, frame)

code += self.emit.emitPOP(frame)

code += self.emit.emitPUSHICONST("false", frame)

code += self.emit.emitLABEL(end\_bin, frame)

else:

true\_label = frame.getNewLabel()

code += self.emit.emitIFTRUE(true\_label, frame)

(\_code, \_) = ast.e2.accept(self, o)

code += \_code

code += self.emit.emitOROP(frame)

end\_bin = frame.getNewLabel()

code += self.emit.emitGOTO(end\_bin, frame)

code += self.emit.emitLABEL(true\_label, frame)

code += self.emit.emitPOP(frame)

code += self.emit.emitPUSHICONST("true", frame)

code += self.emit.emitLABEL(end\_bin, frame)

return code, BoolType()