Đã bắt đầu vào lúc	Tuesday, 3 November 2020, 2:40 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Tuesday, 3 November 2020, 3:19 PM
Thời gian thực hiện	38 phút 50 giây
Điểm	5,80/6,00
Điểm	**9,67** của 10,00 (**97**%)

Câu hỏi **1**

Chính xác

Điểm 1,00 của 1,00

Given the grammar of MP as follows:

program: vardecls EOF;

vardecls: vardecl vardecltail;

vardecltail: vardecl vardecltail | ;

vardecl: mptype ids ';' ;

mptype: INTTYPE | FLOATTYPE;

ids: ID ',' ids | ID;

INTTYPE: 'int';

FLOATTYPE: 'float';

ID: [a-z]+ ;

Please copy the following class into your answer and modify the bodies of its methods to count the terminal nodes in the parse tree?

```
class ASTGeneration(MPVisitor):
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return None
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return None
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        return None
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return None
    def visitMptype(self,ctx:MPParser.MptypeContext):
        return None
    def visitIds(self,ctx:MPParser.IdsContext):
        return None
```

For example:

Test	Result
`"int a;"`	`4`

Answer: (penalty regime: 10, 20, ... %)

```
class ASTGeneration(MPVisitor):
    # program: vardecls EOF;
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return 1 + ctx.vardecls().accept(self)

    # vardecls: vardecl vardecltail;
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return ctx.vardecl().accept(self) + ctx.vardecltail().accept

```

```
    # vardecltail: vardecl vardecltail | ;
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        if(ctx.getChildCount() == 0):
            return 0
        else:
            return ctx.vardecl().accept(self) + ctx.vardecltail().ac

    # vardecl: mptype ids ';' ;
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return 1 + ctx.mptype().accept(self) + ctx.ids().accept(self
```

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

/

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

Given the grammar of MP as follows:

program: vardecls EOF;

vardecls: vardecl vardecltail;

vardecltail: vardecl vardecltail | ;

vardecl: mptype ids ';' ;

mptype: INTTYPE | FLOATTYPE;

ids: ID ',' ids | ID;

INTTYPE: 'int';

FLOATTYPE: 'float';

ID: [a-z]+ ;

Please copy the following class into your answer and modify the bodies of its methods to count the non-terminal nodes in the parse tree?

```
class ASTGeneration(MPVisitor):
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return None
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return None
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        return None
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return None
    def visitMptype(self,ctx:MPParser.MptypeContext):
        return None
    def visitIds(self,ctx:MPParser.IdsContext):
        return None
```

For example:

Test	Result
`"int a;"`	`6`

Answer: (penalty regime: 10, 20, ... %)

```
class ASTGeneration(MPVisitor):
    # program: vardecls EOF;
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return 2 + ctx.vardecls().accept(self)

    # vardecls: vardecl vardecltail;
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return 2 + ctx.vardecl().accept(self) + ctx.vardecltail().ac

```

```
    # vardecltail: vardecl vardecltail | ;
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        if(ctx.getChildCount() == 0):
            return 0
        else:
            return 2 + ctx.vardecl().accept(self) + ctx.vardecltail(

    # vardecl: mptype ids ';' ;
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return 2 + ctx.mptype().accept(self) + ctx.ids().accept(self
```

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **3**

Chính xác

Điểm 0,90 của 1,00

Given the grammar of MP as follows:

program: vardecls EOF;

vardecls: vardecl vardecltail;

vardecltail: vardecl vardecltail | ;

vardecl: mptype ids ';' ;

mptype: INTTYPE | FLOATTYPE;

ids: ID ',' ids | ID;

INTTYPE: 'int';

FLOATTYPE: 'float';

ID: [a-z]+ ;

and AST classes as follows:

/

```
class AST(ABC):
    def __eq__(self, other):
        return self.__dict__ == other.__dict__

    @abstractmethod
    def accept(self, v, param):
        return v.visit(self, param)

class Type(AST):
    __metaclass__ = ABCMeta
    pass

class IntType(Type):
    def __str__(self):
        return "IntType"

    def accept(self, v, param):
        return v.visitIntType(self, param)

class FloatType(Type):
    def __str__(self):
        return "FloatType"

    def accept(self, v, param):
        return v.visitFloatType(self, param)


class Program(AST):
    #decl:list(Decl)
    def __init__(self, decl):
        self.decl = decl

    def __str__(self):
        return "Program([" + ','.join(str(i) for i in self.decl) + "])"

    def accept(self, v: Visitor, param):
        return v.visitProgram(self, param)

class Decl(AST):
    __metaclass__ = ABCMeta
    pass

class VarDecl(Decl):
    #variable:Id
    #varType: Type
    def __init__(self, variable, varType):
        self.variable = variable
        self.varType = varType

    def __str__(self):
```

/

```
        return "VarDecl(" + str(self.variable) + "," + str(self.varType) +
")"

    def accept(self, v, param):
        return v.visitVarDecl(self, param)


class Id(AST):
    #name:string
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return "Id(" + self.name + ")"

    def accept(self, v, param):
        return v.visitId(self, param)
```

Please copy the following class into your answer and modify the bodies of its methods to generate the AST of a MP input?

class ASTGeneration(MPVisitor):

def visitProgram(self,ctx:MPParser.ProgramContext):

return None

def visitVardecls(self,ctx:MPParser.VardeclsContext):

return None

def visitVardecltail(self,ctx:MPParser.VardecltailContext):

return None

def visitVardecl(self,ctx:MPParser.VardeclContext):

return None

def visitMptype(self,ctx:MPParser.MptypeContext):

return None

def visitIds(self,ctx:MPParser.IdsContext):

return None

For example:

Test	Result
`"int a;"`	`Program([VarDecl(Id(a),IntType)])`

Answer: (penalty regime: 10, 20, ... %)

```
1 class ASTGeneration(MPVisitor):
2     # program: vardecls EOF;
3     def visitProgram(self,ctx:MPParser.ProgramContext):
```

```
4           return Program(ctx.vardecls().accept(self))
5
6       # vardecls: vardecl vardecltail;
7       def visitVardecls(self,ctx:MPParser.VardeclsContext):
8           return ctx.vardecl().accept(self) + ctx.vardecltail().accept
9           #1 list [1 đối tượng] + 1 list [nhiều đối tượng], gom vô 1 l
10
11      # vardecltail: vardecl vardecltail | ;
12      def visitVardecltail(self,ctx:MPParser.VardecltailContext):
13          if(ctx.getChildCount() == 0):
14              return []
15          else:
16              return ctx.vardecl().accept(self) + ctx.vardecltail().ac
17
18      # vardecl: mptype ids ';' ;
19      def visitVardecl(self,ctx:MPParser.VardeclContext):
```

Chính xác

Điểm cho bài nộp này: 1,00/1,00. Tính toán cho lần làm bài trước đó, điểm **0,90/1,00**.

Câu hỏi **4**

Đúng một phần

Điểm 0,90 của 1,00

Given the grammar of MP as follows:

program: exp EOF;

exp: term ASSIGN exp | term;

term: factor COMPARE factor | factor;

factor: factor ANDOR operand | operand;

operand: ID | INTLIT | BOOLIT | '(' exp ')';

INTLIT: [0-9]+ ;

BOOLIT: 'True' | 'False' ;

ANDOR: 'and' | 'or' ;

ASSIGN: '+=' | '-=' | '&=' | '|=' | ':=' ;

COMPARE: '=' | '<>' | '>=' | '<=' | '<' | '>' ;

ID: [a-z]+ ;

and AST classes as follows:

```
class AST(ABC):
    def __eq__(self, other):
        return self.__dict__ == other.__dict__

    @abstractmethod
    def accept(self, v, param):
        return v.visit(self, param)

class Expr(AST):
    __metaclass__ = ABCMeta
    pass

class Binary(Expr):
    #op:string:
    #left:Expr
    #right:Expr
    def __init__(self, op, left, right):
        self.op = op
        self.left = left
        self.right = right

    def __str__(self):
        return "Binary(" + self.op + "," + str(self.left) + "," + str(self.r
ight) + ")"

    def accept(self, v, param):
        return v.visitBinaryOp(self, param)

class Id(Expr):
    #value:string
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return "Id(" + self.value + ")"

    def accept(self, v, param):
        return v.visitId(self, param)

class IntLiteral(Expr):
    #value:int
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return "IntLiteral(" + str(self.value) + ")"

    def accept(self, v, param):
        return v.visitIntLiteral(self, param)
```

```
class BooleanLiteral(Expr):
    #value:boolean
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return "BooleanLiteral(" + str(self.value) + ")"

    def accept(self, v, param):
        return v.visitBooleanLiteral(self, param)
```

Please copy the following class into your answer and modify the bodies of its methods to generate the AST of a MP input?

class ASTGeneration(MPVisitor):

def visitProgram(self,ctx:MPParser.ProgramContext):

return None

def visitExp(self,ctx:MPParser.ExpContext):

return None

def visitTerm(self,ctx:MPParser.TermContext):

return None

def visitFactor(self,ctx:MPParser.FactorContext):

return None

def visitOperand(self,ctx:MPParser.OperandContext):

return None

For example:

Test	Result
`"a := b := 4"`	`Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))`

Answer: (penalty regime: 10, 20, ... %)

```
22              right = ctx.factor(1).accept(self)
23              op = ctx.COMPARE().getText()
24              return Binary(op,left,right)
25
26      # factor: factor ANDOR operand | operand;
27      def visitFactor(self,ctx:MPParser.FactorContext):
28          if(ctx.getChildCount() == 1):
29              return ctx.operand().accept(self)
30          else:
31              left = ctx.factor().accept(self)
32              right = ctx.operand().accept(self)
33              op = ctx.ANDOR().getText()
34              return Binary(op,left,right)
35
36      # operand: ID | INTLIT | BOOLIT | '(' exp ')';
```

```
37      def visitOperand(self,ctx:MPParser.OperandContext):
38          if(ctx.getChildCount() == 1):
39              if(ctx.getChild(0) is ctx.ID()):
40                  return Id(ctx.getChild(0).getText())
41              elif(ctx.getChild(0) is ctx.INTLIT()):
```

Đúng một phần

Điểm cho bài nộp này: 0,33/1,00. Tính toán cho lần làm bài trước đó, điểm **0,90/1,00**.

Câu hỏi **5**

Chính xác

Điểm 1,00 của 1,00

Given the grammar of MP as follows:

program: vardecl+ EOF;

vardecl: mptype ids ';' ;

mptype: INTTYPE | FLOATTYPE;

ids: ID (',' ID)*;

INTTYPE: 'int';

FLOATTYPE: 'float';

ID: [a-z]+ ;

and AST classes as follows:

class Program:#decl:list(VarDecl)

class Type(ABC): pass

class IntType(Type): pass

class FloatType(Type): pass

class VarDecl: #variable:Id; varType: Type

class Id: #name:str

Please copy the following class into your answer and modify the bodies of its methods to generate the AST of a MP input?

```
class ASTGeneration(MPVisitor):
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return None
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return None
    def visitMptype(self,ctx:MPParser.MptypeContext):
        return None
    def visitIds(self,ctx:MPParser.IdsContext):
        return None
```

For example:

Test	Result
`"int a;"`	`Program([VarDecl(Id(a),IntType)])`

Answer: (penalty regime: 10, 20, ... %)

```
from functools import reduce
class ASTGeneration(MPVisitor):
    # program: vardecl+ EOF;
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return Program(reduce(
            lambda acc,vardeclItem: acc+vardeclItem.accept(self), ct
            )
        # list_vardecl = ctx.vardecl()
```

```
        # return Program([x for vardecl in list_vardecl for x in var
        # ctx.vardecl() để lấy 1 list
        # vardecl_list là [[vardecl1,vardecl2],[vardecl3],[vardecl4]

    # vardecl: mptype ids ';' ; ids,vardecl+ là list=>list của list=
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        var_type = ctx.mptype().accept(self)
        id_list = ctx.ids().accept(self) #ctx.ids() ko phải 1 list n
        return [VarDecl(id, var_type) for id in id_list]
        #[VarDecl([Ids(a)],IntType),VarDecl([Id(b),Id(c)],FloatType)
```

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **6**

Chính xác

Điểm 1,00 của 1,00

Given the grammar of MP as follows:

program: exp EOF;

exp: (term ASSIGN)* term;

term: factor COMPARE factor | factor;

factor: operand (ANDOR operand)*;

operand: ID | INTLIT | BOOLIT | '(' exp ')';

INTLIT: [0-9]+ ;

BOOLIT: 'True' | 'False' ;

ANDOR: 'and' | 'or' ;

ASSIGN: '+=' | '-=' | '&=' | '|=' | ':=' ;

COMPARE: '=' | '<>' | '>=' | '<=' | '<' | '>' ;

ID: [a-z]+ ;

and AST classes as follows:

class Expr(ABC):

class Binary(Expr): #op:string;left:Expr;right:Expr

class Id(Expr): #value:string

class IntLiteral(Expr): #value:int

class BooleanLiteral(Expr): #value:boolean

Please copy the following class into your answer and modify the bodies of its methods to generate the AST of a MP input?

```
class ASTGeneration(MPVisitor):
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return None
    def visitExp(self,ctx:MPParser.ExpContext):
        return None
    def visitTerm(self,ctx:MPParser.TermContext):
        return None
    def visitFactor(self,ctx:MPParser.FactorContext):
        return None
    def visitOperand(self,ctx:MPParser.OperandContext):
        return None
```

For example:

Test	Result
`"a := b := 4"`	`Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))`

Answer: (penalty regime: 10, 20, ... %)

```
1 from functools import reduce
```

```
class ASTGeneration(MPVisitor):
    # program: exp EOF;
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return ctx.exp().accept(self)

    # exp: (term ASSIGN)* term;
    #=> cách viết của kết hợp phải
    def visitExp(self,ctx:MPParser.ExpContext):
        if(ctx.ASSIGN()):
            lstTermDecs = ctx.term()[::-1] #{term1,term2}
            lastTerm = ctx.term(len(lstTermDecs)-1).accept(self) #te
            lstOpDecs = ctx.ASSIGN()[::-1]
            lstZip = list(zip(lstOpDecs,lstTermDecs[1:]))
            #dùng zip để duyệt cả 2 danh sách song song tương hỗ nha

            return reduce(lambda acc, zipItem: Binary(zipItem[0].get
            #acc là biến cộng dồn, bao gồm một nùi Binary
            #a := b := 4 là kết hợp phải => đảo ngược danh sách
```

Chính xác
Điểm cho bài nộp này: 1,00/1,00.