2017 年春季学期编译原理第一次实验测试用例：目录

1 A 组测试用例

本组测试用例共 9 个，每个仅包含单个的词法或者语法错误。除特殊说明外，不可多报。多报、漏报错误，或者打印语法树都会导致扣分。错误编号和行号之后的说明文字不要求与给出的输出完全一致，仅供助教理解使用，不作为评分依据。

1.1 A-1

输入

```
int func_A1()
{
    int a1, _aaa_aa2;
    float 3f;
}
```

输出

```
Error type B at line 4 : syntax error, near ';'
```

说明：错误类型也可以是 A 类，或者一个 A 一个 B，但是只能在第 4 行。这里有一个非法标识符 3f，注意标识符可以以下划线开始，所以第三行正确。

1.2 A-2

输入

```
int func()
{
    int a, b;
    a = b + 4 * a;
    b = a && 2 || 4;
    return a ~ b;
}
```

输出

```
Error type A at line 6 : mysteriously character '~'
```

说明：必须有 type A 错误；可以多报一个 type B 错误。这里有一个非法的符号 ~。

1.3 A-3

输入

```
int method(){
{
    int a, b, c;
    c = a + b * a;
    b = a && c;
    return b;
}
```

输出

```
Error type B at line 8 : missing '}'
```

说明：第 8 行缺少匹配的括号，可以报错在第七行，或者开始的括号所在的第一二行。

1.4 A-4

输入

```
int main(){
    int a, b;
    a = b + 1;
    b = a + b
}
```

输出

```
Error type B at line 5: syntax error, before '}'
```

说明：第 4 行缺少分号，也可报错在第 4 行。

1.5 A-5

输入

```
int array (int a, int b)
{
    int a1, b1[10,2];
```

```
    b1[a] = b;
    b1[a] = a1 + b1[a];
}
```

输出

```
Error type B at line 3 : syntax error, near ']'
```

说明：第 3 行数组定义格式错误。

1.6 A-6

输入

```
int arrayDef()
{
    int i, a[10][2];
    i = 0;
    while (i < 10) {
        a[i][] = 1;
        a[i][1] = i;
        i = i + 1;
    }
}
```

输出

```
Error type B at line 6 : syntax error, near '['
```

说明：第 6 行数组下标访问时格式不正确，中括号内不可为空。

1.7 A-7

输入

```
struct Product{
    int name;
    int price;
    int date;
```

```
};
int main()
{
    Product a;
    a.name = a.price = 2;
    a.data = 1;
}
```

输出

```
Error type B at line 8 : syntax error, before ';'
```

说明：结构体使用时缺少 struct。

1.8 A-8

输入

```
struct Product{
    int name;
    int price;
    int date;
}
int main()
{
    struct Product a;
    a.name = a.price = 2;
    a.data = 1;
}
```

输出

```
Error type B at line 5 : syntax error, near '}'
```

说明：第 5 行定义结构体时少了分号，也可以报错在第六行。

1.9 A-9

输入

```
int function_1(int a)
{
    return a + 5;
}

int function_2(int b)
{
    c = function_1((2 * b);
    return c;
}
```

输出

```
Error type B at line 8 : syntax error, after '('
```

说明：第 8 行在函数调用时多了一个左括号。

2 B 组测试用例

本组测试用例共 2 个，每个用例包含多处不同的错误。除特殊说明外，漏报、多报错误或者出打印语法树都会导致扣分。

2.1 B-1

输入

```
struct Product
{
    int price;
    int weight;
}p[10],

int Calculate(struct Product product)
{
    if(product.weight < 1)
        return product.price;
```

```
11      else
12          return product.price / 2 * ( product->weight - 1 ) + product.
                price;
13  }
14
15  int main()
16  {
17      int priceA[10], weightA[10];
18      int result;
19      int i, N = 10;
20      while(i < N){
21          priceA[i] = i + (i * 2 - 1);
22          weightA[i] = 2 * i;
23          i = i + 1;
24      }
25      i = 0;
26
27      while(i < N){
28          p[i].price = priceA[i];
29          p[i].weight = weightA[];
30          result = result + Calculate(p[i]]);
31      }
32          return result;
33  }
```

输出

```
1  Error type B at line 7: syntax error, maybe you should check the
       definition.
2  Error type B at line 12: syntax error, after '('.
3  Error type B at line 29: syntax error, before ';'.
4  Error type B at line 30: syntax error, after '('.
```

说明：结构体定义分号错写成逗号，也可以报错在第五行；第 12 行结构体域访问符号用错；第 29 行数组访问下标为空；第 30 行多一个右中括号。

2.2 B-2

输入

```
1  int dot[10];
2  int val = 4;
3
4  struct Vector
5  {
6      int x, y;
7  };
8
9  int dotMultiply(struct Vector v1: struct Vector v2)
10 {
11     return v1.x*v2.y - v1.y*v2.x;
12 }
13
14 int main()
15 {
16     stuct Vector v1, v2;
17     v1.x = 1.2;
18     v1.y = 3.6;
19     v2.x = v1.x * val;
20     v2.y = 6.7;
21
22     return dotMultiply((v1, v2);
23 }
```

输出

```
1  Error type B at line 2, column 10: syntax error
2  Error type A at Line 9, Column 35: Mysterious character ':'
3  Error type B at line 16, column 17: syntax error
```

```
Error type B at line 22, column 28: syntax error
```

说明：第 2 行声明中不可进行初始化；第 9 行逗号错写成冒号，可多报一个 B 类型错误；第 16 行 struct 拼写错误；第 22 行函数调用多一个左括号。

3 C 组测试用例

本组测试用例共 2 个，不包含任何错误，需要输出正确的语法树。除特殊说明外，应与给出的语法树完全相同。语法树打印错误酌情扣分。

3.1 C-1

输入

```
struct Product{
    int name;
} i ;
int main(){
    struct Type{
        struct Product id;
        int t;
    }type;
    i.name = 1;
    type.id.name = 2 + i.name * 5;
    type.t = 4;
    return type.id.name * type.t;
}
```

输出

```
Program (1)
  ExtDefList (1)
    ExtDef (1)
      Specifier (1)
        StructSpecifier (1)
          STRUCT
```

```
        OptTag (1)
          ID: Product
        LC
        DefList (2)
          Def (2)
            Specifier (2)
              TYPE: int
            DecList (2)
              Dec (2)
                VarDec (2)
                  ID: name
            SEMI
        RC
    ExtDecList (3)
      VarDec (3)
        ID: i
    SEMI
  ExtDefList (4)
    ExtDef (4)
      Specifier (4)
        TYPE: int
      FunDec (4)
        ID: main
        LP
        RP
      CompSt (4)
        LC
        DefList (5)
          Def (5)
            Specifier (5)
              StructSpecifier (5)
                STRUCT
```

```
                OptTag (5)
                  ID: Type
                LC
                DefList (6)
                  Def (6)
                    Specifier (6)
                      StructSpecifier (6)
                        STRUCT
                        Tag (6)
                          ID: Product
                    DecList (6)
                      Dec (6)
                        VarDec (6)
                          ID: id
                    SEMI
                  DefList (7)
                    Def (7)
                      Specifier (7)
                        TYPE: int
                      DecList (7)
                        Dec (7)
                          VarDec (7)
                            ID: t
                      SEMI
                RC
              DecList (8)
                Dec (8)
                  VarDec (8)
                    ID: type
              SEMI
            StmtList (9)
              Stmt (9)
```

```
          Exp (9)
            Exp (9)
              Exp (9)
                ID: i
              DOT
              ID: name
            ASSIGNOP
            Exp (9)
              INT: 1
          SEMI
        StmtList (10)
          Stmt (10)
            Exp (10)
              Exp (10)
                Exp (10)
                  Exp (10)
                    ID: type
                  DOT
                  ID: id
                DOT
                ID: name
              ASSIGNOP
              Exp (10)
                Exp (10)
                  INT: 2
                PLUS
                Exp (10)
                  Exp (10)
                    Exp (10)
                      ID: i
                    DOT
                    ID: name
```

```
            STAR
            Exp (10)
              INT: 5
        SEMI
      StmtList (11)
        Stmt (11)
          Exp (11)
            Exp (11)
              Exp (11)
                ID: type
              DOT
              ID: t
            ASSIGNOP
            Exp (11)
              INT: 4
          SEMI
        StmtList (12)
          Stmt (12)
            RETURN
            Exp (12)
              Exp (12)
                Exp (12)
                  Exp (12)
                    Exp (12)
                      ID: type
                    DOT
                    ID: id
                  DOT
                  ID: name
                STAR
                Exp (12)
                  Exp (12)
                    ID: type
```

```
                            DOT
                            ID: t
                        SEMI
            RC
```

说明：使用的空格可以换位 Tab，注意缩进。

3.2 C-2

输入

```
int a[10][5];
int sum(){
    int i, j, N = 10, M = 5;
    int result;
    while(i < 10)
    {
        while(j < 5)
        {
            result = result + a[i][j];
            j = j + 1;
        }
        i = i + 1;
    }
    return result;
}
int main() {
        int i1, j1, s;

        while(i1 < 10) {
            a[i1][0] = 1;
            a[i1][1] = 2 * i1;
            a[i1][2] = 1 + i1 * 2;
            a[i1][3] = i1;
            a[i1][4] = i1 * i1;
```

```
            i1 = i1 + 1;
        }

        return sum();
}
```

输出

```
Program (1)
  ExtDefList (1)
    ExtDef (1)
      Specifier (1)
        TYPE: int
      ExtDecList (1)
        VarDec (1)
          VarDec (1)
            VarDec (1)
              ID: a
            LB
            INT: 10
            RB
          LB
          INT: 5
          RB
      SEMI
    ExtDefList (2)
      ExtDef (2)
        Specifier (2)
          TYPE: int
        FunDec (2)
          ID: sum
          LP
          RP
        CompSt (2)
```

```
        LC
        DefList (3)
          Def (3)
            Specifier (3)
              TYPE: int
            DecList (3)
              Dec (3)
                VarDec (3)
                  ID: i
              COMMA
              DecList (3)
                Dec (3)
                  VarDec (3)
                    ID: j
                COMMA
                DecList (3)
                  Dec (3)
                    VarDec (3)
                      ID: N
                    ASSIGNOP
                    Exp (3)
                      INT: 10
                  COMMA
                  DecList (3)
                    Dec (3)
                      VarDec (3)
                        ID: M
                      ASSIGNOP
                      Exp (3)
                        INT: 5
            SEMI
          DefList (4)
```

```
    Def (4)
      Specifier (4)
        TYPE: int
      DecList (4)
        Dec (4)
          VarDec (4)
            ID: result
      SEMI
  StmtList (5)
    Stmt (5)
      WHILE
      LP
      Exp (5)
        Exp (5)
          ID: i
        RELOP
        Exp (5)
          INT: 10
      RP
      Stmt (6)
        CompSt (6)
          LC
          StmtList (7)
            Stmt (7)
              WHILE
              LP
              Exp (7)
                Exp (7)
                  ID: j
                RELOP
                Exp (7)
                  INT: 5
```

```
RP
Stmt (8)
  CompSt (8)
    LC
    StmtList (9)
      Stmt (9)
        Exp (9)
          Exp (9)
            ID: result
          ASSIGNOP
          Exp (9)
            Exp (9)
              ID: result
            PLUS
            Exp (9)
              Exp (9)
                Exp (9)
                  ID: a
                LB
                Exp (9)
                  ID: i
                RB
              LB
              Exp (9)
                ID: j
              RB
        SEMI
      StmtList (10)
        Stmt (10)
          Exp (10)
            Exp (10)
              ID: j
```

```
                                    ASSIGNOP
                                    Exp (10)
                                      Exp (10)
                                        ID: j
                                      PLUS
                                      Exp (10)
                                        INT: 1
                                  SEMI
                            RC
                      StmtList (12)
                        Stmt (12)
                          Exp (12)
                            Exp (12)
                              ID: i
                            ASSIGNOP
                            Exp (12)
                              Exp (12)
                                ID: i
                              PLUS
                              Exp (12)
                                INT: 1
                            SEMI
                  RC
            StmtList (14)
              Stmt (14)
                RETURN
                Exp (14)
                  ID: result
                SEMI
        RC
    ExtDefList (16)
      ExtDef (16)
```

```
Specifier (16)
  TYPE: int
FunDec (16)
  ID: main
  LP
  RP
CompSt (16)
  LC
  DefList (17)
    Def (17)
      Specifier (17)
        TYPE: int
      DecList (17)
        Dec (17)
          VarDec (17)
            ID: i1
        COMMA
        DecList (17)
          Dec (17)
            VarDec (17)
              ID: j1
          COMMA
          DecList (17)
            Dec (17)
              VarDec (17)
                ID: s
      SEMI
  StmtList (19)
    Stmt (19)
      WHILE
      LP
      Exp (19)
```

```
    Exp (19)
      ID: i1
    RELOP
    Exp (19)
      INT: 10
  RP
  Stmt (19)
    CompSt (19)
      LC
      StmtList (20)
        Stmt (20)
          Exp (20)
            Exp (20)
              Exp (20)
                Exp (20)
                  ID: a
                LB
                Exp (20)
                  ID: i1
                RB
              LB
              Exp (20)
                INT: 0
              RB
            ASSIGNOP
            Exp (20)
              INT: 1
          SEMI
        StmtList (21)
          Stmt (21)
            Exp (21)
              Exp (21)
```

```
          Exp (21)
            Exp (21)
              ID: a
            LB
            Exp (21)
              ID: i1
            RB
          LB
          Exp (21)
            INT: 1
          RB
        ASSIGNOP
        Exp (21)
          Exp (21)
            INT: 2
          STAR
          Exp (21)
            ID: i1
    SEMI
  StmtList (22)
    Stmt (22)
      Exp (22)
        Exp (22)
          Exp (22)
            Exp (22)
              ID: a
            LB
            Exp (22)
              ID: i1
            RB
          LB
          Exp (22)
```

```
                INT: 2
              RB
            ASSIGNOP
            Exp (22)
              Exp (22)
                Exp (22)
                  INT: 1
                PLUS
                Exp (22)
                  ID: i1
              STAR
              Exp (22)
                INT: 2
          SEMI
        StmtList (23)
          Stmt (23)
            Exp (23)
              Exp (23)
                Exp (23)
                  Exp (23)
                    ID: a
                  LB
                  Exp (23)
                    ID: i1
                  RB
                LB
                Exp (23)
                  INT: 3
                RB
              ASSIGNOP
              Exp (23)
                ID: i1
```

```
      SEMI
    StmtList (24)
      Stmt (24)
        Exp (24)
          Exp (24)
            Exp (24)
              Exp (24)
                Exp (24)
                  ID: a
                LB
                Exp (24)
                  ID: i1
                RB
              LB
              Exp (24)
                INT: 4
              RB
            ASSIGNOP
            Exp (24)
              Exp (24)
                ID: i1
              STAR
              Exp (24)
                ID: i1
        SEMI
      StmtList (25)
        Stmt (25)
          Exp (25)
            Exp (25)
              ID: i1
            ASSIGNOP
            Exp (25)
              Exp (25)
```

```
                                        ID: i1
                                    PLUS
                                    Exp (25)
                                      INT: 1
                                SEMI
                    RC
                StmtList (28)
                  Stmt (28)
                    RETURN
                    Exp (28)
                      ID: sum
                      LP
                      RP
                    SEMI
            RC
```

说明：考察对数组的翻译。

4 D 组测试用例

本组测试用例共 3 个，针对不同分组进行测试。对应分组的同学需要输出语法树，提示错误则不得分；其他分组的同学只要提示错误即可，如果打印了语法树，则将视为违规，将会倒扣分。

4.1 D-1

输入

```
int func_test()
{
    int _dec_ = 209;
    int _oct_ = 0115;
    int _dhex_ = 0x06Fab - _dec_number;
    int _result_ = - dhex + oct * ( dec - 0X000F );
}
```

输出

```
Program (1)
  ExtDefList (1)
    ExtDef (1)
      Specifier (1)
        TYPE: int
      FunDec (1)
        ID: func_test
        LP
        RP
      CompSt (2)
        LC
        DefList (3)
          Def (3)
            Specifier (3)
              TYPE: int
            DecList (3)
              Dec (3)
                VarDec (3)
                  ID: _dec_
                ASSIGNOP
                Exp (3)
                  INT: 209
            SEMI
          DefList (4)
            Def (4)
              Specifier (4)
                TYPE: int
              DecList (4)
                Dec (4)
                  VarDec (4)
                    ID: _oct_
                  ASSIGNOP
```

```
      Exp (4)
        INT: 77
    SEMI
  DefList (5)
    Def (5)
      Specifier (5)
        TYPE: int
      DecList (5)
        Dec (5)
          VarDec (5)
            ID: _dhex_
          ASSIGNOP
          Exp (5)
            Exp (5)
              INT: 28587
            MINUS
            Exp (5)
              ID: _dec_number
      SEMI
    DefList (6)
      Def (6)
        Specifier (6)
          TYPE: int
        DecList (6)
          Dec (6)
            VarDec (6)
              ID: _result_
            ASSIGNOP
            Exp (6)
              Exp (6)
                Exp (6)
                  MINUS
```

```
                                Exp (6)
                                  ID: dhex
                              PLUS
                              Exp (6)
                                ID: oct
                            STAR
                            Exp (6)
                              LP
                              Exp (6)
                                Exp (6)
                                  ID: dec
                                MINUS
                                Exp (6)
                                  INT: 15
                              RP
                    SEMI
          RC
```

说明：1.1 分组的同学需要输出该语法树，8 进制和 16 进制数必须正确转换（209、77、28587 和 15）；其他分组的同学只要提示有错误，而且不输出语法树即可。

4.2 D-2

输入

```
int float_test()
{
    float X_1 = .12E-3;
    float X_2 = 1.2e+3;
    float X_3 = 123.4E-4;
    float X_4 = 12.e+3;
    float X_5 = 0.123E4;
    float result = (1.2E1 + X_3) + X_1;
}
```

输出

```
Program (1)
  ExtDefList (1)
    ExtDef (1)
      Specifier (1)
        TYPE: int
      FunDec (1)
        ID: float_test
        LP
        RP
      CompSt (2)
        LC
        DefList (3)
          Def (3)
            Specifier (3)
              TYPE: float
            DecList (3)
              Dec (3)
                VarDec (3)
                  ID: X_1
                ASSIGNOP
                Exp (3)
                  FLOAT: 0.000120
            SEMI
          DefList (4)
            Def (4)
              Specifier (4)
                TYPE: float
              DecList (4)
                Dec (4)
                  VarDec (4)
                    ID: X_2
```

```
      ASSIGNOP
      Exp (4)
        FLOAT: 1200.000000
    SEMI
  DefList (5)
    Def (5)
      Specifier (5)
        TYPE: float
      DecList (5)
        Dec (5)
          VarDec (5)
            ID: X_3
          ASSIGNOP
          Exp (5)
            FLOAT: 0.012340
      SEMI
    DefList (6)
      Def (6)
        Specifier (6)
          TYPE: float
        DecList (6)
          Dec (6)
            VarDec (6)
              ID: X_4
            ASSIGNOP
            Exp (6)
              FLOAT: 12000.000000
        SEMI
      DefList (7)
        Def (7)
          Specifier (7)
            TYPE: float
```

```
                    DecList (7)
                      Dec (7)
                        VarDec (7)
                          ID: X_5
                        ASSIGNOP
                        Exp (7)
                          FLOAT: 1230.000000
                    SEMI
                  DefList (8)
                    Def (8)
                      Specifier (8)
                        TYPE: float
                      DecList (8)
                        Dec (8)
                          VarDec (8)
                            ID: result
                          ASSIGNOP
                          Exp (8)
                            Exp (8)
                              LP
                              Exp (8)
                                Exp (8)
                                  FLOAT: 12.000000
                                PLUS
                                Exp (8)
                                  ID: X_3
                              RP
                            PLUS
                            Exp (8)
                              ID: X_1
                      SEMI
        RC
```

说明：1.2 分组的同学需要输出语法树注意科学计数法浮点数的正确转换。其他分组同学只要提示出错，而且不输出语法树即可。

4.3 D-3

输入

```
/**
int a, b;
**comments
/\\\\\\\\\\\\\\\\\\\/
/
*/
int /*/**\function\*/maxFunction(int a, //int y) {
//parameters: a,b
int /*z*/ b){
        int z;
        /*****
        int z = a;
        ****\*/
        if(a > b) {/*{
                z = 1; a = 3;
        b = a * 2;/*}  \\\
        \\
        */      z = a;
        }
        else { //b > a comments*//\\//*
        //\\*//\\*//\\//*//\\*//\\//
                z = b;
        }
        return z/*ends\/\/\/\/\/\/*/;
}
```

输出

```
Program (7)
```

```
ExtDefList (7)
  ExtDef (7)
    Specifier (7)
      TYPE: int
    FunDec (7)
      ID: maxFunction
      LP
      VarList (7)
        ParamDec (7)
          Specifier (7)
            TYPE: int
          VarDec (7)
            ID: a
        COMMA
        VarList (9)
          ParamDec (9)
            Specifier (9)
              TYPE: int
            VarDec (9)
              ID: b
      RP
    CompSt (9)
      LC
      DefList (10)
        Def (10)
          Specifier (10)
            TYPE: int
          DecList (10)
            Dec (10)
              VarDec (10)
                ID: z
          SEMI
```

```
        StmtList (14)
          Stmt (14)
            IF
            LP
            Exp (14)
              Exp (14)
                ID: a
              RELOP
              Exp (14)
                ID: b
            RP
            Stmt (14)
              CompSt (14)
                LC
                StmtList (18)
                  Stmt (18)
                    Exp (18)
                      Exp (18)
                        ID: z
                      ASSIGNOP
                      Exp (18)
                        ID: a
                    SEMI
                RC
            ELSE
            Stmt (20)
              CompSt (20)
                LC
                StmtList (22)
                  Stmt (22)
                    Exp (22)
                      Exp (22)
```

```
                        ID: z
                      ASSIGNOP
                      Exp (22)
                        ID: b
                    SEMI
                RC
            StmtList (24)
              Stmt (24)
                RETURN
                Exp (24)
                  ID: z
                SEMI
        RC
```

说明：1.3 分组的同学需要输出语法树，不能提示有语法错误；其他分组同学只需提示有错误，且不输出语法树即可。

5 E 组测试用例

本组测试用例共 6 个，针对不同分组进行测试

5.1 E1.1

这组测试用例针对 1.1 分组的同学

输入（E1-1）

```
int test_for_wrong_oct_number()
{
    int _correct_oct_number_ = 007216;
    int _wrong_oct_number_ = 01826;
    int _correct_oct_number2_ = 0000126;
}
```

输出

```
Error type A at Line 4: Illegal octal number "01826"
```

说明：仅 1.1 分组同学需要测试这个用例，针对错误的 8 进制数 01826，识别成错误类型 B 也可以。

输入（E1-2）

```
int test_for_wrong_dhex_number()
{
    int _wrong_dhex_number_ = 0xFs396;
    int _wrong_dhex_number2_ = 0xxF396;
}
```

输出

```
Error type A at Line 3: Illegal hexadecimal number "0xFs396"
Error type A at Line 4: Illegal hexadecimal number "0xxF396"
```

说明：仅 1.1 分组同学需要测试这个用例，针对错误的 16 进制数 0xFs396 和 0xxF396，识别成错误类型 B 也可以。

5.2 E1.2

输入（E2-1）

```
float function()
{
    float x1 = .e-1;
    float x2 = 2.34e.-1;

    return x1 + x2;
}
```

输出

```
Error type A at line 3:  Illegal float number '.e-1'
Error type A at line 4:  Illegal float number '2.34e.'
```

说明：仅 1.2 分组同学需要测试这组用例，错误在于.e-1 和 2.34e.，识别成错误类型 B 也可以。

输入（E2-2）

```
float function()
```

```
{
    float a = e1.2;
    float b = e-;
}
```

输出

```
Error type A at line 3:  Illegal float number '.2'
Error type B at line 4: syntax error.  (unexpected near ';')
```

说明：仅 1.2 分组同学需要测试这组用例，错误在于指数是浮点数 1.2 和 -，识别成错误类型 B 和 A 也可以。

5.3 E1.3

输入（E3-1）

```
/*
* @code
*/

/*
* @param int a
*/

int func(int a) {
        int i = 1, N = 10;//\/*/\/\/\/\/\/\/ i = i +1;
        int sum;
        while(i /*>=<*/ <= N) {
                sum /*/ ~~ */= sum + i;
                i = i + 1;
        /*    TODO
                definition 12345
        */
        }

```

```
        put_int/*(*/(sum);   // assert(sum == ?);

        return /*-1;*/0;
}
/* end of function/*\*/
```

输出

```
Program (9)
  ExtDefList (9)
    ExtDef (9)
      Specifier (9)
        TYPE: int
      FunDec (9)
        ID: func
        LP
        VarList (9)
          ParamDec (9)
            Specifier (9)
              TYPE: int
            VarDec (9)
              ID: a
        RP
      CompSt (9)
        LC
        DefList (10)
          Def (10)
            Specifier (10)
              TYPE: int
            DecList (10)
              Dec (10)
                VarDec (10)
                  ID: i
                ASSIGNOP
```

```
      Exp (10)
        INT: 1
    COMMA
    DecList (10)
      Dec (10)
        VarDec (10)
          ID: N
        ASSIGNOP
        Exp (10)
          INT: 10
  SEMI
DefList (11)
  Def (11)
    Specifier (11)
      TYPE: int
    DecList (11)
      Dec (11)
        VarDec (11)
          ID: sum
    SEMI
StmtList (12)
  Stmt (12)
    WHILE
    LP
    Exp (12)
      Exp (12)
        ID: i
      RELOP
      Exp (12)
        ID: N
    RP
    Stmt (12)
```

```
    CompSt (12)
      LC
      StmtList (13)
        Stmt (13)
          Exp (13)
            Exp (13)
              ID: sum
            ASSIGNOP
            Exp (13)
              Exp (13)
                ID: sum
              PLUS
              Exp (13)
                ID: i
          SEMI
        StmtList (14)
          Stmt (14)
            Exp (14)
              Exp (14)
                ID: i
              ASSIGNOP
              Exp (14)
                Exp (14)
                  ID: i
                PLUS
                Exp (14)
                  INT: 1
            SEMI
      RC
  StmtList (20)
    Stmt (20)
      Exp (20)
```

```
                ID: put_int
                LP
                Args (20)
                  Exp (20)
                    ID: sum
                RP
              SEMI
            StmtList (22)
              Stmt (22)
                RETURN
                Exp (22)
                  INT: 0
                SEMI
        RC
```

说明：必须输出正确的语法树，否则该用例不得分

输入（E3-2）

```
/*
* comments
*/
int main() {
        int n, n2,/**/ n1, n0//\//*/;

        while(n < 10) {
                n2 = n / 1 + 2;
                /*This is a comment.\
                /\//\/******\/\/\/\/
                if(n == (n0)) {
                        put_int(n);
                }
                */
                n1 = (n / 10) + 10;
                n0 = n + 10;
```

```
17                  //put_int(n);\\*/;
18                  if(n == func(n2) * func(n1) + func(n0))
19                          put_int(/*(m+*/n));
20                  /*\}*\/*
21                  }*/
22                  n /*_1*/= n + 1;
23          }
24          return /**/0;
25  }
26
27
28  int func(int n) { //function
29          return n /*\*/* n /**\2*/* n;
30  }
31
32  /**end of function~
```

输出

```
1  Error type B at line 7, near 'while'
2  Error type B at line 19, near ')'
3  Error type A at Line 32: met EOF
```

说明：第 5 行分号被注释掉，也可以报错在第 5 或 7 行。第 19 行函数调用多一个括号。最后一个错误针对未终止的注释进行测试，如果打印了语法树，或者程序异常终止、死循环无法退出等，则该用例不得分。不限定错误类型以及提示方式，但是出错位置必须限定在 32 行或者以后的位置；直接提示“未终止的注释”也可以。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与王慧妍助教联系，注意同时抄送给许老师。