Started on	Thursday, 9 March 2023, 2:20 PM
State	Finished
Completed on	Thursday, 9 March 2023, 2:25 PM
Time taken	5 mins 58 secs
Marks	5.70/6.00
Grade	**9.50** out of 10.00 (**95**%)

Question 1

Correct

Mark 0.90 out of 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, ... %)

```
from antlr4 import *
class ASTGeneration(MPVisitor):
    #program: vardecls EOF
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return self.visit(ctx.vardecls()) + 1 #1 la EOF
    # vardecltail : vardecl vardecltail;
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail())
    # vardecltail : vardecl vardecltail |
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        if (ctx.vardecl()):
            return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail())
        return 0
    #vardecl: mptype ids ';'
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return self.visit(ctx.mptype()) + self.visit(ctx.ids()) + 1
```

```
18      def visitMptype(self,ctx:MPParser.MptypeContext):
19          return 1
20      #ids: ID ',' ids | ID;
21      def visitIds(self,ctx:MPParser.IdsContext):
22          if ctx.getChildCount() == 1:
```

	Test	Expected	Got	
✔	"int a;"	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.90/1.00**.

Question 2

Correct

Mark 1.00 out of 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):

    def visitProgram(self,ctx:MPParser.ProgramContext):
        return self.visit(ctx.vardecls())
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail()) + 2
#vardecl: mptype ids ';' ;

    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        if (ctx.vardecl()):
            return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail())
        return 0
#vardecl: mptype ids ';' ;
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        return self.visit(ctx.mptype()) + self.visit(ctx.ids()) + 2

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

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

	Test	Expected	Got	
✔	"int a;"	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 3

Correct

Mark 1.00 out of 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, ... %)

```
class ASTGeneration(MPVisitor):

    def visitProgram(self,ctx:MPParser.ProgramContext):
        return Program(self.visit(ctx.vardecls()))
    def visitVardecls(self,ctx:MPParser.VardeclsContext):
        return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail())
# vardecltail: vardecl vardecltail | ; Trả về list các vardecl
    def visitVardecltail(self,ctx:MPParser.VardecltailContext):
        if (ctx.getChildCount() == 0):
            return []
        return self.visit(ctx.vardecl()) + self.visit(ctx.vardecltail())
# Trả về list các phần tử, cùng kiểu của nó.
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        mptype = self.visit(ctx.mptype())
        idlist = self.visit(ctx.ids())
        return [VarDecl(x,mptype) for x in idlist]
    def visitMptype(self,ctx:MPParser.MptypeContext):
        return IntType() if ctx.INTTYPE() else FloatType()

    def visitIds(self,ctx:MPParser.IdsContext):
        if (ctx.getChildCount() == 1):
            return [Id(ctx.ID().getText())]
```

	Test	Expected
✔	`"int a;"`	`Program([VarDecl(Id(a),IntType)])`
✔	`"""int a,b;"""`	`Program([VarDecl(Id(a),IntType),VarDecl(Id(b),IntType)])`
✔	`"int a;float b;"`	`Program([VarDecl(Id(a),IntType),VarDecl(Id(b),FloatType)])`

	Test	Expected
✔	`"int a,b;float c;"`	`Program([VarDecl(Id(a),IntType),VarDecl(Id(b),IntType),VarDecl(Id(c),FloatType)])`
✔	`"int a,b;float c,d,e;"`	`Program([VarDecl(Id(a),IntType),VarDecl(Id(b),IntType),VarDecl(Id(c),FloatType),VarDecl(Id(d),FloatType`

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 4

Correct

Mark 1.00 out of 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):
    def visitProgram(self,ctx:MPParser.ProgramContext):
        return Program(reduce(lambda prev, curr: prev + self.visit(curr), ctx.vardecl(), []))
#1 vardecl cua antlr khi visit, tra ve 1 list cac vardecl cua ast.
#2 list long trong list -> flatten.
    def visitVardecl(self,ctx:MPParser.VardeclContext):
        mptype = self.visit(ctx.mptype())
        ids = self.visit(ctx.ids())
        return list(map(lambda x: VarDecl(x,mptype),ids))

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

        return IntType() if ctx.INTTYPE() else FloatType()

```

```
16    def visitIds(self,ctx:MPParser.IdsContext):
17
18        return list(map(lambda x: Id(x.getText()), ctx.ID()))
```

	Test	Expected	Got	
✔	"int a;"	Program([VarDecl(Id(a),IntType)])	Program([VarDecl(Id(a),IntType)])	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 5

Correct

Mark 0.90 out of 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.right) + ")"

    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, ... %)

```
class ASTGeneration(MPVisitor):

    def visitProgram(self,ctx:MPParser.ProgramContext):
        return self.visit(ctx.exp())

    def visitExp(self,ctx:MPParser.ExpContext):
        if ctx.ASSIGN():
            left = self.visit(ctx.term())
            right = self.visit(ctx.exp())
            return Binary(ctx.ASSIGN().getText(),left,right)
        return self.visit(ctx.term())

    def visitTerm(self,ctx:MPParser.TermContext):
        if ctx.COMPARE():
            left = self.visit(ctx.factor(0))
            right = self.visit(ctx.factor(1))
            return Binary(ctx.COMPARE().getText(),left,right)
        return self.visit(ctx.factor(0))

    def visitFactor(self,ctx:MPParser.FactorContext):
        if ctx.ANDOR():
            left = self.visit(ctx.factor())
```

	Test	Expected	Got	
✔	`"a := b := 4"`	`Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))`	`Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.90/1.00**.

Question 6

Correct

Mark 0.90 out of 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, ... %)

```
class ASTGeneration(MPVisitor):

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

        return self.visit(ctx.exp())

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

```
        terms = [self.visit(x) for x in ctx.term()] #[operand(d), _operand(e), _operand(f)]
        assigns = [x.getText() for x in ctx.ASSIGN()]
        right = terms[-1]
        for i in range(len(assigns)):
            op = assigns[len(assigns) - i- 1] # left22222222222222222222222222222222
            left = terms[len(assigns) - i- 1]
            right = Binary(op,left,right)
        return right
    def visitTerm(self,ctx:MPParser.TermContext):
        if ctx.COMPARE():
            left = self.visit(ctx.factor(0))
            right = self.visit(ctx.factor(1))
            return Binary(ctx.COMPARE().getText(),left,right)
        return self.visit(ctx.factor(0))
    def visitFactor(self,ctx:MPParser.FactorContext):
```

	Test	Expected	Got	
✔	"a := b := 4"	Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))	Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4)))	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.90/1.00**.