**[CC03] AST Programming Code 2**

[Back](https://lms.hcmut.edu.vn/mod/quiz/view.php?id=172700)

Time left 0:16:50

Hide

Top of Form

Question **1**

Correct

Marked out of 1.00

Flag question

Question text

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 return the height of the parse tree? Your code starts at line 10.

class Height(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

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

from functools import reduce

class Height(MPVisitor):

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

return 1 + self.visit(ctx.vardecls())

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

return 1 + reduce(max, [self.visit(ctx.vardecl()), self.visit(ctx.vardecltail())])

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

if ctx.vardecl():

return 1 + reduce(max, [self.visit(ctx.vardecl()), self.visit(ctx.vardecltail())])

else:

return 1

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

return 1 + reduce(max, [self.visit(ctx.mptype()), self.visit(ctx.ids())])

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

return 2

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

if ctx.ids():

return 1 + self.visit(ctx.ids())

else:

return 2

CheckQuestion 1

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | "int a;" | 5 | 5 |  |
|  | """int a,b;""" | 6 | 6 |  |
|  | "int a;float b;" | 6 | 6 |  |
|  | "int a,b;float c;" | 6 | 6 |  |
|  | "int a,b;float c,d,e;" | 8 | 8 |  |
|  | "int" | 2 | 2 |  |
|  | "a" | 2 | 2 |  |
|  | "a,b,c" | 4 | 4 |  |
|  | "int a;" | 3 | 3 |  |
|  | "int a,b,c;" | 5 | 5 |  |
|  | "int a;float b;" | 5 | 5 |  |
|  | "" | 1 | 1 |  |

Passed all tests!

Top of Form

Question **2**

Correct

Marked out of 1.00

Flag question

Question text

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 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

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

from functools import reduce

class ASTGeneration(MPVisitor):

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

return self.visit(ctx.exp())

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

if ctx.getChildCount() == 3:

return reduce(lambda x, y: Binary(ctx.ASSIGN().getText(), x, y), [self.visitTerm(ctx.term()), self.visitExp(ctx.exp())])

return self.visit(ctx.term())

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

if ctx.getChildCount() == 3:

return reduce(lambda x, y: Binary(ctx.COMPARE().getText(), x, y), [self.visitFactor(ctx.factor(0)), self.visitFactor(ctx.factor(1))])

return self.visit(ctx.factor(0))

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

if ctx.getChildCount() == 3:

return Binary(ctx.ANDOR().getText(), self.visitFactor(ctx.factor()), self.visitOperand(ctx.operand()))

return self.visit(ctx.operand())

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

if ctx.ID():

return Id(ctx.ID().getText())

if ctx.INTLIT():

return IntLiteral(int(ctx.INTLIT().getText()))

if ctx.BOOLIT():

return BooleanLiteral(True if ctx.BOOLIT().getText() == "True" else False)

else:

return self.visit(ctx.exp())

CheckQuestion 2

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | "a := b := 4" | Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4))) | Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4))) |  |
|  | """a += b -= a and (b > 3)""" | Binary(+=,Id(a),Binary(-=,Id(b),Binary(and,Id(a),Binary(>,Id(b),IntLiteral(3))))) | Binary(+=,Id(a),Binary(-=,Id(b),Binary(and,Id(a),Binary(>,Id(b),IntLiteral(3))))) |  |
|  | "a or b and True" | Binary(and,Binary(or,Id(a),Id(b)),BooleanLiteral(True)) | Binary(and,Binary(or,Id(a),Id(b)),BooleanLiteral(True)) |  |

Passed all tests!

Bottom of Form

Question **3**

Correct

Marked out of 1.00

Flag question

Question text

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

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

from functools import reduce

class ASTGeneration(MPVisitor):

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

return self.visit(ctx.exp())

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

if len(ctx.ASSIGN()) == 0: return self.visit(ctx.term(0))

return reduce(lambda x,y: Binary(y[0].getText(), self.visit(y[1]), x), list(zip(ctx.ASSIGN(), ctx.term()[:-1]))[::-1], self.visit(ctx.term()[-1]))

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

if ctx.getChildCount() == 3:

return Binary(ctx.COMPARE().getText(), self.visit(ctx.factor(0)), self.visit(ctx.factor(1)))

return self.visit(ctx.factor(0))

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

return reduce(lambda x, y: Binary(y[0].getText(), x, self.visit(y[1])), list(zip(ctx.ANDOR(), ctx.operand()[1:])), self.visit(ctx.operand(0)))

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

if ctx.ID():

return Id(ctx.ID().getText())

if ctx.INTLIT():

return IntLiteral(int(ctx.INTLIT().getText()))

if ctx.BOOLIT():

return BooleanLiteral(True if ctx.BOOLIT().getText() == "True" else False)

else:

return self.visit(ctx.exp())

CheckQuestion 3

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | "a := b := 4" | Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4))) | Binary(:=,Id(a),Binary(:=,Id(b),IntLiteral(4))) |  |
|  | """a += b -= a and (b > 3)""" | Binary(+=,Id(a),Binary(-=,Id(b),Binary(and,Id(a),Binary(>,Id(b),IntLiteral(3))))) | Binary(+=,Id(a),Binary(-=,Id(b),Binary(and,Id(a),Binary(>,Id(b),IntLiteral(3))))) |  |
|  | "a or b and True" | Binary(and,Binary(or,Id(a),Id(b)),BooleanLiteral(True)) | Binary(and,Binary(or,Id(a),Id(b)),BooleanLiteral(True)) |  |

Passed all tests!

Question **4**

Correct

Marked out of 1.00

Flag question

Question text

Given the grammar of MP as follows:

program: mptype EOF;

arraytype:  primtype dimens  ;

mptype: primtype | arraytype;

primtype: INTTYPE | FLOATTYPE;

dimens: dimen+;

dimen: '[' num '..' num ']';

num: '-'? INTLIT;

INTLIT: [0-9]+ ;

INTTYPE: 'integer';

FLOATTYPE: 'real';

and AST classes as follows:

class Type():abstract

class CompoundType(Type):abstract

class UnionType(CompoundType):#firstType:Type,secondType:primType

class ArrayType(CompoundType):#indexType:Type,eleType:primType

class PrimType(Type):abstract

class IntType(PrimType): pass

class FloatType(PrimType): pass

class RangeType(PrimType): #lowbound:int; highbound:int

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 visitMptype(self,ctx:MPParser.MptypeContext):

        return None

    def visitArraytype(self,ctx:MPParser.ArraytypeContext):

        return None

    def visitPrimtype(self,ctx:MPParser.PrimtypeContext):

        return None

    def visitDimens(self,ctx:MPParser.DimensContext):

        return None

    def visitDimen(self,ctx:MPParser.DimenContext):

        return None

    def visitNum(self,ctx:MPParser.DimenContext):

        return None

**For example:**

| **Test** | **Result** |
| --- | --- |
| "integer[1..3]" | ArrayType(RangeType(1,3),IntType) |

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

from functools import reduce

class ASTGeneration(MPVisitor):

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

return self.visit(ctx.mptype())

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

if ctx.primtype():

return self.visit(ctx.primtype())

elif ctx.arraytype():

return reduce(lambda x, y: ArrayType(x, y), [self.visit(ctx.arraytype().dimens()), self.visit(ctx.arraytype().primtype())])

def visitDimens(self, ctx:MPParser.DimensContext):

ranges = [self.visit(dim) for dim in ctx.dimen()]

return ranges[0] if len(ranges) == 1 else reduce(lambda x, y: UnionType(x, y), ranges)

def visitDimen(self, ctx:MPParser.DimenContext):

return RangeType(self.visit(ctx.num(0)), self.visit(ctx.num(1)))

def visitNum(self, ctx:MPParser.NumContext):

num\_str = ctx.getText()

if ctx.getChild(0).getText() == '-':

num\_str = '-' + num\_str[1:]

return int(num\_str)

def visitPrimtype(self, ctx:MPParser.PrimtypeContext):

if ctx.FLOATTYPE():

return FloatType()

elif ctx.INTTYPE():

return IntType()

CheckQuestion 4

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | "real [-3..0][-10..-1]" | ArrayType(UnionType(RangeType(-3,0),RangeType(-10,-1)),FloatType) | ArrayType(UnionType(RangeType(-3,0),RangeType(-10,-1)),FloatType) |  |
|  | "integer[1..3]" | ArrayType(RangeType(1,3),IntType) | ArrayType(RangeType(1,3),IntType) |  |
|  | """integer [1..100][-5..20][100..3000]""" | ArrayType(UnionType(UnionType(RangeType(1,100),RangeType(-5,20)),RangeType(100,3000)),IntType) | ArrayType(UnionType(UnionType(RangeType(1,100),RangeType(-5,20)),RangeType(100,3000)),IntType) |  |

Passed all tests!

Bottom of Form

### Question 5

Correct

Marked out of 1.00

Flag question

Question text

Given the grammar of MP as follows:

program: mptype EOF;

arraytype:  primtype dimen | arraytype dimen  ;

mptype: primtype | arraytype;

primtype: INTTYPE | FLOATTYPE;

dimen: '[' num '..' num ']';

num: '-'? INTLIT;

INTLIT: [0-9]+ ;

INTTYPE: 'integer';

FLOATTYPE: 'real';

and AST classes as follows:

class Type():abstract

class CompoundType(Type):abstract

class UnionType(CompoundType):#firstType:Type,secondType:primType

class ArrayType(CompoundType):#indexType:Type,eleType:primType

class PrimType(Type):abstract

class IntType(PrimType): pass

class FloatType(PrimType): pass

class RangeType(PrimType): #lowbound:int; highbound:int

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 visitMptype(self,ctx:MPParser.MptypeContext):

        return None

    def visitArraytype(self,ctx:MPParser.ArraytypeContext):

        return None

    def visitPrimtype(self,ctx:MPParser.PrimtypeContext):

        return None

    def visitDimen(self,ctx:MPParser.DimenContext):

        return None

    def visitNum(self,ctx:MPParser.DimenContext):

        return None

**For example:**

| **Test** | **Result** |
| --- | --- |
| "integer[1..3]" | ArrayType(RangeType(1,3),IntType) |

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

from functools import reduce

class ASTGeneration(MPVisitor):

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

return self.visit(ctx.mptype())

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

if ctx.primtype():

return self.visit(ctx.primtype())

return self.visit(ctx.arraytype())

def visitArraytype(self, ctx:MPParser.ArraytypeContext, lst=None):

if lst is None: lst = []

newLst = lst + [self.visit(ctx.dimen())]

if ctx.primtype():

reversedLst = newLst[::-1]

if reversedLst:

union\_type = reduce(lambda z, x: UnionType(z, x), reversedLst[1:], reversedLst[0])

return ArrayType(union\_type, self.visit(ctx.primtype()))

else:

return self.visitArraytype(ctx.arraytype(), newLst)

def visitPrimtype(self,ctx:MPParser.PrimtypeContext):

if ctx.INTTYPE(): return IntType()

return FloatType()

def visitDimen(self,ctx:MPParser.DimenContext):

return RangeType(self.visit(ctx.num()[0]), self.visit(ctx.num()[1]))

def visitNum(self,ctx:MPParser.DimenContext):

if ctx.getChildCount() == 2: return - (int(ctx.INTLIT().getText()))

return int(ctx.INTLIT().getText())

CheckQuestion 5

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | "real [-3..0][-10..-1]" | ArrayType(UnionType(RangeType(-3,0),RangeType(-10,-1)),FloatType) | ArrayType(UnionType(RangeType(-3,0),RangeType(-10,-1)),FloatType) |  |
|  | "integer[1..3]" | ArrayType(RangeType(1,3),IntType) | ArrayType(RangeType(1,3),IntType) |  |
|  | """integer [1..100][-5..20][100..3000]""" | ArrayType(UnionType(UnionType(RangeType(1,100),RangeType(-5,20)),RangeType(100,3000)),IntType) | ArrayType(UnionType(UnionType(RangeType(1,100),RangeType(-5,20)),RangeType(100,3000)),IntType) |  |

Passed all tests!

from functools import reduce

class ASTGeneration(MPVisitor):

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

return self.visit(ctx.mptype())

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

if ctx.primtype():

return self.visit(ctx.primtype())

return self.visit(ctx.arraytype())

def visitArraytype(self, ctx:MPParser.ArraytypeContext, lst=None):

if lst is None: lst = []

newLst = lst + [self.visit(ctx.dimen())]

if ctx.primtype():

reversedLst = newLst[::-1]

if reversedLst:

union\_type = reduce(lambda z, x: UnionType(z, x), reversedLst[1:], reversedLst[0])

return ArrayType(union\_type, self.visit(ctx.primtype()))

else:

return self.visit(ctx.arraytype())

def visitPrimtype(self,ctx:MPParser.PrimtypeContext):

if ctx.INTTYPE(): return IntType()

return FloatType()

def visitDimen(self,ctx:MPParser.DimenContext):

return RangeType(self.visit(ctx.num()[0]), self.visit(ctx.num()[1]))

def visitNum(self,ctx:MPParser.DimenContext):

if ctx.getChildCount() == 2: return - (int(ctx.INTLIT().getText()))

return int(ctx.INTLIT().getText())