

## 2017 年春季学期编译原理第一次实验测试用例：目录

<b>1</b>	<b>A 组测试用例</b>	<b>2</b>
1.1	A-1 . . . . .	2
1.2	A-2 . . . . .	2
1.3	A-3 . . . . .	3
1.4	A-4 . . . . .	3
1.5	A-5 . . . . .	3
1.6	A-6 . . . . .	4
1.7	A-7 . . . . .	4
1.8	A-8 . . . . .	5
1.9	A-9 . . . . .	5
<b>2</b>	<b>B 组测试用例</b>	<b>6</b>
2.1	B-1 . . . . .	6
2.2	B-2 . . . . .	8
<b>3</b>	<b>C 组测试用例</b>	<b>9</b>
3.1	C-1 . . . . .	9
3.2	C-2 . . . . .	14
<b>4</b>	<b>D 组测试用例</b>	<b>25</b>
4.1	D-1 . . . . .	25
4.2	D-2 . . . . .	28
4.3	D-3 . . . . .	32
<b>5</b>	<b>E 组测试用例</b>	<b>35</b>
5.1	E1.1 . . . . .	35
5.2	E1.2 . . . . .	36
5.3	E1.3 . . . . .	37
<b>6</b>	<b>结束语</b>	<b>42</b>

## 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 a1, _aaa_aa2;
4     float 3f;
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 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

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

输出

```
1 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

```
1 int method() {  
2 {  
3     int a, b, c;  
4     c = a + b * a;  
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，可以报错在第七行，或者开始的括号所在的第一二行。

### 1.4 A-4

输入

```
1 int main() {  
2     int a, b;  
3     a = b + 1;  
4     b = a + b  
5 }
```

输出

```
1 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

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

```

4     b1[a] = b;
5     b1[a] = a1 + 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。

## 1.6 A-6

输入

```

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

```

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，中括号内不可为空。

## 1.7 A-7

输入

```

1 struct Product{
2     int name;
3     int price;
4     int date;

```

```

5 };
6 int main()
7 {
8     Product a;
9     a.name = a.price = 2;
10    a.data = 1;
11 }

```

输出

```

1 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

```

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

```

输出

```

1 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

```

1 int function_1(int a)
2 {
3     return a + 5;
4 }
5
6 int function_2(int b)
7 {
8     c = function_1((2 * b);
9     return c;
10 }

```

输出

```

1 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

```

1 struct Product
2 {
3     int price;
4     int weight;
5 }p[10],
6
7 int Calculate(struct Product product)
8 {
9     if(product.weight < 1)
10         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
```



```
4 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

```
1 struct Product{
2     int name;
3 } i ;
4 int main(){
5     struct Type{
6         struct Product id;
7         int t;
8     }type;
9     i.name = 1;
10    type.id.name = 2 + i.name * 5;
11    type.t = 4;
12    return type.id.name * type.t;
13 }
```

输出

```
1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         StructSpecifier (1)
6           STRUCT
```

```

7      OptTag (1)
8      ID: Product
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: name
18             SEMI
19         RC
20     ExtDecList (3)
21         VarDec (3)
22             ID: i
23     SEMI
24 ExtDefList (4)
25     ExtDef (4)
26         Specifier (4)
27             TYPE: int
28     FunDec (4)
29         ID: main
30         LP
31         RP
32     CompSt (4)
33         LC
34         DefList (5)
35             Def (5)
36                 Specifier (5)
37                     StructSpecifier (5)
38                         STRUCT

```

```

39         OptTag (5)
40         ID: Type
41     LC
42     DefList (6)
43         Def (6)
44             Specifier (6)
45                 StructSpecifier (6)
46                     STRUCT
47                     Tag (6)
48                         ID: Product
49                 DecList (6)
50                     Dec (6)
51                         VarDec (6)
52                             ID: id
53             SEMI
54         DefList (7)
55             Def (7)
56                 Specifier (7)
57                     TYPE: int
58                 DecList (7)
59                     Dec (7)
60                         VarDec (7)
61                             ID: t
62             SEMI
63         RC
64     DecList (8)
65         Dec (8)
66         VarDec (8)
67             ID: type
68     SEMI
69 StmtList (9)
70     Stmt (9)

```

71	Exp (9)
72	Exp (9)
73	Exp (9)
74	ID: i
75	DOT
76	ID: name
77	ASSIGNOP
78	Exp (9)
79	INT: 1
80	SEMI
81	StmtList (10)
82	Stmt (10)
83	Exp (10)
84	Exp (10)
85	Exp (10)
86	Exp (10)
87	ID: type
88	DOT
89	ID: id
90	DOT
91	ID: name
92	ASSIGNOP
93	Exp (10)
94	Exp (10)
95	INT: 2
96	PLUS
97	Exp (10)
98	Exp (10)
99	Exp (10)
100	ID: i
101	DOT
102	ID: name

103	STAR
104	Exp (10)
105	INT: 5
106	SEMI
107	StmtList (11)
108	Stmt (11)
109	Exp (11)
110	Exp (11)
111	Exp (11)
112	ID: type
113	DOT
114	ID: t
115	ASSIGNOP
116	Exp (11)
117	INT: 4
118	SEMI
119	StmtList (12)
120	Stmt (12)
121	RETURN
122	Exp (12)
123	Exp (12)
124	Exp (12)
125	Exp (12)
126	ID: type
127	DOT
128	ID: id
129	DOT
130	ID: name
131	STAR
132	Exp (12)
133	Exp (12)
134	ID: type

135	DOT
136	ID: t
137	SEMI
138	RC

说明：使用的空格可以换位 Tab，注意缩进。

## 3.2 C-2

输入

```

1  int a[10][5];
2  int sum() {
3      int i, j, N = 10, M = 5;
4      int result;
5      while(i < 10)
6      {
7          while(j < 5)
8          {
9              result = result + a[i][j];
10             j = j + 1;
11         }
12         i = i + 1;
13     }
14     return result;
15 }
16 int main() {
17     int i1, j1, s;
18
19     while(i1 < 10) {
20         a[i1][0] = 1;
21         a[i1][1] = 2 * i1;
22         a[i1][2] = 1 + i1 * 2;
23         a[i1][3] = i1;
24         a[i1][4] = i1 * i1;

```

```

25         i1 = i1 + 1;
26     }
27
28     return sum();
29 }

```

## 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: a
11              LB
12              INT: 10
13              RB
14              LB
15              INT: 5
16              RB
17       SEMI
18   ExtDefList (2)
19     ExtDef (2)
20       Specifier (2)
21         TYPE: int
22       FunDec (2)
23         ID: sum
24         LP
25         RP
26       CompSt (2)

```

```

27      LC
28      DefList (3)
29          Def (3)
30              Specifier (3)
31                  TYPE: int
32              DecList (3)
33                  Dec (3)
34                      VarDec (3)
35                          ID: i
36                  COMMA
37              DecList (3)
38                  Dec (3)
39                      VarDec (3)
40                          ID: j
41                  COMMA
42              DecList (3)
43                  Dec (3)
44                      VarDec (3)
45                          ID: N
46                  ASSIGNOP
47                      Exp (3)
48                          INT: 10
49                  COMMA
50              DecList (3)
51                  Dec (3)
52                      VarDec (3)
53                          ID: M
54                  ASSIGNOP
55                      Exp (3)
56                          INT: 5
57          SEMI
58      DefList (4)

```



```

59         Def (4)
60             Specifier (4)
61                 TYPE: int
62             DecList (4)
63                 Dec (4)
64                     VarDec (4)
65                         ID: result
66             SEMI
67 StmtList (5)
68     Stmt (5)
69         WHILE
70             LP
71             Exp (5)
72                 Exp (5)
73                     ID: i
74             RELOP
75             Exp (5)
76                 INT: 10
77             RP
78 Stmt (6)
79     CompSt (6)
80         LC
81         StmtList (7)
82             Stmt (7)
83                 WHILE
84                     LP
85                     Exp (7)
86                         Exp (7)
87                             ID: j
88                     RELOP
89                     Exp (7)
90                         INT: 5

```

91	RP
92	Stmt (8)
93	CompSt (8)
94	LC
95	StmtList (9)
96	Stmt (9)
97	Exp (9)
98	Exp (9)
99	ID: result
100	ASSIGNOP
101	Exp (9)
102	Exp (9)
103	ID: result
104	PLUS
105	Exp (9)
106	Exp (9)
107	Exp (9)
108	ID: a
109	LB
110	Exp (9)
111	ID: i
112	RB
113	LB
114	Exp (9)
115	ID: j
116	RB
117	SEMI
118	StmtList (10)
119	Stmt (10)
120	Exp (10)
121	Exp (10)
122	ID: j

123	ASSIGNOP
124	Exp (10)
125	Exp (10)
126	ID: j
127	PLUS
128	Exp (10)
129	INT: 1
130	SEMI
131	RC
132	StmtList (12)
133	Stmt (12)
134	Exp (12)
135	Exp (12)
136	ID: i
137	ASSIGNOP
138	Exp (12)
139	Exp (12)
140	ID: i
141	PLUS
142	Exp (12)
143	INT: 1
144	SEMI
145	RC
146	StmtList (14)
147	Stmt (14)
148	RETURN
149	Exp (14)
150	ID: result
151	SEMI
152	RC
153	ExtDefList (16)
154	ExtDef (16)

```

155     Specifier (16)
156         TYPE: int
157     FunDec (16)
158         ID: main
159         LP
160         RP
161     CompSt (16)
162         LC
163         DefList (17)
164             Def (17)
165                 Specifier (17)
166                     TYPE: int
167                 DecList (17)
168                     Dec (17)
169                     VarDec (17)
170                         ID: i1
171                     COMMA
172                     DecList (17)
173                         Dec (17)
174                         VarDec (17)
175                             ID: j1
176                         COMMA
177                         DecList (17)
178                             Dec (17)
179                             VarDec (17)
180                                 ID: s
181                     SEMI
182         StmtList (19)
183             Stmt (19)
184                 WHILE
185                 LP
186                 Exp (19)

```

187	Exp (19)
188	ID: i1
189	RELOP
190	Exp (19)
191	INT: 10
192	RP
193	Stmt (19)
194	CompSt (19)
195	LC
196	StmtList (20)
197	Stmt (20)
198	Exp (20)
199	Exp (20)
200	Exp (20)
201	Exp (20)
202	ID: a
203	LB
204	Exp (20)
205	ID: i1
206	RB
207	LB
208	Exp (20)
209	INT: 0
210	RB
211	ASSIGNOP
212	Exp (20)
213	INT: 1
214	SEMI
215	StmtList (21)
216	Stmt (21)
217	Exp (21)
218	Exp (21)

219	Exp (21)
220	Exp (21)
221	ID: a
222	LB
223	Exp (21)
224	ID: i1
225	RB
226	LB
227	Exp (21)
228	INT: 1
229	RB
230	ASSIGNOP
231	Exp (21)
232	Exp (21)
233	INT: 2
234	STAR
235	Exp (21)
236	ID: i1
237	SEMI
238	StmtList (22)
239	Stmt (22)
240	Exp (22)
241	Exp (22)
242	Exp (22)
243	Exp (22)
244	ID: a
245	LB
246	Exp (22)
247	ID: i1
248	RB
249	LB
250	Exp (22)

251	INT: 2
252	RB
253	ASSIGNOP
254	Exp (22)
255	Exp (22)
256	Exp (22)
257	INT: 1
258	PLUS
259	Exp (22)
260	ID: i1
261	STAR
262	Exp (22)
263	INT: 2
264	SEMI
265	StmtList (23)
266	Stmt (23)
267	Exp (23)
268	Exp (23)
269	Exp (23)
270	Exp (23)
271	ID: a
272	LB
273	Exp (23)
274	ID: i1
275	RB
276	LB
277	Exp (23)
278	INT: 3
279	RB
280	ASSIGNOP
281	Exp (23)
282	ID: i1

283	SEMI
284	StmtList (24)
285	Stmt (24)
286	Exp (24)
287	Exp (24)
288	Exp (24)
289	Exp (24)
290	ID: a
291	LB
292	Exp (24)
293	ID: i1
294	RB
295	LB
296	Exp (24)
297	INT: 4
298	RB
299	ASSIGNOP
300	Exp (24)
301	Exp (24)
302	ID: i1
303	STAR
304	Exp (24)
305	ID: i1
306	SEMI
307	StmtList (25)
308	Stmt (25)
309	Exp (25)
310	Exp (25)
311	ID: i1
312	ASSIGNOP
313	Exp (25)
314	Exp (25)



```

315                                ID: i1
316                                PLUS
317                                Exp (25)
318                                INT: 1
319                                SEMI
320                                RC
321                                StmtList (28)
322                                Stmt (28)
323                                RETURN
324                                Exp (28)
325                                ID: sum
326                                LP
327                                RP
328                                SEMI
329                                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

```

1 int func_test()
2 {
3     int _dec_ = 209;
4     int _oct_ = 0115;
5     int _dhex_ = 0x06Fab - _dec_number;
6     int _result_ = - dhex + oct * ( dec - 0X000F );
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: 209
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: 77
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: 28587
48             MINUS
49             Exp (5)
50                 ID: _dec_number
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                     Exp (6)
64                         MINUS

```

```

65             Exp (6)
66             ID: dhex
67             PLUS
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: 15
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（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

```

1  int float_test()
2  {
3      float X_1 = .12E-3;
4      float X_2 = 1.2e+3;
5      float X_3 = 123.4E-4;
6      float X_4 = 12.e+3;
7      float X_5 = 0.123E4;
8      float result = (1.2E1 + X_3) + X_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: 0.000120
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: 1200.000000
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: 0.012340
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: 12000.000000
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: 1230.000000
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: result
80	ASSIGNOP
81	Exp (8)
82	Exp (8)
83	LP
84	Exp (8)
85	Exp (8)
86	FLOAT: 12.000000
87	PLUS
88	Exp (8)
89	ID: X_3
90	RP
91	PLUS
92	Exp (8)
93	ID: X_1
94	SEMI
95	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

```
1  /**
2  int a, b;
3  **comments
4  /\/////////////////////////////////
5  /
6  */
7  int /*/**\function\*/maxFunction(int a, //int y) {
8  //parameters: a,b
9  int /*z*/ b) {
10     int z;
11     /*****
12     int z = a;
13     *****\*/
14     if(a > b) {/*{
15         z = 1; a = 3;
16     b = a * 2; /*}  \\\
17     \\\
18     */      z = a;
19     }
20     else { //b > a comments*//\\/*
21     //\\*//\\*//\\/*//\\*//\\/*
22         z = b;
23     }
24     return z/*ends\\//\\//\\//\\/*/;
25 }
```

输出

```
1 Program (7)
```



```

2  ExtDefList (7)
3      ExtDef (7)
4          Specifier (7)
5              TYPE: int
6          FunDec (7)
7              ID: maxFunction
8              LP
9              VarList (7)
10                 ParamDec (7)
11                     Specifier (7)
12                         TYPE: int
13                 VarDec (7)
14                     ID: a
15                 COMMA
16                 VarList (9)
17                     ParamDec (9)
18                         Specifier (9)
19                             TYPE: int
20                     VarDec (9)
21                         ID: b
22             RP
23  CompSt (9)
24      LC
25      DefList (10)
26          Def (10)
27              Specifier (10)
28                  TYPE: int
29              DecList (10)
30                  Dec (10)
31                      VarDec (10)
32                          ID: z
33          SEMI

```

```

34      StmtList (14)
35      Stmt (14)
36      IF
37      LP
38      Exp (14)
39      Exp (14)
40      ID: a
41      RELOP
42      Exp (14)
43      ID: b
44      RP
45      Stmt (14)
46      CompSt (14)
47      LC
48      StmtList (18)
49      Stmt (18)
50      Exp (18)
51      Exp (18)
52      ID: z
53      ASSIGNOP
54      Exp (18)
55      ID: a
56      SEMI
57      RC
58      ELSE
59      Stmt (20)
60      CompSt (20)
61      LC
62      StmtList (22)
63      Stmt (22)
64      Exp (22)
65      Exp (22)

```

```

66             ID: z
67             ASSIGNOP
68             Exp (22)
69             ID: b
70             SEMI
71             RC
72         StmtList (24)
73         Stmt (24)
74         RETURN
75         Exp (24)
76         ID: z
77         SEMI
78     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）

```

1  int test_for_wrong_oct_number()
2  {
3      int _correct_oct_number_ = 007216;
4      int _wrong_oct_number_ = 01826;
5      int _correct_oct_number2_ = 0000126;
6  }

```

输出

```

1  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）

```
1 int test_for_wrong_dhex_number()  
2 {  
3     int _wrong_dhex_number_ = 0xFs396;  
4     int _wrong_dhex_number2_ = 0xxF396;  
5 }
```

输出

```
1 Error type A at Line 3: Illegal hexadecimal number "0xFs396"  
2 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）

```
1 float function()  
2 {  
3     float x1 = .e-1;  
4     float x2 = 2.34e.-1;  
5  
6     return x1 + x2;  
7 }
```

输出

```
1 Error type A at line 3: Illegal float number '.e-1'  
2 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）

```
1 float function()
```

```
2 {  
3     float a = e1.2;  
4     float b = e-;  
5 }
```

输出

```
1 | Error type A at line 3:  Illegal float number '.2'
```

```
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)

```

1  /*
2  * @code
3  */
4
5  /*
6  * @param int a
7  */
8
9  int func(int a) {
10     int i = 1, N = 10; //\\*\\/\\\\*\\/\\\\*\\/\\\\*\\/\\\\
    i = i + 1;
11     int sum;
12     while(i /*>=<*/ <= N) {
13         sum /*/ ~~ */= sum + i;
14         i = i + 1;
15     /*    TODO
16         definition 12345
17     */
18 }
19

```

```

20     put_int/* */(sum);  // assert(sum == ?);
21
22     return /*-1;*/0;
23 }
24 /* end of function/*\*/

```

## 输出

```

1 Program (9)
2   ExtDefList (9)
3     ExtDef (9)
4       Specifier (9)
5         TYPE: int
6       FunDec (9)
7         ID: func
8         LP
9         VarList (9)
10          ParamDec (9)
11            Specifier (9)
12              TYPE: int
13            VarDec (9)
14              ID: a
15          RP
16        CompSt (9)
17          LC
18          DefList (10)
19            Def (10)
20              Specifier (10)
21                TYPE: int
22              DecList (10)
23                Dec (10)
24                  VarDec (10)
25                    ID: i
26                  ASSIGNOP

```

```

27         Exp (10)
28         INT: 1
29     COMMA
30     DecList (10)
31         Dec (10)
32         VarDec (10)
33             ID: N
34         ASSIGNOP
35         Exp (10)
36         INT: 10
37     SEMI
38 DefList (11)
39     Def (11)
40         Specifier (11)
41             TYPE: int
42         DecList (11)
43         Dec (11)
44         VarDec (11)
45             ID: sum
46     SEMI
47 StmtList (12)
48     Stmt (12)
49         WHILE
50         LP
51         Exp (12)
52         Exp (12)
53             ID: i
54         RELOP
55         Exp (12)
56             ID: N
57     RP
58     Stmt (12)

```

```

59         CompSt (12)
60         LC
61         StmtList (13)
62         Stmt (13)
63         Exp (13)
64         Exp (13)
65         ID: sum
66         ASSIGNOP
67         Exp (13)
68         Exp (13)
69         ID: sum
70         PLUS
71         Exp (13)
72         ID: i
73         SEMI
74         StmtList (14)
75         Stmt (14)
76         Exp (14)
77         Exp (14)
78         ID: i
79         ASSIGNOP
80         Exp (14)
81         Exp (14)
82         ID: i
83         PLUS
84         Exp (14)
85         INT: 1
86         SEMI
87         RC
88     StmtList (20)
89     Stmt (20)
90     Exp (20)

```



```

91         ID: put_int
92         LP
93         Args (20)
94         Exp (20)
95         ID: sum
96         RP
97         SEMI
98         StmtList (22)
99         Stmt (22)
100        RETURN
101        Exp (22)
102        INT: 0
103        SEMI
104    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）

```

1  /*
2  * comments
3  */
4  int main() {
5      int n, n2, /**/ n1, n0//\\/**/;
6
7      while(n < 10) {
8          n2 = n / 1 + 2;
9          /*This is a comment.\
10         /\//\/***\//\//\//
11         if(n == (n0)) {
12             put_int(n);
13         }
14         */
15         n1 = (n / 10) + 10;
16         n0 = n + 10;

```

```

17         //put_int(n);\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。