Đã bắt đầu vào lúc Tuesday, 2 October 2018, 2:12 PM
Tình trạng Đã hoàn thành
Hoàn thành vào lúc Tuesday, 2 October 2018, 2:36 PM
Thời gian thực hiện 24 phút 33 giây
Điểm **6,48** của 10,00 (**65**%)

Câu hỏi **1**
Hoàn thành
Điểm 2,25 của 3,00

Cho văn phạm được viết trên ANTLR như sau:

program: vardecls ;

vardecls: vardecl+ ;

vardecl: type ids ;

type: INTTYPE | FLOATTYPE ;

ids: ID COMMA ids | ID ;

Và AST tương ứng với văn phạm trên được định nghĩa RÚT GỌN trên Python như sau:

```
class AST(ABC)
class Program(AST):
  def __init__(self,decls:List[VarDecl]):
class VarDecl(AST):
 def __init__(self,typ:Type,id:List[String]):
class Type(AST)
class IntType(Type)
class FloatType(Type)
```

Hãy điền vào các chỗ trống để sinh ra AST trên, với các qui ước sau:

- Chỉ sử dụng khoảng trắng khi cần thiết
- Chỉ viết trên 1 dòng
- Nếu cần dùng 1 biến thì đặt tên x, nếu cần 2 biến thì biến đầu tiên xuất hiện có tên x và biến xuất hiện kế tiếp có tên y.

```
class ASTGeneration(MPVisitor):
 def visitProgram(self,ctx:MPParser.ProgramContext):
   return Program( self.visit(ctx.vardecls()) )

 def visitVardecls(self,ctx:MPParser.VardeclsContext):
   return [self.visit(x)for x in ctx.vardecl()] # có 1 for trong đáp án

 def visitVardecl(self,ctx:MPParser.VardeclContext):
   return VarDecl(self.visit(ctx.type()),self.vi

 def visitType(self,ctx:MPParser.TypeContext):
   return IntType() if ctx.INTTYPE() else FloatType()
 def visitIds(self,ctx:MPParser.IdsContext):
   return [ctx.ID().getText()]+self.visit(ctx.ids if ctx.getChildCount() == 3 else [ctx.ID().getText()] #nối 2 list bằng dấu +, không có
khoảng trắng
```

Câu hỏi **2**

Hoàn thành

Điểm 1,23 của 4,00

Cho văn phạm MP được viết trên ANTLR như sau:

exp: term COMPARE term | term ; # COMPARE is none-association

term: (factor EXPONENT)* factor ; # EXPONENT is right-association

factor: operand (ANDOR operand)* ; # ANDOR is left-association

operand: INTLIT | BOOLIT | LB exp RB ;

Và AST tương ứng với văn phạm trên được định nghĩa RÚT GỌN trên Python như sau:

```
class Exp(ABC)
class Binary(Exp):
  def __init__(self,op:String,left:Exp,right:Exp): #dùng getText() để lấy String ứng với op
class IntLit(Exp):
 def __init__(self,val:int):
class BoolLit(Exp):
 def __init__(self,val:boolean):
```

Hãy điền vào các chỗ trống để sinh ra AST trên, với các qui ước sau:

- Chỉ sử dụng khoảng trắng khi cần thiết

- Chỉ viết trên 1 dòng

- Nếu cần dùng 1 biến thì đặt tên x, nếu cần 2 biến thì biến đầu tiên xuất hiện có tên x và biến xuất hiện kế tiếp trong cùng chỗ trống sẽ có tên y.

- **x[::-1] trả về danh sách x bị đảo ngược, x[1:] trả về danh sách x không có phần tử đầu tiên, zip(l1,l2) tạo ra danh sách có các phần tử là các cặp tương ứng từ l1 và l2.**

- Giả sử có hàm toBool(String) để đổi 1 String thành 1 giá trị boolean

```
from functools import reduce
class ASTGeneration(MPVisitor):
 def visitExp(self,ctx:MPParser.ExpContext):
   return Binary(ctx.COMPARE().getText(),s if ctx.COMPARE() else self.visit(ctx.term(0))

 def visitTerm(self,ctx:MPParser.TermContext):
   rl = ctx.factor()[::-1]
   cl = zip(ctx.EXPONENT()[::-1],rl[1:])
   dl = zip(ctx.EXPONENT(),ctx.factor()[1:])
   return                                  # có 1 reduce trong đáp án, không có for hay map

 def visitFactor(self,ctx:MPParser.FactorContext):
   rl = ctx.operand()[::-1]
   cl = zip(ctx.ANDOR()[::-1],rl[1:])
   dl = zip(ctx.ANDOR(),ctx.operand()[1:])
   return                                  # có 1 reduce trong đáp án, không có for hay map

 def visitOperand(self,ctx:MPParser.OperandContext):
   return self.visit(ctx.exp()) if ctx.getChildCount() == 3 else IntLit(int(ctx.INTLIT().getText())) if ctx.INTLIT() else
BoolLit(toBool(ctx.BOOLIT().getTe
```

+ init = rl[0] -> visit
+ func: lambda x, y:
    + x: factor
    + y: element of cl -> tuple (EXPONENT, factor)
        -> y[0] = exponent -> getText()
        -> y[1] = factor -> visit

```
reduce(
   lambda x,y: Binary( y[0].getText(),  x,  self.visit(y[1]) ),
   cl,
   self.visit( rl[0] )
```

Câu hỏi **3**

Hoàn thành

Điểm 3,00 của 3,00

Cho văn phạm MP được viết trên ANTLR như sau:

program: vardecl+ EOF;

vardecl: type ids SEMI ;

type: INTTYPE | FLOATTYPE | ARRAY LB INTLIT RB OF type ;

ids: ID COMMA ids | ID ;

Hãy điền vào chỗ trống để hoàn thành một visitor để đếm các node trung gian (các node ứng với các ký hiệu không kết thúc) trên cây phân tích cú pháp (parse tree)? Để so trùng khớp với đáp án, hãy lưu ý:

- Chỉ dùng khoảng trắng khi cần thiết

- Chỉ viết trên 1 dòng

- Tên biến lần lượt là x, y, z (nếu chỉ có 1 biến thì phải dùng x, nếu có 2 biến thì biến xuất hiện đầu tiên là x, kế đó là y,...)

- Nếu có số nguyên trong biểu thức cộng + thì chỉ được phép có 1 số nguyên và số nguyên này phải là toán hạng bên trái nhất, ví dụ 3 + self.visit()

- Các chú thích đi kèm với mỗi chỗ trống

```
class Count(MPVisitor):
  def visitProgram(self,ctx:MPParser.ProgramContext):
    return 1+sum([self.visit(x)for x in ctx.vard  #  đáp án có dùng for

  def visitVardecl(self,ctx:MPParser.VardeclContext):
    return 1+self.visit(ctx.type())+self.visit(ctx

  def visitType(self,ctx:MPParser.TypeContext):
    return 1+(self.visit(ctx.type())  if ctx.type() else 0)

  def visitIds(self,ctx:MPParser.IdsContext):
    return 1+self.visit(ctx.ids())  if ctx.ids() else 1
```