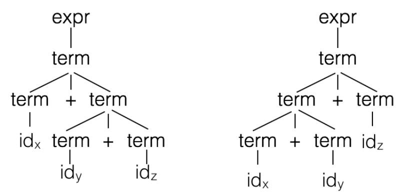
CS 558 Programming Languages Fall 2023 – Practice Mid-term Exam Sample Solutions

1.(a) Yes, the grammar is ambiguous, because there are expressions with two distinct parse trees. For example, the expression `x+y+z` has these two parse trees:



(b) No such program can exist. The only way evaluation order can affect the result is because of side-effects, and the only side-effect in the language is assignment. The operands of `'+'` and `'*'` are always terms; evaluating terms never causes an assignment. So it can't matter which order the operands are evaluated.

2.(a,b,e) Marked below. For (e), the idea is that to achieve dynamic scope, we simply set the initial environment for evaluating functions to the *caller's* environment (rather than the lexical environment of the callee, which for this language is empty).

```
case class Program(fdefs:List[FunDef], body:Expr) {}        // ((fdefs) body)
case class FunDef(name:String, param:String, body:Expr) {}  // (name param body)
sealed abstract class Expr {}
case class Num(n:Integer) extends Expr                       // n
case class Var(x:String) extends Expr                        // x
case class Add(l:Expr,r:Expr) extends Expr                   // (+ e e)
case class Apply(f:String,e:Expr) extends Expr               // (@ f e)
case class Let(x:String,e:Expr,b:Expr) extends Expr          // (let x e b)

type Env = Map[String,Int]
val emptyEnv : Env = Map[String,Int]()

def interp(p:Program): Int = {
  def interpE(env:Env,e:Expr) : Int = e match {
    case Num(n) => n
    case Var(x) => env.getOrElse(x, throw InterpException("undefined variable:" + x))
    case Add(l,r) => interpE(env,l) + interpE(env,r)
    case Let(x,e,b) => {
      val v = interpE(env,e)
      interpE(env + (x->v),b)
    }
    case Apply(f,e) => {
      val v = interpE(env,e)
      for (fdef <- p.fdefs)
        if (fdef.name == f)   env
          return interpE(emptyEnv + (fdef.param->v),fdef.body)
      throw InterpException("undefined function:" + f)
    }
  }
  interpE(emptyEnv,p.body)
}
```

(c) Under static scoping, the use of `y` in the body of `g` is free, so interpreting the program raises the exception `InterException("undefined variable:y")`.

(d) Under dynamic scoping, the binding of `y` is visible inside `g`, so the program evaluates to 21 + 21 = 42.

3. (a) The result of `h` is the result of `g`, which is the value of `b`. But `b` was never initialized and its values is therefore undefined.

(b) In the generated code, the stack location of `a` in `f` is the same as the location of local variable `b` in `g`. The call to `f` sets this location to `42`; the code in `g` reads this location and uses it as the value for `b`, which is then returned. (Note: Without knowing what assembly code `gcc` generates, you can't be sure this is why the program behaves as it does, but it's the most probable explanation.)

(c) A Java compiler would reject the program (during semantic analysis), because the use of `b` before it has been defined violates Java's "definite assignment" property.

(d) We would need to change the declaration of `int b` to `var b : Int = ...` or `val b :Int = ...`, for some expression ..., because Scala does not allow uninitialized local variables.

4.(a)

$$S\texttt{;}\ \texttt{while}\ \texttt{(}\neg\ E\texttt{)}\ \texttt{do}\ S$$

(b)

$$\frac{\{P\}S\{Q\} \quad \{Q \wedge \neg E\}S\{Q\}}{\{P\}\ \texttt{repeat}\ S\ \texttt{until}\ E\ \{Q \wedge E\}}\ \text{(REPEAT-UNTIL)}$$

5.

$$\frac{\langle e_{c1}, E, S\rangle \Downarrow \langle v_{c1}, S'\rangle \quad \langle e_{c2}, E, S'\rangle \Downarrow \langle v_{c2}, S''\rangle \quad v_{c1} \neq 0 \quad \langle e_{b1}, E, S''\rangle \Downarrow \langle v_{b1}, S'''\rangle}{\langle \texttt{(gifnz}\ e_{c1}\ \ e_{b1}\ \ e_{c2}\ \ e_{b2}\texttt{)}, E, S\rangle \Downarrow \langle v_{b1}, S'''\rangle}\ \text{(Gif-1)}$$

$$\frac{\langle e_{c1}, E, S\rangle \Downarrow \langle v_{c1}, S'\rangle \quad \langle e_{c2}, E, S'\rangle \Downarrow \langle v_{c2}, S''\rangle \quad v_{c2} \neq 0 \quad \langle e_{b2}, E, S''\rangle \Downarrow \langle v_{b2}, S'''\rangle}{\langle \texttt{(gifnz}\ e_{c1}\ \ e_{b1}\ \ e_{c2}\ \ e_{b2}\texttt{)}, E, S\rangle \Downarrow \langle v_{b2}, S'''\rangle}\ \text{(Gif-2)}$$