

编译原理第一次实验测试用例：目录

1	A 组测试用例	2
1.1	A-1	2
1.2	A-2	2
1.3	A-3	3
1.4	A-4	3
1.5	A-5	4
1.6	A-6	4
1.7	A-7	4
1.8	A-8	5
1.9	A-9	6
2	B 组测试用例	7
2.1	B-1	7
2.2	B-2	8
3	C 组测试用例	10
3.1	C-1	10
3.2	C-2	15
4	D 组测试用例	25
4.1	D-1	25
4.2	D-2	28
4.3	D-3	32
5	E 组测试用例	35
5.1	E1.1	35
5.2	E1.2	36
5.3	E1.3	37
6	结束语	43

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

```
1 int func_A1()
2 {
3     int _3_test;
4     float 6_wrong;
5 }
```

输出

```
1 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 6_wrong，注意标识符可以以下划线开始，所以第 3 行正确。

1.2 A-2

输入

```
1 int sub()
2 {
3     int a, b, c ;
4     c = 2;
5     a = b / c + 3 * a + 12;
6     b = !a && 4 >= 4 || a != 6;
7     c = a % b;
8     return b;
9 }
```

输出

```
1 Error type B at line 7: 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

```
1 int method() {  
2 {  
3     int a, b, c; {  
4         c = a + b * a + 4;  
5         b = a && c;  
6         return b; }  
7 }
```

输出

```
1 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 7 行，或者开始的括号所在的 1、2 行。

1.4 A-4

输入

```
1 int main() {  
2     int a;  
3     a = a + 1;  
4     a = a - 2;  
5     return a  
6 }
```

输出

```
1 Error type B at line 6: syntax error, before '}'
```

说明：第 5 行缺少分号，可以报错在第 5 行。

1.5 A-5

输入

```
1 int array (int a, int b)
2 {
3     int a1, b1[3], c[-3];
4     b1[a1] = b;
5     b1[1] = a + b1[a];
6 }
```

输出

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

说明：第3行数组定义格式错误，出现负数。

1.6 A-6

输入

```
1 int Array_Def()
2 {
3     int i, a[10][2], b;
4     i = 0;
5     while (i < b) {
6         a[i][3][3+1,2] = b;
7         a[5][1] = i;
8     }
9 }
```

输出

```
1 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，中括号内出现逗号。

1.7 A-7

输入

```

1 struct Product{
2     int name;
3     int price;
4     int date;
5 };
6 struct{
7     float test;
8 };
9 struct;
10 int main()
11 {
12     struct Product a;
13     a.name = a.price = 2;
14     a.data = 1;
15 }

```

输出

```

1 Error type B at line 9: syntax error, near ';'

```

说明：第 9 行定义结构体时缺少结构体名称。

1.8 A-8

输入

```

1 struct Product{
2     int name;
3     int price;
4     int date;
5 };
6 int main()
7 {
8     struct Inside_Function{
9         int name;

```

```

10     int price;
11 };
12 struct Product a;
13 a.name = a.price = 2;
14 a.data = 1;
15 }

```

输出

```

1 Error type B at line 11: syntax error, near '}'

```

说明：第 11 行，在方法体内定义结构体时缺少声明的变量名。

1.9 A-9

输入

```

1 int function_1(int a) {
2     return a + 3;
3 }
4
5 float function_2(int b)
6 {
7     int c;
8     c = b * 3 ;
9     if(c == 12) return 3.5;
10    else return 1.2;
11 }
12
13 float functionMain(int b)
14 {
15     int a = b;
16     return function_2(function_1((2 * b) + 3));
17 }

```

输出

```
1 Error type B at line 16: syntax error, near ') '
```

说明：第 10 行函数调用时多了一个右括号。

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、多报错误或者打印语法树都会导致扣分。

2.1 B-1

输入

```
1 struct Test_Product{
2     int name;
3     float type,
4 }p[10];
5
6 float Test_Function(struct Test_Product test) {
7     if(test.name >= 3 || test.type == 2.3) {
8         return 3.2;
9     } else{
10         return test.type /2 * 7 + test,type;
11     }
12 }
13
14 int main()
15 {
16     int name[10];
17     float type[10];
18     int result;
19     int i, N;
20     while(i != N){
21         p[i].name = name[i];
22         p[i].type = (type[i * 3] - 2.2) / 2.3;
```

```

23         i = i + 1;
24     }
25
26     while(i == 0) {
27         N = N + p[i].name;
28         p[i].type = p[1 0].type * p[9].type;
29     }
30     return N;
31 }

```

输出

```

1 Error type B at line 4: syntax error, near '}'
2 Error type B at line 10: syntax error, near ', '
3 Error type B at line 20: syntax error, near '!'
4 Error type B at line 28: syntax error, near '0'

```

说明：第3行末尾分号错写成逗号，也可以报错在第4行；第10行结构体域访问. 错写成,；第20行不等号中多了一个空格；第28行数组访问下标10之间多了一个空格。

2.2 B-2

输入

```

1 struct Result{
2     int i;
3     int j;
4 }text1, text2; text3;
5
6 int gcd(int a, int b) {
7     if (b == 0)
8         return a;
9     else
10        return gcd(b, a % b);
11 }
12

```



```

13 int main() {
14     struct Result six;
15     int x1, x2;
16
17     x1 = 1;
18     x2 = 7;
19     six.i = 7;
20     six.j = six.i + text1.i - 3;
21     if (six.j < text2.j) {
22         retun six.j + 3;
23     }
24
25     x1 = 14;
26     x2 = 21;
27     while(x1 < x2) {
28         x1 = x1+ gcd(x1, x2);
29     }
30
31     return x1 > 3 x2;
32 }

```

输出

```

1 Error type B at line 4: syntax error, near 'text3'
2 Error type A at line 10: Mysterious character '%'
3 Error type B at line 22: syntax error, near 'six'
4 Error type B at line 31: syntax error, near 'x2'

```

说明：第 4 行多个结构体变量声明中 text2 与 text3 中间逗号错写为分号；第 10 行使用未定义符号%，也可以多报一个 B 类型错误；第 22 行 return 拼写错误；第 29 行多出现一个数字 3。

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

```
1 struct Test_type{
2     int id;
3     float value;
4 }test1;
5
6 int main() {
7     struct Test_type_type{
8         struct Test_type nested1;
9         struct Test_type_nested{
10             int idd;
11         }nested2;
12     }type;
13
14     type.nested1 = test1;
15     type.nested2.idd = type.nested1.id;
16     return type.nested2.idd * type.nested1.id;
17 }
```

输出

```
1 Program(1)
2   ExtDefList(1)
3     ExtDef(1)
4       Specifier(1)
5         StructSpecifier(1)
6           STRUCT
7           OptTag(1)
8             ID:Test_type
9           LC
10          DefList(2)
11            Def(2)
```

```

12         Specifier(2)
13         TYPE:int
14         DecList(2)
15         Dec(2)
16         VarDec(2)
17         ID:id
18         SEMI
19         DefList(3)
20         Def(3)
21         Specifier(3)
22         TYPE:float
23         DecList(3)
24         Dec(3)
25         VarDec(3)
26         ID:value
27         SEMI
28         RC
29         ExtDecList(4)
30         VarDec(4)
31         ID:test1
32         SEMI
33         ExtDefList(6)
34         ExtDef(6)
35         Specifier(6)
36         TYPE:int
37         FunDec(6)
38         ID:main
39         LP
40         RP
41         CompSt(6)
42         LC
43         DefList(7)

```

```

44     Def (7)
45         Specifier (7)
46             StructSpecifier (7)
47                 STRUCT
48                 OptTag (7)
49                     ID:Test_type_type
50                 LC
51             DefList (8)
52                 Def (8)
53                     Specifier (8)
54                         StructSpecifier (8)
55                             STRUCT
56                             Tag (8)
57                                 ID:Test_type
58                         DecList (8)
59                             Dec (8)
60                                 VarDec (8)
61                                     ID:nested1
62                             SEMI
63                         DefList (9)
64                             Def (9)
65                                 Specifier (9)
66                                     StructSpecifier (9)
67                                         STRUCT
68                                         OptTag (9)
69                                             ID:Test_type_nested
70                                         LC
71                                         DefList (10)
72                                             Def (10)
73                                                 Specifier (10)
74                                                     TYPE:int
75                                                     DecList (10)

```

76	Dec(10)
77	VarDec(10)
78	ID:idd
79	SEMI
80	RC
81	DecList(11)
82	Dec(11)
83	VarDec(11)
84	ID:nested2
85	SEMI
86	RC
87	DecList(12)
88	Dec(12)
89	VarDec(12)
90	ID:type
91	SEMI
92	StmtList(14)
93	Stmt(14)
94	Exp(14)
95	Exp(14)
96	Exp(14)
97	ID:type
98	DOT
99	ID:nested1
100	ASSIGNOP
101	Exp(14)
102	ID:test1
103	SEMI
104	StmtList(15)
105	Stmt(15)
106	Exp(15)
107	Exp(15)

108	Exp (15)
109	Exp (15)
110	ID: type
111	DOT
112	ID: nested2
113	DOT
114	ID: idd
115	ASSIGNOP
116	Exp (15)
117	Exp (15)
118	Exp (15)
119	ID: type
120	DOT
121	ID: nested1
122	DOT
123	ID: id
124	SEMI
125	
126	StmtList(16)
127	Stmt (16)
128	RETURN
129	Exp (16)
130	Exp (16)
131	Exp (16)
132	Exp (16)
133	ID: type
134	DOT
135	ID: nested2
136	DOT
137	ID: idd
138	STAR
139	Exp (16)

```

140             Exp (16)
141             Exp (16)
142             ID: type
143             DOT
144             ID: nested1
145             DOT
146             ID: id
147             SEMI
148             RC

```

说明：使用的空格可以用 **Tab** 替换，注意缩进

3.2 C-2

输入

```

1  int map[10][10];
2  int dis[10][10];
3  int floyd(int num){
4      int i,j,k;
5      i = j = k = 0;
6      while(i < num) {
7          while(j < num) {
8              while(k < num) {
9                  if (dis[j][i] + dis[i][k] < dis[j][k]) {
10                     i = j + k;
11                     }
12                     k = k + 1;
13                 }
14                 j = j + 1;
15             }
16             i = i + 1;
17         }
18     return 1;
19 }

```

```

20
21 int main(){
22     int number, i;
23     number = 3;
24     dis = map;
25     floyd(number);
26     return number;
27 }

```

输出

```

1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         TYPE: int
6       ExtDecList (1)
7         VarDec (1)
8           VarDec (1)
9             VarDec (1)
10               ID: map
11               LB
12               INT: 10
13               RB
14             LB
15             INT: 10
16             RB
17         SEMI
18       ExtDefList (2)
19         ExtDef (2)
20           Specifier (2)
21             TYPE: int
22           ExtDecList (2)
23             VarDec (2)

```



```

24         VarDec (2)
25         VarDec (2)
26             ID: dis
27             LB
28             INT: 10
29             RB
30             LB
31             INT: 10
32             RB
33     SEMI
34 ExtDefList (3)
35     ExtDef (3)
36         Specifier (3)
37             TYPE: int
38     FunDec (3)
39         ID: floyd
40         LP
41         VarList (3)
42             ParamDec (3)
43                 Specifier (3)
44                     TYPE: int
45                 VarDec (3)
46                     ID: num
47             RP
48     CompSt (3)
49         LC
50         DefList (4)
51             Def (4)
52                 Specifier (4)
53                     TYPE: int
54                 DecList (4)
55                     Dec (4)

```

```

56         VarDec (4)
57         ID: i
58     COMMA
59     DecList (4)
60         Dec (4)
61         VarDec (4)
62         ID: j
63     COMMA
64     DecList (4)
65         Dec (4)
66         VarDec (4)
67         ID: k
68     SEMI
69 StmtList (5)
70     Stmt (5)
71     Exp (5)
72     Exp (5)
73     ID: i
74     ASSIGNOP
75     Exp (5)
76     Exp (5)
77     ID: j
78     ASSIGNOP
79     Exp (5)
80     Exp (5)
81     ID: k
82     ASSIGNOP
83     Exp (5)
84     INT: 0
85     SEMI
86 StmtList (6)
87     Stmt (6)

```

```

88      WHILE
89      LP
90      Exp (6)
91      Exp (6)
92      ID: i
93      RELOP
94      Exp (6)
95      ID: num
96      RP
97      Stmt (6)
98      CompSt (6)
99      LC
100     StmtList (7)
101     Stmt (7)
102     WHILE
103     LP
104     Exp (7)
105     Exp (7)
106     ID: j
107     RELOP
108     Exp (7)
109     ID: num
110     RP
111     Stmt (7)
112     CompSt (7)
113     LC
114     StmtList (8)
115     Stmt (8)
116     WHILE
117     LP
118     Exp (8)
119     Exp (8)

```

120	ID: k
121	RELOP
122	Exp (8)
123	ID: num
124	RP
125	Stmt (8)
126	CompSt (8)
127	LC
128	StmtList (9)
129	Stmt (9)
130	IF
131	LP
132	Exp (9)
133	Exp (9)
134	Exp (9)
135	Exp (9)
136	Exp (9)
137	ID: dis
138	LB
139	Exp (9)
140	ID: j
141	RB
142	LB
143	Exp (9)
144	ID: i
145	RB
146	PLUS
147	Exp (9)
148	Exp (9)
149	Exp (9)
150	ID: dis
151	LB

152	Exp (9)
153	ID: i
154	RB
155	LB
156	Exp (9)
157	ID: k
158	RB
159	RELOP
160	Exp (9)
161	Exp (9)
162	Exp (9)
163	ID: dis
164	LB
165	Exp (9)
166	ID: j
167	RB
168	LB
169	Exp (9)
170	ID: k
171	RB
172	RP
173	Stmt (9)
174	CompSt (9)
175	LC
176	StmtList (10)
177	Stmt (10)
178	Exp (10)
179	Exp (10)
180	ID: i
181	ASSIGNOP
182	Exp (10)
183	Exp (10)

184	ID: j
185	PLUS
186	Exp (10)
187	ID: k
188	SEMI
189	RC
190	StmtList (12)
191	Stmt (12)
192	Exp (12)
193	Exp (12)
194	ID: k
195	ASSIGNOP
196	Exp (12)
197	Exp (12)
198	ID: k
199	PLUS
200	Exp (12)
201	INT: 1
202	SEMI
203	RC
204	StmtList (14)
205	Stmt (14)
206	Exp (14)
207	Exp (14)
208	ID: j
209	ASSIGNOP
210	Exp (14)
211	Exp (14)
212	ID: j
213	PLUS
214	Exp (14)
215	INT: 1

```

216 SEMI
217 RC
218 StmtList (16)
219 Stmt (16)
220 Exp (16)
221 Exp (16)
222 ID: i
223 ASSIGNOP
224 Exp (16)
225 Exp (16)
226 ID: i
227 PLUS
228 Exp (16)
229 INT: 1
230 SEMI
231 RC
232 StmtList (18)
233 Stmt (18)
234 RETURN
235 Exp (18)
236 INT: 1
237 SEMI
238 RC
239 ExtDefList (21)
240 ExtDef (21)
241 Specifier (21)
242 TYPE: int
243 FunDec (21)
244 ID: main
245 LP
246 RP
247 CompSt (21)

```

```

248      LC
249      DefList (22)
250      Def (22)
251      Specifier (22)
252      TYPE: int
253      DecList (22)
254      Dec (22)
255      VarDec (22)
256      ID: number
257      COMMA
258      DecList (22)
259      Dec (22)
260      VarDec (22)
261      ID: i
262      SEMI
263      StmtList (23)
264      Stmt (23)
265      Exp (23)
266      Exp (23)
267      ID: number
268      ASSIGNOP
269      Exp (23)
270      INT: 3
271      SEMI
272      StmtList (24)
273      Stmt (24)
274      Exp (24)
275      Exp (24)
276      ID: dis
277      ASSIGNOP
278      Exp (24)
279      ID: map

```



```

280             SEMI
281         StmtList (25)
282             Stmt (25)
283                 Exp (25)
284                     ID: floyd
285                     LP
286                     Args (25)
287                         Exp (25)
288                             ID: number
289                             RP
290             SEMI
291         StmtList (26)
292             Stmt (26)
293                 RETURN
294                 Exp (26)
295                     ID: number
296             SEMI
297     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；其他分组的同学只需要在对应位置提示错误即可，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

```

1 int func_test()
2 {
3     int _dec_ = 312;
4     int _oct_ = 0733;

```

```

5      int _dhex_ = 0xC612D + _oct_;
6      int _result_ = - _dhex_ + _oct_ * ( _dec_ - 0X3aB );
7  }

```

输出

```

1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         TYPE: int
6       FunDec (1)
7         ID: func_test
8         LP
9         RP
10      CompSt (2)
11        LC
12        DefList (3)
13          Def (3)
14            Specifier (3)
15              TYPE: int
16            Declist (3)
17              Dec (3)
18                VarDec (3)
19                  ID: _dec_
20                  ASSIGNOP
21                  Exp (3)
22                    INT: 312
23          SEMI
24        DefList (4)
25          Def (4)
26            Specifier (4)
27              TYPE: int
28            Declist (4)

```

```

29         Dec (4)
30         VarDec (4)
31             ID: _oct_
32         ASSIGNOP
33         Exp (4)
34             INT: 475
35     SEMI
36 DefList (5)
37     Def (5)
38         Specifier (5)
39             TYPE: int
40         DecList (5)
41             Dec (5)
42                 VarDec (5)
43                     ID: _dhex_
44                 ASSIGNOP
45                 Exp (5)
46                     Exp (5)
47                         INT: 811309
48                     PLUS
49                     Exp (5)
50                         ID: _oct_
51     SEMI
52 DefList (6)
53     Def (6)
54         Specifier (6)
55             TYPE: int
56         DecList (6)
57             Dec (6)
58                 VarDec (6)
59                     ID: _result_
60                 ASSIGNOP

```

```

61          Exp (6)
62          Exp (6)
63          MINUS
64          Exp (6)
65          ID: _dhex_
66          PLUS
67          Exp (6)
68          Exp (6)
69          ID: _oct_
70          STAR
71          Exp (6)
72          LP
73          Exp (6)
74          Exp (6)
75          ID: _dec_
76          MINUS
77          Exp (6)
78          INT: 939
79          RP
80          SEMI
81          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（475、811309 和 939）；其他分组的同学只要提示相应的错误，而且不输出语法树即可。

4.2 D-2

输入

```

1 int float_test()
2 {
3     float X_1 = 1.E3;
4     float X_2 = 2.0e-4;
5     float X_3 = 12.3E-1;
6     float X_4 = 123.4e2;

```

```

7     float X_5 = .123E2;
8     float X_6 = 1.e+1;
9 }

```

输出

```

1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         TYPE: int
6       FunDec (1)
7         ID: float_test
8         LP
9         RP
10      CompSt (2)
11        LC
12        DefList (3)
13          Def (3)
14            Specifier (3)
15              TYPE: float
16            DecList (3)
17              Dec (3)
18                VarDec (3)
19                  ID: X_1
20                  ASSIGNOP
21                  Exp (3)
22                    FLOAT: 1000.000000
23              SEMI
24            DefList (4)
25              Def (4)
26                Specifier (4)
27                  TYPE: float
28                DecList (4)

```

```

29         Dec (4)
30         VarDec (4)
31         ID: X_2
32         ASSIGNOP
33         Exp (4)
34         FLOAT: 0.000200
35     SEMI
36 DefList (5)
37     Def (5)
38         Specifier (5)
39         TYPE: float
40         DecList (5)
41         Dec (5)
42         VarDec (5)
43         ID: X_3
44         ASSIGNOP
45         Exp (5)
46         FLOAT: 1.230000
47     SEMI
48 DefList (6)
49     Def (6)
50         Specifier (6)
51         TYPE: float
52         DecList (6)
53         Dec (6)
54         VarDec (6)
55         ID: X_4
56         ASSIGNOP
57         Exp (6)
58         FLOAT: 12340.000000
59     SEMI
60 DefList (7)

```

```

61         Def (7)
62             Specifier (7)
63                 TYPE: float
64             DecList (7)
65                 Dec (7)
66                     VarDec (7)
67                         ID: X_5
68                     ASSIGNOP
69                     Exp (7)
70                         FLOAT: 12.300000
71             SEMI
72     DefList (8)
73         Def (8)
74             Specifier (8)
75                 TYPE: float
76             DecList (8)
77                 Dec (8)
78                     VarDec (8)
79                         ID: X_6
80                     ASSIGNOP
81                     Exp (8)
82                         FLOAT: 10.000000
83             SEMI
84 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。其它分组同学只需要提示相应错误，而且不输出语法树即可。

4.3 D-3

输入

```

1  /* a comment */
2  // also a comment
3  // aaaa //

```

```

4  /*****
5  aasdfasdf
6  *****/
7  ///\\
8  *****/
9  int a, b;
10 /***
11 * main
12 * //ss
13 **/
14 int main() {
15     // int a,b
16     // /*\
17     int c;
18     c = a;
19     b = a + c + //aasdfdsb;
20     /*asdfasff */ b;
21     if(c <= b) { //b > a comments*///\\/*
22         ///\\*///\\*///\\/*///\\*///\\//
23         return c;
24     } else
25
26         return b;
27 }

```

输出

```

1 Program (9)
2   ExtDefList (9)
3     ExtDef (9)
4       Specifier (9)
5         TYPE: int
6       ExtDecList (9)
7         VarDec (9)

```



```

8         ID: a
9         COMMA
10        ExtDecList (9)
11        VarDec (9)
12        ID: b
13    SEMI
14    ExtDefList (14)
15    ExtDef (14)
16    Specifier (14)
17        TYPE: int
18    FunDec (14)
19        ID: main
20        LP
21        RP
22    CompSt (14)
23    LC
24    DefList (17)
25    Def (17)
26    Specifier (17)
27        TYPE: int
28    DecList (17)
29    Dec (17)
30    VarDec (17)
31        ID: c
32    SEMI
33    StmtList (18)
34    Stmt (18)
35    Exp (18)
36    Exp (18)
37        ID: c
38    ASSIGNOP
39    Exp (18)

```

40	ID: a
41	SEMI
42	StmtList (19)
43	Stmt (19)
44	Exp (19)
45	Exp (19)
46	ID: b
47	ASSIGNOP
48	Exp (19)
49	Exp (19)
50	Exp (19)
51	ID: a
52	PLUS
53	Exp (19)
54	ID: c
55	PLUS
56	Exp (20)
57	ID: b
58	SEMI
59	StmtList (21)
60	Stmt (21)
61	IF
62	LP
63	Exp (21)
64	Exp (21)
65	ID: c
66	RELOP
67	Exp (21)
68	ID: b
69	RP
70	Stmt (21)
71	CompSt (21)

```

72             LC
73             StmtList (23)
74             Stmt (23)
75             RETURN
76             Exp (23)
77             ID: c
78             SEMI
79             RC
80         ELSE
81         Stmt (26)
82         RETURN
83         Exp (26)
84         ID: b
85         SEMI
86     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；其他分组同学只需要提示相应错误，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）

```

1  int test_for_wrong_oct_number()
2  {
3      int _correct_oct_number_ = 003575;
4      int _decimal_number = 1836;
5      int _wrong_oct_number_ = 01836;
6      int _correct_oct_number2_ = 00001266;
7  }

```

输出

```
1 Error type A at Line 5: Illegal octal number "01836"
```

说明：仅 1.1 分组的同学需要测试这个用例，针对错误的 8 进制数 01836，识别成错误类型 B 也可以。

输入（E1-2）

```
1 int test_for_wrong_dhex_number()
2 {
3     int xF245 = 0;
4     int _correct_dhex_number = 0xF369;
5     int _wrong_dhex_number_ = 0xFG369;
6     int _wrong_dhex_number2_ = 0xx245;
7     int _correct_not_dhex_number3 = xF245;
8 }
```

输出

```
1 Error type A at Line 5: Illegal hexadecimal number "0xFG369"
2 Error type A at Line 6: Illegal hexadecimal number "0xx245"
```

说明：仅 1.1 分组的同学需要测试这个用例，针对错误的 16 进制数 0xFG369 与 0xx245，识别成错误类型 B 也可以。

5.2 E1.2

这组测试用例针对 1.2 分组的同学。

输入（E2-1）

```
1 float function()
2 {
3     float x4 = 1.e1;
4     float x1 = 4e;
5     float x2 = 1.e1.3;
6
7     return x1 + x2;
8 }
```

输出

```
1 Error type A at Line 4: Illegal float number "4e"  
2 Error type A at Line 5: Illegal float number "1.e1.3"
```

说明：仅 1.2 分组的同学需要测试这个用例，针对错误浮点数 4e 和 1.e1.3，识别成错误类型 B 也可以。

输入（E2-2）

```
1 float function()  
2 {  
3     float a = 12.34e.-4;  
4     float b = .e;  
5 }
```

输出

```
1 Error type A at Line 3: Illegal float number "12.34e.-4"  
2 Error type B at line 4: syntax error, unexpected DOT
```

说明：仅 1.2 分组的同学需要测试这个用例，针对错误浮点数 12.34e.-4 和 .e，识别成错误类型 B 和 A 也可以。

5.3 E1.3

这组测试用例针对 1.3 分组的同学。

输入（E3-1）

```
1 /*****  
2 // @Start  
3 ~^~^~@#%#@#%^  
4 **/  
5 /*  
6 // @param int a  
7 /*  
8     int i  
9 */  
10 int test(int a) {
```

```

11  return a * /**?jj\\\\ / // // // // */ 3;
12  }
13  /**@Override/**/
14  int method() {
15      int i = 3; // \\ /* \\ \\ \\ \\ \\ \\ \\ // i = i +1;
16      int sum // = 12;
17      /*^_^*/ = 3;
18      while(i <= /*test run!*/3) {
19          sum /*blast */= sum + i;
20          i = i * 2;
21          /*      TODO
22
23              */
24      }
25
26      test/*(*/(sum); // should be executed
27
28      return /*-1;*/sum/**i -1 +2;*/;
29  }
30  /* \\**//**\\*end of function/*\\*/

```

输出

```

1  Program (10)
2      ExtDefList (10)
3          ExtDef (10)
4              Specifier (10)
5                  TYPE: int
6              FunDec (10)
7                  ID: test
8                  LP
9                  VarList (10)
10                     ParamDec (10)
11                     Specifier (10)

```

```

12         TYPE: int
13     VarDec (10)
14         ID: a
15     RP
16 CompSt (10)
17     LC
18     StmtList (11)
19         Stmt (11)
20             RETURN
21             Exp (11)
22                 Exp (11)
23                     ID: a
24                     MUL
25                     Exp (11)
26                         INT: 3
27             SEMI
28     RC
29 ExtDefList (14)
30     ExtDef (14)
31         Specifier (14)
32             TYPE: int
33     FunDec (14)
34         ID: method
35     LP
36     RP
37 CompSt (14)
38     LC
39     DefList (15)
40         Def (15)
41             Specifier (15)
42                 TYPE: int
43         DecList (15)

```

```

44         Dec (15)
45         VarDec (15)
46             ID: i
47         ASSIGNOP
48         Exp (15)
49             INT: 3
50     SEMI
51 DefList (16)
52     Def (16)
53         Specifier (16)
54             TYPE: int
55         DecList (16)
56             Dec (16)
57                 VarDec (16)
58                     ID: sum
59                 ASSIGNOP
60                 Exp (17)
61                     INT: 3
62             SEMI
63 StmtList (18)
64     Stmt (18)
65         WHILE
66         LP
67         Exp (18)
68             Exp (18)
69                 ID: i
70             RELOP
71             Exp (18)
72                 INT: 3
73         RP
74         Stmt (18)
75             CompSt (18)

```


76	LC
77	StmtList (19)
78	Stmt (19)
79	Exp (19)
80	Exp (19)
81	ID: sum
82	ASSIGNOP
83	Exp (19)
84	Exp (19)
85	ID: sum
86	ADD
87	Exp (19)
88	ID: i
89	SEMI
90	StmtList (20)
91	Stmt (20)
92	Exp (20)
93	Exp (20)
94	ID: i
95	ASSIGNOP
96	Exp (20)
97	Exp (20)
98	ID: i
99	MUL
100	Exp (20)
101	INT: 2
102	SEMI
103	RC
104	StmtList (26)
105	Stmt (26)
106	Exp (26)
107	ID: test

```

108         LP
109         Args (26)
110         Exp (26)
111         ID: sum
112         RP
113         SEMI
114         StmtList (28)
115         Stmt (28)
116         RETURN
117         Exp (28)
118         ID: sum
119         SEMI
120     RC

```

说明：仅 1.3 分组的同学需要测试这个用例，必须输出正确的语法树。

输入（E3-2）

```

1  /**comment
2  still
3  */
4  int test() {
5      int x;
6      float p;
7      if (x == 0){ //sadfasfd
8  */
9          return x;
10     }
11     else return x * 3;
12 //}
13 int main() {
14     int x,y //;
15     x = 3;
16     while (x > /*4*/3) {
17         y = y + /*\//\//\//\*//\//\****\*/ 1;

```

```

18     }
19     while ( /* (* / x > 1 ) ) {
20         y = y - 2 * 3 / 4 + x;
21     x = x + 1;
22     }
23     return y;
24 }
25 /* end? ~^

```

输出

```

1 Error type B at 8: syntax error, near '*/'
2 Error type B at 13: syntax error, near 'int'
3 Error type B at 15: syntax error, near 'x'
4 Error type B at 19: syntax error, near '{'
5 Error type B at 26: syntax error, unexpected $end

```

说明：仅 1.3 分组的同学需要测试这个用例。第 8 行出现一个未匹配的 */；第 12 行的右大括号被注释掉，报在第 13 行也可以；第 14 行分号被注释掉，报在第 15 行也可以。第 19 行 while 语句中第二个左括号被注释掉。最后一个错误针对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 25 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，可以写邮件与王珏助教联系，注意同时抄送给许老师。