

编译原理第一次实验测试用例：目录

1	A 组测试用例	3
1.1	A-1	3
1.2	A-2	3
1.3	A-3	4
1.4	A-4	4
1.5	A-5	5
1.6	A-6	5
1.7	A-7	6
1.8	A-8	7
1.9	A-9	7
1.10	A-10	8
2	B 组测试用例	10
2.1	B-1	10
2.2	B-2	12
3	C 组测试用例	13
3.1	C-1	13
3.2	C-2	27
4	D 组测试用例	47
4.1	D-1	47
4.2	D-2	49
4.3	D-3	52
5	E 组测试用例	57
5.1	E1-1	57
5.2	E1-2	58
5.3	E2-1	59
5.4	E2-2	60
5.5	E3-1	63
5.6	E3-2	64

1 A 组测试用例

本组测试用例共 10 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 main () {  
2     int i = 1.100;  
3     float f = 1;  
4     int j = i * f + 1.00.1;  
5 }
```

输出

```
1 Error type A at line 4: Illegal number '1.00.1'
```

说明：也可以报成 B 类错误。

1.2 A-2

输入

```
1 int f0o() {  
2     return 0;  
3 }  
4  
5 int main() {  
6     int li = f0o();  
7 }
```

输出

```
1 Error type A at line 6: Illegal identifier 'li'
```

说明：也可以报成 B 类错误。

1.3 A-3

输入

```
1 struct {  
2     int x;  
3     int y;  
4     int z;  
5 } s1, s2;  
6  
7 int foo() {  
8     struct s int;  
9 }
```

输出

```
1 Error type B at line 8: Expect identifier;
```

说明：关键字不能作为标识符。

1.4 A-4

输入

```
1 int i, j, k;  
2 int a[100];  
3  
4 float f = 1.00;  
5  
6 int hello() {  
7     int a[100];  
8 }
```

输出

```
1 Error type B at line 4: Global variable cannot have initializer;
```

说明：全局变量定义时不能初始化。

1.5 A-5

输入

```
1 int foo() {  
2     int i, j;  
3     i = 0;  
4     j = (i * 5 + 42) / 23;  
5     float f;  
6     return i;  
7 }
```

输出

```
1 Error type B at line 5: Unexpected variable declaration
```

说明：变量声明的部分必须在语句块的开始部分。

1.6 A-6

输入

```
1 struct Car {  
2     int color;  
3     float weight;  
4 };  
5  
6 struct Fee {  
7     Car car;  
8     int count;  
9 };  
10  
11 int main() {  
12     struct Car car;  
13     car.color = 0;  
14     car.weight = 1.111;  
15 }
```

输出

```
1 Error type B at line 7: Illegal type specifier
```

说明：缺少 `struct` 关键字。

1.7 A-7

输入

```
1 int a1[100];
2 int a2[100];
3
4 int init(int a[100]) {
5     int i;
6     while (i < 100) {
7         a[i] = i;
8         i = i + 1;
9     }
10 }
11
12 int add(int a[100], int b[100]) {
13     int i = 0;
14     int res[100];
15     while (i < 100) {
16         res[i] = a[i] + b[i];
17         i += 1;
18     }
19 }
20
21 int main() {
22     init(a1);
23     init(a2);
24     add(a1, a2);
25 }
```

输出

```
1 Error type B at line 17: '+= ' is not supported
```

说明：C-没有“+=”运算符。

1.8 A-8

输入

```
1 struct Oops say() {  
2     struct Point {  
3         int x;  
4         int y;  
5         int z;  
6     } p1, p2;  
7     return 0;  
8 }  
9  
10 int;  
11 int;  
12  
13 struct st {  
14     int s1;  
15     float s2;  
16     struct st s3, s4;  
17     int arr[10.0];  
18 };
```

输出

```
1 Error type B at line 17: array length can only be integer
```

说明：数组的长度必须为整数类型。

1.9 A-9

输入

```

1  int turn;
2  int flag[2];
3
4  int doSth() {
5      int i = 0;
6      while (i < 42) {
7          i = i + 1;
8      }
9  }
10
11 int t1() {
12     turn = 1;
13     while (flag[1] == 1 && turn == 1) {
14
15     }
16     doSth();
17     flag[0] = 0;
18 }
19
20 int t2() {
21     turn = 0;
22     while (flag[0] == 1 && turn == 0);
23     doSth();
24     flag[1] = 0;
25 }

```

输出

```

1 Error type B at line 22: Expect statement after while condition

```

说明：C-中没有空语句。

1.10 A-10

输入


```
1  int a[10][10];
2  int b[10][10];
3
4  struct container {
5      int res;
6      int cnt;
7      float avg;
8      int sum[10];
9  };
10
11 int comp(int a[10][10], int b[10][10]) {
12     if (a[5][3] > a[2][1]) {
13         return a[0][0];
14     } else if (a[1][2] > b[0][1]) {
15         return b[1][1];
16     } else {
17         return b[4][4];
18     }
19 }
20
21 int cal() {
22     int i = 0;
23     int j = 0;
24     int res;
25     struct container ct;
26     while (i < 10) {
27         while (j < 10) {
28             res = res + a[i][j] * b[i][j];
29             j = j + 1;
30         }
31         i = i + 1;
32     }
```

```

33     ct.res = res;
34     ct.cnt = i * j;
35     ct.avg = ct.res / ct.cnt;
36     i = 0;
37     j = 0;
38     while (i < 10) {
39         ct.sum[i] = 0;
40         while (j < 10) {
41             ct.sum[i] = ct.sum[i,] + a[i][j] + b[i][j];
42             j = j + 1;
43         }
44         i = i + 1;
45     }
46 }

```

输出

```

1 Error type B at line 41: Unexpected ', '

```

说明：数组索引中只能出现一个整数。

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 int _a0_b1(int arr[10], float x, int y) {
2     int i = 0, j = -1; int k = 1--2*3;
3     struct ss {
4         float ff;
5         int _i, _j, _k;
6         struct {

```

```

7      int a, b;
8      } s;
9  } s1, s2;
10
11 while (i < 10) {
12     if (a > b) {
13         return 1 + 2 / (a - b) + 3 * 3.14;
14     } else {
15         int tmp = a;
16         a = a + b;
17         b = b - tmp;
18         return a * b * tmp;
19     }
20     ;
21     i = i + 1;
22 }
23 }
24
25 int _F_o__0(struct { int i; int j; } ss, float f, int i) {
26     int a[100];
27     float b[100];
28     ss.i = 0h;
29     ss.j = 100;
30     while (true) {
31         a[ss.i] = b[ss.i];
32         ss.i = ss.i + 1;
33         if (ss.i >= 100) {
34             return a[ss.i - 1,];
35         }
36     }
37     ss.j = ss.i + ss.j;
38 }

```

输出

```
1 Error type B at line 20: Unexpected ';'
2 Error type A at line 28: illegal identifier '0h';
3 Error type B at line 34: Unexpected ','
```

说明：20 行多了一个“;”；28 行的“0h”不是合法的整数常量，这里也可以报成 B 类错误；34 行的数组索引中多了一个“,”。

2.2 B-2

输入

```
1 struct Info {
2     int amount;
3     int level;
4     int fee;
5 };
6
7 struct Info calculate(struct Info input, int a, int l) {
8     struct Info res;
9     if (a > 0) {
10         res.amount = a;
11     } else {
12         res.amount = input.amount;
13     }
14     if (l > 0) {
15         res.level = l;
16     } else {
17         res.level = input.level;
18     }
19
20     if (res.level == 0) {
21         res.fee = ----430.43;
22     } else if (res.level == 1) {
```

```

23     res.fee = (res.amount - 43.33) * res.level + 43.33 * (res.level +
        1);
24 } else if (res.level == 2) {
25     res.fee = res.amount / 3 + res.amount * (res.level + 100);
26 } else
27     res.fee = 100 * ((100 / res.amount) - res.level);
28
29 return res + res * a-;
30 }
31
32 int main() {
33     int m, n, k = 1 + 2 * 4;
34     struct ss {
35         struct { };
36         int i, j = 0;
37     } s;
38 }

```

输出

```

1 Error type B at line 22: Unexpected '=' after '='
2 Error type B at line 29: Expect expression after '-'
3 Error type B at line 35: Expect identifier after '}'

```

说明：22 行多了一个空格；29 行“-”后面少了一个表达式；35 行少了变量名，结构体中可以再定义结构体类型的变量，而不能只声明结构体类型。

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  int main() {
2      bin2dec(1001001);
3      dec2bin(1234567);
4  }
5
6  int rem(int a, int b) {
7      return a - a / b * b;
8  }
9
10 int bin2dec(int n) {
11     int dec = 0, i = 0, rem;
12     while (n != 0) {
13         rem = rem(n, 10);
14         n = n / 10;
15         dec = dec + rem * pow(2, i);
16         i = i + 1;
17     }
18     return dec;
19 }
20
21 int dec2bin(int n) {
22     if (n <= 1)
23         return n;
24     else {
25         dec2bin(n / 2);
26         write(rem(n, 2));
27     }
28 }
29
30 int cal() {
31     int z_x = 0;
32     while (1 * 2 + (2 - 3 * 4 )) {

```

```

33     if (a > b)
34         dec2bin(a);
35     else {
36         int z_x = x / (y * y - z);
37         bin2dec(b);
38     }
39 }
40 }

```

输出

```

1 Program (1)
2 ExtDefList (1)
3   ExtDef (1)
4     Specifier (1)
5       TYPE: int
6     FunDec (1)
7       ID: main
8       LP
9       RP
10    CompSt (1)
11      LC
12      StmtList (2)
13        Stmt (2)
14          Exp (2)
15            ID: bin2dec
16            LP
17            Args (2)
18              Exp (2)
19                INT: 1001001
20              RP
21            SEMI
22          StmtList (3)
23            Stmt (3)

```

```

24         Exp (3)
25         ID: dec2bin
26         LP
27         Args (3)
28         Exp (3)
29         INT: 1234567
30         RP
31     SEMI
32 RC
33 ExtDefList (6)
34     ExtDef (6)
35         Specifier (6)
36         TYPE: int
37     FunDec (6)
38         ID: rem
39         LP
40         VarList (6)
41             ParamDec (6)
42                 Specifier (6)
43                 TYPE: int
44             VarDec (6)
45                 ID: a
46         COMMA
47         VarList (6)
48             ParamDec (6)
49                 Specifier (6)
50                 TYPE: int
51             VarDec (6)
52                 ID: b
53         RP
54     CompSt (6)
55         LC

```



```

56      StmtList (7)
57      Stmt (7)
58      RETURN
59      Exp (7)
60      Exp (7)
61      ID: a
62      MINUS
63      Exp (7)
64      Exp (7)
65      Exp (7)
66      ID: a
67      DIV
68      Exp (7)
69      ID: b
70      STAR
71      Exp (7)
72      ID: b
73      SEMI
74      RC
75  ExtDefList (10)
76      ExtDef (10)
77      Specifier (10)
78      TYPE: int
79      FunDec (10)
80      ID: bin2dec
81      LP
82      VarList (10)
83      ParamDec (10)
84      Specifier (10)
85      TYPE: int
86      VarDec (10)
87      ID: n

```

```

88         RP
89     CompSt (10)
90         LC
91     DefList (11)
92         Def (11)
93             Specifier (11)
94                 TYPE: int
95             DecList (11)
96                 Dec (11)
97                     VarDec (11)
98                         ID: dec
99                         ASSIGNOP
100                        Exp (11)
101                            INT: 0
102                        COMMA
103                        DecList (11)
104                            Dec (11)
105                                VarDec (11)
106                                    ID: i
107                                    ASSIGNOP
108                                    Exp (11)
109                                        INT: 0
110                                    COMMA
111                                    DecList (11)
112                                        Dec (11)
113                                            VarDec (11)
114                                                ID: rem
115        SEMI
116    StmtList (12)
117        Stmt (12)
118            WHILE
119                LP

```

120	Exp (12)
121	Exp (12)
122	ID: n
123	RELOP
124	Exp (12)
125	INT: 0
126	RP
127	Stmt (12)
128	CompSt (12)
129	LC
130	StmtList (13)
131	Stmt (13)
132	Exp (13)
133	Exp (13)
134	ID: rem
135	ASSIGNOP
136	Exp (13)
137	ID: rem
138	LP
139	Args (13)
140	Exp (13)
141	ID: n
142	COMMA
143	Args (13)
144	Exp (13)
145	INT: 10
146	RP
147	SEMI
148	StmtList (14)
149	Stmt (14)
150	Exp (14)
151	Exp (14)

152	ID: n
153	ASSIGNOP
154	Exp (14)
155	Exp (14)
156	ID: n
157	DIV
158	Exp (14)
159	INT: 10
160	SEMI
161	StmtList (15)
162	Stmt (15)
163	Exp (15)
164	Exp (15)
165	ID: dec
166	ASSIGNOP
167	Exp (15)
168	Exp (15)
169	ID: dec
170	PLUS
171	Exp (15)
172	Exp (15)
173	ID: rem
174	STAR
175	Exp (15)
176	ID: pow
177	LP
178	Args (15)
179	Exp (15)
180	INT: 2
181	COMMA
182	Args (15)
183	Exp (15)

```

184                                     ID: i
185                                     RP
186                                     SEMI
187                               StmtList (16)
188                               Stmt (16)
189                               Exp (16)
190                               Exp (16)
191                               ID: i
192                               ASSIGNOP
193                               Exp (16)
194                               Exp (16)
195                               ID: i
196                               PLUS
197                               Exp (16)
198                               INT: 1
199                               SEMI
200                               RC
201                               StmtList (18)
202                               Stmt (18)
203                               RETURN
204                               Exp (18)
205                               ID: dec
206                               SEMI
207                               RC
208                               ExtDefList (21)
209                               ExtDef (21)
210                               Specifier (21)
211                               TYPE: int
212                               FunDec (21)
213                               ID: dec2bin
214                               LP
215                               VarList (21)

```

```

216         ParamDec (21)
217             Specifier (21)
218                 TYPE: int
219             VarDec (21)
220                 ID: n
221         RP
222     CompSt (21)
223         LC
224         StmtList (22)
225             Stmt (22)
226                 IF
227                     LP
228                     Exp (22)
229                         Exp (22)
230                             ID: n
231                             RELOP
232                             Exp (22)
233                                 INT: 1
234                     RP
235                     Stmt (23)
236                         RETURN
237                         Exp (23)
238                             ID: n
239                         SEMI
240                     ELSE
241                     Stmt (24)
242                         CompSt (24)
243                             LC
244                             StmtList (25)
245                                 Stmt (25)
246                                     Exp (25)
247                                         ID: dec2bin

```

248	LP
249	Args (25)
250	Exp (25)
251	Exp (25)
252	ID: n
253	DIV
254	Exp (25)
255	INT: 2
256	RP
257	SEMI
258	StmtList (26)
259	Stmt (26)
260	Exp (26)
261	ID: write
262	LP
263	Args (26)
264	Exp (26)
265	ID: rem
266	LP
267	Args (26)
268	Exp (26)
269	ID: n
270	COMMA
271	Args (26)
272	Exp (26)
273	INT: 2
274	RP
275	RP
276	SEMI
277	RC
278	RC
279	ExtDefList (30)

```

280     ExtDef (30)
281         Specifier (30)
282             TYPE: int
283     FunDec (30)
284         ID: cal
285         LP
286         RP
287     CompSt (30)
288         LC
289     DefList (31)
290         Def (31)
291             Specifier (31)
292                 TYPE: int
293             DecList (31)
294                 Dec (31)
295                     VarDec (31)
296                         ID: z_x
297                         ASSIGNOP
298                         Exp (31)
299                         INT: 0
300             SEMI
301     StmtList (32)
302         Stmt (32)
303             WHILE
304                 LP
305                 Exp (32)
306                 Exp (32)
307                 Exp (32)
308                 INT: 1
309                 STAR
310                 Exp (32)
311                 INT: 2

```


312	PLUS
313	Exp (32)
314	LP
315	Exp (32)
316	Exp (32)
317	INT: 2
318	MINUS
319	Exp (32)
320	Exp (32)
321	INT: 3
322	STAR
323	Exp (32)
324	INT: 4
325	RP
326	RP
327	Stmt (32)
328	CompSt (32)
329	LC
330	StmtList (33)
331	Stmt (33)
332	IF
333	LP
334	Exp (33)
335	Exp (33)
336	ID: a
337	RELOP
338	Exp (33)
339	ID: b
340	RP
341	Stmt (34)
342	Exp (34)
343	ID: dec2bin

344	LP
345	Args (34)
346	Exp (34)
347	ID: a
348	RP
349	SEMI
350	ELSE
351	Stmt (35)
352	CompSt (35)
353	LC
354	DefList (36)
355	Def (36)
356	Specifier (36)
357	TYPE: int
358	DecList (36)
359	Dec (36)
360	VarDec (36)
361	ID: z_x
362	ASSIGNOP
363	Exp (36)
364	Exp (36)
365	ID: x
366	DIV
367	Exp (36)
368	LP
369	Exp (36)
370	Exp (36)
371	Exp (36)
372	ID: y
373	STAR
374	Exp (36)
375	ID: y

```

376                                     MINUS
377                                     Exp (36)
378                                     ID: z
379                                     RP
380                                     SEMI
381                               StmtList (37)
382                               Stmt (37)
383                               Exp (37)
384                               ID: bin2dec
385                               LP
386                               Args (37)
387                               Exp (37)
388                               ID: b
389                               RP
390                               SEMI
391                               RC
392                               RC
393                               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 struct Data {
2     struct a {
3         int x, y, z = 1;
4     } s1;
5     struct b s2;
6     float f_1, f__2 = 2.2, _f3;
7     int a[10][11];
8 } xs;
9
10 struct Data comp() {

```

```

11  xs.s1.x = xs.s1.y * xs.s1.z + xs.s1.z;
12  xs.s2 = xs.s2;
13  xs.f_1 = xs.f__2;
14  if (xs.f_1 > 0.01) {
15      xs._f3 = xs._f3 * xs.f_1 / xs.f__2;
16      xs.f__2 = xs.f__2 + 1;
17  } else
18      xs.f_1 = xs.f__2;
19
20  while (xs.s1.x > xs.s1.y) {
21      int i = 0;
22      while (i > xs.s1.z) {
23          xs._f3 = 0.1 * xs._f3;
24          xs.s1.z = xs.s1.z * 2;
25          if (xs.a[i][i]) {
26              return xs;
27          }
28          xs.s1.x = xs.s1.y / i;
29      }
30  }
31 }
32
33 int main() {
34     int _i = 0, _j = 0;
35     comp();
36     while (i < 10) {
37         while (j < 10) {
38             xs.a[i][j] = i * j + xs.s1.x;
39         }
40         xs.s1.y = xs.a[i][j];
41     }
42     return xs.a[2][3];

```

43 }

输出

```
1 Program (1)
2 ExtDefList (1)
3   ExtDef (1)
4     Specifier (1)
5       StructSpecifier (1)
6         STRUCT
7         OptTag (1)
8         ID: Data
9         LC
10      DefList (2)
11        Def (2)
12          Specifier (2)
13            StructSpecifier (2)
14              STRUCT
15              OptTag (2)
16              ID: a
17              LC
18            DefList (3)
19              Def (3)
20                Specifier (3)
21                  TYPE: int
22                DecList (3)
23                  Dec (3)
24                    VarDec (3)
25                      ID: x
26                    COMMA
27                  DecList (3)
28                    Dec (3)
29                      VarDec (3)
30                        ID: y
```

```

31          COMMA
32          DecList (3)
33          Dec (3)
34          VarDec (3)
35          ID: z
36          ASSIGNOP
37          Exp (3)
38          INT: 1
39          SEMI
40          RC
41      DecList (4)
42          Dec (4)
43          VarDec (4)
44          ID: s1
45      SEMI
46  DefList (5)
47      Def (5)
48          Specifier (5)
49          StructSpecifier (5)
50          STRUCT
51          Tag (5)
52          ID: b
53          DecList (5)
54          Dec (5)
55          VarDec (5)
56          ID: s2
57      SEMI
58  DefList (6)
59      Def (6)
60          Specifier (6)
61          TYPE: float
62          DecList (6)

```

```

63         Dec (6)
64         VarDec (6)
65             ID: f_1
66         COMMA
67         DecList (6)
68             Dec (6)
69                 VarDec (6)
70                     ID: f__2
71                 ASSIGNOP
72                 Exp (6)
73                     FLOAT: 2.200000
74                 COMMA
75                 DecList (6)
76                     Dec (6)
77                         VarDec (6)
78                             ID: _f3
79         SEMI
80     DefList (7)
81         Def (7)
82             Specifier (7)
83                 TYPE: int
84             DecList (7)
85                 Dec (7)
86                     VarDec (7)
87                         VarDec (7)
88                             VarDec (7)
89                                 ID: a
90                                 LB
91                                 INT: 10
92                                 RB
93                                 LB
94                                 INT: 11

```

```

95                                     RB
96                                 SEMI
97                             RC
98         ExtDecList (8)
99             VarDec (8)
100                 ID: xs
101         SEMI
102     ExtDefList (10)
103         ExtDef (10)
104             Specifier (10)
105                 StructSpecifier (10)
106                     STRUCT
107                         Tag (10)
108                             ID: Data
109         FunDec (10)
110             ID: comp
111             LP
112             RP
113         CompSt (10)
114             LC
115             StmtList (11)
116                 Stmt (11)
117                     Exp (11)
118                         Exp (11)
119                             Exp (11)
120                                 Exp (11)
121                                     ID: xs
122                                         DOT
123                                             ID: s1
124                                                 DOT
125                                                     ID: x
126                 ASSIGNOP

```


127	Exp (11)
128	Exp (11)
129	Exp (11)
130	Exp (11)
131	Exp (11)
132	ID: xs
133	DOT
134	ID: s1
135	DOT
136	ID: y
137	STAR
138	Exp (11)
139	Exp (11)
140	Exp (11)
141	ID: xs
142	DOT
143	ID: s1
144	DOT
145	ID: z
146	PLUS
147	Exp (11)
148	Exp (11)
149	Exp (11)
150	ID: xs
151	DOT
152	ID: s1
153	DOT
154	ID: z
155	SEMI
156	StmtList (12)
157	Stmt (12)
158	Exp (12)

```

159         Exp (12)
160         Exp (12)
161         ID: xs
162         DOT
163         ID: s2
164         ASSIGNOP
165         Exp (12)
166         Exp (12)
167         ID: xs
168         DOT
169         ID: s2
170         SEMI
171     StmtList (13)
172     Stmt (13)
173     Exp (13)
174     Exp (13)
175     Exp (13)
176     ID: xs
177     DOT
178     ID: f_1
179     ASSIGNOP
180     Exp (13)
181     Exp (13)
182     ID: xs
183     DOT
184     ID: f__2
185     SEMI
186     StmtList (14)
187     Stmt (14)
188     IF
189     LP
190     Exp (14)

```

191	Exp (14)
192	Exp (14)
193	ID: xs
194	DOT
195	ID: f_1
196	RELOP
197	Exp (14)
198	FLOAT: 0.010000
199	RP
200	Stmt (14)
201	CompSt (14)
202	LC
203	StmtList (15)
204	Stmt (15)
205	Exp (15)
206	Exp (15)
207	Exp (15)
208	ID: xs
209	DOT
210	ID: _f3
211	ASSIGNOP
212	Exp (15)
213	Exp (15)
214	Exp (15)
215	Exp (15)
216	ID: xs
217	DOT
218	ID: _f3
219	STAR
220	Exp (15)
221	Exp (15)
222	ID: xs

```

223         DOT
224         ID: f_1
225     DIV
226     Exp (15)
227         Exp (15)
228         ID: xs
229     DOT
230     ID: f__2
231 SEMI
232 StmtList (16)
233     Stmt (16)
234     Exp (16)
235     Exp (16)
236     Exp (16)
237     ID: xs
238     DOT
239     ID: f__2
240 ASSIGNOP
241 Exp (16)
242     Exp (16)
243     Exp (16)
244     ID: xs
245     DOT
246     ID: f__2
247 PLUS
248     Exp (16)
249     INT: 1
250 SEMI
251 RC
252 ELSE
253 Stmt (18)
254     Exp (18)

```

255	Exp (18)
256	Exp (18)
257	ID: xs
258	DOT
259	ID: f_1
260	ASSIGNOP
261	Exp (18)
262	Exp (18)
263	ID: xs
264	DOT
265	ID: f__2
266	SEMI
267	StmtList (20)
268	Stmt (20)
269	WHILE
270	LP
271	Exp (20)
272	Exp (20)
273	Exp (20)
274	Exp (20)
275	ID: xs
276	DOT
277	ID: s1
278	DOT
279	ID: x
280	RELOP
281	Exp (20)
282	Exp (20)
283	Exp (20)
284	ID: xs
285	DOT
286	ID: s1

```

287         DOT
288         ID: y
289     RP
290     Stmt (20)
291         CompSt (20)
292             LC
293             DefList (21)
294                 Def (21)
295                     Specifier (21)
296                         TYPE: int
297                     DeclList (21)
298                         Dec (21)
299                             VarDec (21)
300                                 ID: i
301                                 ASSIGNOP
302                                 Exp (21)
303                                 INT: 0
304                 SEMI
305             StmtList (22)
306                 Stmt (22)
307                     WHILE
308                         LP
309                         Exp (22)
310                             Exp (22)
311                                 ID: i
312                                 RELOP
313                             Exp (22)
314                                 Exp (22)
315                                     Exp (22)
316                                         ID: xs
317                                 DOT
318                                     ID: s1

```

319	DOT
320	ID: z
321	RP
322	Stmt (22)
323	CompSt (22)
324	LC
325	StmtList (23)
326	Stmt (23)
327	Exp (23)
328	Exp (23)
329	Exp (23)
330	ID: xs
331	DOT
332	ID: _f3
333	ASSIGNOP
334	Exp (23)
335	Exp (23)
336	FLOAT: 0.100000
337	STAR
338	Exp (23)
339	Exp (23)
340	ID: xs
341	DOT
342	ID: _f3
343	SEMI
344	StmtList (24)
345	Stmt (24)
346	Exp (24)
347	Exp (24)
348	Exp (24)
349	Exp (24)
350	ID: xs

351	DOT
352	ID: s1
353	DOT
354	ID: z
355	ASSIGNOP
356	Exp (24)
357	Exp (24)
358	Exp (24)
359	Exp (24)
360	ID: xs
361	DOT
362	ID: s1
363	DOT
364	ID: z
365	STAR
366	Exp (24)
367	INT: 2
368	SEMI
369	StmtList (25)
370	Stmt (25)
371	IF
372	LP
373	Exp (25)
374	Exp (25)
375	Exp (25)
376	Exp (25)
377	ID: xs
378	DOT
379	ID: a
380	LB
381	Exp (25)
382	ID: i

383	RB
384	LB
385	Exp (25)
386	ID: i
387	RB
388	RP
389	Stmt (25)
390	CompSt (25)
391	LC
392	StmtList (26)
393	Stmt (26)
394	RETURN
395	Exp (26)
396	ID: xs
397	SEMI
398	RC
399	StmtList (28)
400	Stmt (28)
401	Exp (28)
402	Exp (28)
403	Exp (28)
404	Exp (28)
405	ID: xs
406	DOT
407	ID: s1
408	DOT
409	ID: x
410	ASSIGNOP
411	Exp (28)
412	Exp (28)
413	Exp (28)
414	Exp (28)

```

415                                     ID: xs
416                                     DOT
417                                     ID: s1
418                                     DOT
419                                     ID: y
420                                     DIV
421                                     Exp (28)
422                                     ID: i
423                                     SEMI
424                                     RC
425                                     RC
426                                     RC
427 ExtDefList (33)
428   ExtDef (33)
429     Specifier (33)
430       TYPE: int
431     FunDec (33)
432       ID: main
433       LP
434       RP
435     CompSt (33)
436       LC
437     DefList (34)
438       Def (34)
439         Specifier (34)
440           TYPE: int
441         DecList (34)
442           Dec (34)
443             VarDec (34)
444               ID: _i
445             ASSIGNOP
446             Exp (34)

```

```

447             INT: 0
448         COMMA
449         DecList (34)
450         Dec (34)
451         VarDec (34)
452         ID: _j
453         ASSIGNOP
454         Exp (34)
455         INT: 0
456     SEMI
457 StmtList (35)
458     Stmt (35)
459     Exp (35)
460     ID: comp
461     LP
462     RP
463     SEMI
464     StmtList (36)
465     Stmt (36)
466     WHILE
467     LP
468     Exp (36)
469     Exp (36)
470     ID: i
471     RELOP
472     Exp (36)
473     INT: 10
474     RP
475     Stmt (36)
476     CompSt (36)
477     LC
478     StmtList (37)

```

479	Stmt (37)
480	WHILE
481	LP
482	Exp (37)
483	Exp (37)
484	ID: j
485	RELOP
486	Exp (37)
487	INT: 10
488	RP
489	Stmt (37)
490	CompSt (37)
491	LC
492	StmtList (38)
493	Stmt (38)
494	Exp (38)
495	Exp (38)
496	Exp (38)
497	Exp (38)
498	Exp (38)
499	ID: xs
500	DOT
501	ID: a
502	LB
503	Exp (38)
504	ID: i
505	RB
506	LB
507	Exp (38)
508	ID: j
509	RB
510	ASSIGNOP

511	Exp (38)
512	Exp (38)
513	Exp (38)
514	ID: i
515	STAR
516	Exp (38)
517	ID: j
518	PLUS
519	Exp (38)
520	Exp (38)
521	Exp (38)
522	ID: xs
523	DOT
524	ID: s1
525	DOT
526	ID: x
527	SEMI
528	RC
529	StmtList (40)
530	Stmt (40)
531	Exp (40)
532	Exp (40)
533	Exp (40)
534	Exp (40)
535	ID: xs
536	DOT
537	ID: s1
538	DOT
539	ID: y
540	ASSIGNOP
541	Exp (40)
542	Exp (40)

543	Exp (40)
544	Exp (40)
545	ID: xs
546	DOT
547	ID: a
548	LB
549	Exp (40)
550	ID: i
551	RB
552	LB
553	Exp (40)
554	ID: j
555	RB
556	SEMI
557	RC
558	StmtList (42)
559	Stmt (42)
560	RETURN
561	Exp (42)
562	Exp (42)
563	Exp (42)
564	Exp (42)
565	ID: xs
566	DOT
567	ID: a
568	LB
569	Exp (42)
570	INT: 2
571	RB
572	LB
573	Exp (42)
574	INT: 3

575
576
577

RB
SEMI
RC

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 main() {  
2     int i = 122 * 0x34;  
3     int j = i * i + 182 / i;  
4     int k = 0323 + j;  
5 }
```

输出

```
1 Program (1)  
2 ExtDefList (1)  
3   ExtDef (1)  
4     Specifier (1)  
5       TYPE: int  
6     FunDec (1)  
7       ID: main  
8       LP  
9       RP  
10      CompSt (1)  
11        LC  
12        DefList (2)  
13          Def (2)
```

```

14     Specifier (2)
15         TYPE: int
16     DecList (2)
17         Dec (2)
18             VarDec (2)
19                 ID: i
20             ASSIGNOP
21             Exp (2)
22                 Exp (2)
23                     INT: 122
24                 STAR
25                 Exp (2)
26                     INT: 52
27     SEMI
28     DefList (3)
29         Def (3)
30             Specifier (3)
31                 TYPE: int
32             DecList (3)
33                 Dec (3)
34                     VarDec (3)
35                         ID: j
36                     ASSIGNOP
37                     Exp (3)
38                         Exp (3)
39                             Exp (3)
40                                 ID: i
41                             STAR
42                             Exp (3)
43                                 ID: i
44                             PLUS
45                             Exp (3)

```



```

46             Exp (3)
47                 INT: 182
48             DIV
49             Exp (3)
50                 ID: i
51         SEMI
52     DefList (4)
53         Def (4)
54             Specifier (4)
55                 TYPE: int
56             Declist (4)
57                 Dec (4)
58                     VarDec (4)
59                         ID: k
60                     ASSIGNOP
61                     Exp (4)
62                         Exp (4)
63                             INT: 211
64                         PLUS
65                         Exp (4)
66                             ID: j
67         SEMI
68     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；其他分组的同学只要提示相应的错误（不输出语法树即）可。

4.2 D-2

输入

```

1 int main() {
2     float f_1 = 0.232342;
3     float f_2 = .23e-10;
4     float f_3 = f_1 * f_2 - f_1;

```

```
5   f_1 = f_1 * 32.E-12;
6 }
```

输出

```
1 Program (1)
2 ExtDefList (1)
3   ExtDef (1)
4     Specifier (1)
5       TYPE: int
6     FunDec (1)
7       ID: main
8       LP
9       RP
10    CompSt (1)
11      LC
12      DefList (2)
13        Def (2)
14          Specifier (2)
15            TYPE: float
16          DecList (2)
17            Dec (2)
18              VarDec (2)
19                ID: f_1
20                ASSIGNOP
21                Exp (2)
22                  FLOAT: 0.232342
23            SEMI
24          DefList (3)
25            Def (3)
26              Specifier (3)
27                TYPE: float
28              DecList (3)
29                Dec (3)
```

```

30         VarDec (3)
31             ID: f_2
32         ASSIGNOP
33         Exp (3)
34             FLOAT: 0.000000
35     SEMI
36 DefList (4)
37     Def (4)
38         Specifier (4)
39             TYPE: float
40         DecList (4)
41             Dec (4)
42                 VarDec (4)
43                     ID: f_3
44                 ASSIGNOP
45                 Exp (4)
46                     Exp (4)
47                         Exp (4)
48                             ID: f_1
49                         STAR
50                         Exp (4)
51                             ID: f_2
52                         MINUS
53                         Exp (4)
54                             ID: f_1
55     SEMI
56 StmtList (5)
57     Stmt (5)
58         Exp (5)
59             Exp (5)
60                 ID: f_1
61             ASSIGNOP

```

```

62         Exp (5)
63         Exp (5)
64         ID: f_1
65         STAR
66         Exp (5)
67         FLOAT: 0.000000
68     SEMI
69 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（不输出语法树）即可。

4.3 D-3

输入

```

1  /*
2  ** Traverse a Lua closure, marking its prototype and its upvalues.
3  ** (Both can be NULL while closure is being created.)
4  */
5  int traverseLclosure (struct global_State g, struct LClosure cl) {
6      int i = 0;
7      markobjectN(g, cl.p); /* mark its prototype */
8      while (i < cl.nupvalues) { /* visit its upvalues */
9          struct UpVal uv = cl.upvals[i];
10         markobjectN(g, uv); /* mark upvalue */
11         i = i + 1;
12     }
13     return 1 + cl.nupvalues;
14 }

```

输出

```

1 Program (5)
2 ExtDefList (5)
3   ExtDef (5)

```

```

4     Specifier (5)
5         TYPE: int
6     FunDec (5)
7         ID: traverseLclosure
8         LP
9         VarList (5)
10            ParamDec (5)
11                Specifier (5)
12                    StructSpecifier (5)
13                        STRUCT
14                            Tag (5)
15                                ID: global_State
16            VarDec (5)
17                ID: g
18        COMMA
19        VarList (5)
20            ParamDec (5)
21                Specifier (5)
22                    StructSpecifier (5)
23                        STRUCT
24                            Tag (5)
25                                ID: LClosure
26            VarDec (5)
27                ID: cl
28        RP
29    CompSt (5)
30        LC
31        DefList (6)
32            Def (6)
33                Specifier (6)
34                    TYPE: int
35                DeclList (6)

```

```

36         Dec (6)
37         VarDec (6)
38             ID: i
39         ASSIGNOP
40         Exp (6)
41             INT: 0
42     SEMI
43 StmtList (7)
44     Stmt (7)
45     Exp (7)
46         ID: markobjectN
47     LP
48     Args (7)
49         Exp (7)
50             ID: g
51     COMMA
52     Args (7)
53         Exp (7)
54             Exp (7)
55                 ID: cl
56     DOT
57         ID: p
58     RP
59     SEMI
60 StmtList (8)
61     Stmt (8)
62     WHILE
63     LP
64     Exp (8)
65         Exp (8)
66             ID: i
67     RELOP

```

```

68         Exp (8)
69         Exp (8)
70         ID: cl
71         DOT
72         ID: nupvalues
73     RP
74     Stmt (8)
75     CompSt (8)
76     LC
77     DefList (9)
78     Def (9)
79     Specifier (9)
80     StructSpecifier (9)
81     STRUCT
82     Tag (9)
83     ID: UpVal
84     DecList (9)
85     Dec (9)
86     VarDec (9)
87     ID: uv
88     ASSIGNOP
89     Exp (9)
90     Exp (9)
91     Exp (9)
92     ID: cl
93     DOT
94     ID: upvals
95     LB
96     Exp (9)
97     ID: i
98     RB
99     SEMI

```

```

100      StmtList (10)
101      Stmt (10)
102      Exp (10)
103      ID: markobjectN
104      LP
105      Args (10)
106      Exp (10)
107      ID: g
108      COMMA
109      Args (10)
110      Exp (10)
111      ID: uv
112      RP
113      SEMI
114      StmtList (11)
115      Stmt (11)
116      Exp (11)
117      Exp (11)
118      ID: i
119      ASSIGNOP
120      Exp (11)
121      Exp (11)
122      ID: i
123      PLUS
124      Exp (11)
125      INT: 1
126      SEMI
127      RC
128      StmtList (13)
129      Stmt (13)
130      RETURN
131      Exp (13)

```



```

132         Exp (13)
133             INT: 1
134         PLUS
135         Exp (13)
136             Exp (13)
137                 ID: c1
138             DOT
139                 ID: nupvalues
140         SEMI
141     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（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

```

1 int main() {
2     int b10 = 0655378;
3     int b15 = 16777216 + b10 / -0xfd;
4     int b18 = -0x7fffg;
5     int b20 = -0x1ffffd;
6     int b21 = -0312 + 0xdead;
7     int b23 = 065536;
8     int b24 = -0xffffffff;
9 }

```

输出

```

1 Error type A at line 2: Illegal octal number '0655378'

```

```
2 Error type A at line 4: Illegal hexadecimal number '0x7fffg'
```

说明：仅 1.1 分组的同学需要测试这个用例，这两处错误都可以识别成错误 B。

5.2 E1-2

这组测试用例针对 1.1 分组的同学。

输入

```
1 int ENCODER(struct Obj johab)
2 {
3     while (inpos < inlen) {
4         if (c < 0x80) {
5             WRITEBYTE1(c);
6             NEXT(1, 1);
7         }
8
9         if (c > 0xFFFF)
10            return 1;
11
12        REQUIRE_OUTBUF(2);
13
14        if (c >= 0x3131 && c <= 0x3163)
15            code = u2johabjamo[c - 0x3131];
16        else if (TRYMAP_ENC(cp949, code, c)) {
17            int c1, c2, t2;
18            int t1;
19
20            assert((code && 0x8000) == 0);
21            c1 = code + 0777;
22            c2 = code - 0xff;
23            if (((c1 >= 0x012221 && c1 <= 0213232) ||
24                (c1 >= 0xac4a && c1 <= 0xab7d)) &&
25                (c2 >= 0xef21 && c2 <= 0xcd7e)) {
```

```

26         t1 = (c1 - 0x4323fddd + (c1 - 0x2132fefd + 0x1b2abdc d) + (c1
           - 0x21 + 0x197));
27         t2 = ((t1 + 1) + 0x5e - 0) + (c2 - 0x21);
28         OUTBYTE1(t1 + 1);
29         OUTBYTE2(t2 - 0x4e - t2 + 0x31 + t2 + 0x43);
30         NEXT(1, 2);
31         continue;
32     }
33     else
34         return 1;
35 }
36 else
37     return 1;
38 }
39 return 0;
40 }

```

输出

```

1 // 语法树过大，不在这里展示，请关注随本文档一同发放的测试用例文件。

```

说明：仅 1.1 分组的同学需要测试这个用例，并输出语法树。

5.3 E2-1

这组测试用例针对 1.2 分组的同学。

输入

```

1 float main() {
2     float ik_j = 0.001e;
3     float _j = ik_j + 0123.2323E+32;
4     struct {
5         float f1;
6         float f2;
7     } s;
8     float mm = s.f1 * _j - 0.2132.0e232;

```

```
9 }
```

输出

```
1 Error type A at line 2: Invalid floating point number
2 Error type A at line 8: Invalid floating point number
```

说明：仅 1.2 分组的同学需要测试这个用例，这里的两个错误都可以识别成 B 类错误。

5.4 E2-2

这组测试用例针对 1.2 分组的同学。

输入

```
1 float fo() {
2     float f1 = e1.e1;
3     float f2 = -213.e1;
4     f1 = f1 / 0.1e+1;
5     f2 = f1 * 0213.320e+2 - (.1e1 - 1.e1);
6 }
```

输出

```
1 Program (1)
2 ExtDefList (1)
3     ExtDef (1)
4         Specifier (1)
5             TYPE: float
6         FunDec (1)
7             ID: fo
8             LP
9             RP
10        CompSt (1)
11            LC
12            DefList (2)
13                Def (2)
14                    Specifier (2)
```

```

15         TYPE: float
16     DecList (2)
17         Dec (2)
18             VarDec (2)
19                 ID: f1
20             ASSIGNOP
21             Exp (2)
22                 Exp (2)
23                     ID: e1
24                 DOT
25                     ID: e1
26         SEMI
27     DefList (3)
28         Def (3)
29             Specifier (3)
30                 TYPE: float
31             DecList (3)
32                 Dec (3)
33                     VarDec (3)
34                         ID: f2
35                     ASSIGNOP
36                     Exp (3)
37                         MINUS
38                         Exp (3)
39                             FLOAT: 2130.000000
40         SEMI
41     StmtList (4)
42         Stmt (4)
43             Exp (4)
44                 Exp (4)
45                     ID: f1
46             ASSIGNOP

```

47	Exp (4)
48	Exp (4)
49	ID: f1
50	DIV
51	Exp (4)
52	FLOAT: 1.000000
53	SEMI
54	StmtList (5)
55	Stmt (5)
56	Exp (5)
57	Exp (5)
58	ID: f2
59	ASSIGNOP
60	Exp (5)
61	Exp (5)
62	Exp (5)
63	ID: f1
64	STAR
65	Exp (5)
66	FLOAT: 21332.000000
67	MINUS
68	Exp (5)
69	LP
70	Exp (5)
71	Exp (5)
72	FLOAT: 1.000000
73	MINUS
74	Exp (5)
75	FLOAT: 10.000000
76	RP
77	SEMI
78	RC

说明：仅 1.2 分组的同学需要测试这个用例，并输出语法树。

5.5 E3-1

这组测试用例针对 1.3 分组的同学。

输入

```
1  /*
2  ** Does a young collection. First, mark 'OLD1' objects. Then does the
3  ** atomic step. Then, sweep all lists and advance pointers. Finally,
4  ** finish the collection.
5  */
6  int youngcollection (struct lua_State L, struct global_State g) {
7      struct GCObject psurvival; /* to point to first non-dead survival
8      object */
9      struct GCObject dummy; /* dummy out parameter to 'sweepgen' */
10     lua_assert(g.gcstate == GCSpropagate);
11     if (g.firstold1) { /* are there regular OLD1 objects? */
12         markold(g, g.firstold1, g.reallyold); /* mark them */
13         g.firstold1 = NULL; /* no more OLD1 objects (for now) */
14     }
15     markold(g, g.finobj, g.finobjrold);
16     markold(g, g.tobefnz, NULL);
17     atomic(L);
18
19     /* sweep nursery and get a pointer to its last live element */
20     g.gcstate = GCSswpallgc;
21     psurvival = sweepgen(L, g, g.allgc, g.survival, g.firstold1);
22     /* sweep 'survival' */
23     sweepgen(L, g, psurvival, g.old1, g.firstold1);
24     g.reallyold
25     /* kdfjaueiowu klj;;;???!kljfkldsaj f--23=
26     ==2q-403w-40-32 * / jlkfdj //i/
27     kllfdsa'''''';!!
```

```

27     */ = g.old1;
28     g.old1 = psurvival; /* 'survival' survivals are old now */
29     g.survival = g.allgc; /* all news are survivals */
30
31     /* repeat for 'finobj' lists */
32     dummy = NULL /* no 'firstold1' optimization for 'finobj' lists */
33     psurvival = sweepgen(L, g, g.finobj, g.finobjsur, dummy);
34     /* sweep 'survival' */
35     sweepgen(L, g, psurvival, g.finobjold1, dummy);
36     g.finobjrold = g.finobjold1;
37     g.finobjold1 = psurvival; /* 'survival' survivals are old now */
38     g.finobjsur = g.finobj; /* all news are survivals */
39
40     sweepgen(L, g, g.tobefnz, NULL, dummy);
41     finishgencycle(L, g);
42 }

```

输出

```

1 Error type B at line 32: Expect ';'

```

说明：仅 1.3 分组的同学需要测试这个用例。第 32 行的这个错误也可以报在第 33 行。

5.6 E3-2

这组测试用例针对 1.3 分组的同学。

输入

```

1 /*
2 ** Compare two strings 'ls' x 'rs', returning an integer less-equal-
3 ** -greater than zero if 'ls' is less-equal-greater than 'rs'.
4 ** The code is a little tricky because it allows '\0' in the strings
5 ** and it uses 'strcoll' (to respect locales) for each segments
6 ** of the strings.
7 */
8 int l_strcmp (struct TString ls, struct TString rs) {

```



```

9  int l = getstr(ls);
10 int ll = tsslen(ls);
11 int r = getstr(rs);
12 int lr = tsslen(rs);
13 while (1) { /* for each segment */
14     int temp = strcoll(l, r);
15     if (temp != 0) /* not equal? */
16         return temp; /* done */
17     else { /* strings are equal up to a '\0' */
18         int len = strlen(l); /* index of first '\0' in both strings */
19         if (len == lr) /* 'rs' is finished? */
20             return (len == ll) - 0 + 1; /* check 'ls' */
21         else if (len == ll) /* 'ls' is finished? */
22             return -1; /* 'ls' is less than 'rs' ('rs' is not finished)
23                 */
24         /* both strings longer than 'len'; go on comparing after the
25            '\0' */
26         len = len + 1;
27         l = l + len; ll = ll - len; r = r + len; lr = lr - len;
28     }
29 }
30 /*
31 *****
32 *          Prolog          *
33 *****
34 */
35 int /* some stuff */ foo() { // other things
36     int a /* /* kljkfldjkdafkljaslkfjda !!!!
37         jfdklsajkjll fdljsakfldsaj fdfdkljda sa fda?????
38         djakfljdaskl //....///

```

```

39  */ = 323; // jlkfdjs afd sa
40  int // fldsajflkdjsa fdlsjafk
41  b = /*****/ ----1;
42  }

```

输出

```

1  Program (8)
2  ExtDefList (8)
3    ExtDef (8)
4      Specifier (8)
5        TYPE: int
6      FunDec (8)
7        ID: l_strcmp
8        LP
9        VarList (8)
10         ParamDec (8)
11           Specifier (8)
12             StructSpecifier (8)
13               STRUCT
14               Tag (8)
15                 ID: TString
16         VarDec (8)
17           ID: ls
18         COMMA
19         VarList (8)
20           ParamDec (8)
21             Specifier (8)
22               StructSpecifier (8)
23                 STRUCT
24                 Tag (8)
25                   ID: TString
26           VarDec (8)
27             ID: rs

```

```

28     RP
29 CompSt (8)
30     LC
31     DefList (9)
32         Def (9)
33             Specifier (9)
34                 TYPE: int
35             DecList (9)
36                 Dec (9)
37                     VarDec (9)
38                         ID: l
39                         ASSIGNOP
40                         Exp (9)
41                             ID: getstr
42                             LP
43                             Args (9)
44                                 Exp (9)
45                                     ID: ls
46                                     RP
47                     SEMI
48             DefList (10)
49                 Def (10)
50                     Specifier (10)
51                         TYPE: int
52                     DecList (10)
53                         Dec (10)
54                             VarDec (10)
55                                 ID: l1
56                                 ASSIGNOP
57                                 Exp (10)
58                                     ID: tsslen
59                                     LP

```

```

60         Args (10)
61         Exp (10)
62         ID: ls
63     RP
64 SEMI
65 DefList (11)
66     Def (11)
67         Specifier (11)
68         TYPE: int
69         DecList (11)
70         Dec (11)
71         VarDec (11)
72         ID: r
73         ASSIGNOP
74         Exp (11)
75         ID: getstr
76         LP
77         Args (11)
78         Exp (11)
79         ID: rs
80     RP
81 SEMI
82 DefList (12)
83     Def (12)
84         Specifier (12)
85         TYPE: int
86         DecList (12)
87         Dec (12)
88         VarDec (12)
89         ID: lr
90         ASSIGNOP
91         Exp (12)

```

```

92             ID: tsslen
93             LP
94             Args (12)
95                 Exp (12)
96                 ID: rs
97             RP
98         SEMI
99     StmtList (13)
100     Stmt (13)
101         WHILE
102             LP
103             Exp (13)
104                 INT: 1
105             RP
106             Stmt (13)
107                 CompSt (13)
108                     LC
109                     DefList (14)
110                         Def (14)
111                             Specifier (14)
112                                 TYPE: int
113                             DeclList (14)
114                                 Dec (14)
115                                     VarDec (14)
116                                         ID: temp
117                                     ASSIGNOP
118                                     Exp (14)
119                                         ID: strcoll
120                                     LP
121                                         Args (14)
122                                             Exp (14)
123                                                 ID: l

```

124	COMMA
125	Args (14)
126	Exp (14)
127	ID: r
128	RP
129	SEMI
130	StmtList (15)
131	Stmt (15)
132	IF
133	LP
134	Exp (15)
135	Exp (15)
136	ID: temp
137	RELOP
138	Exp (15)
139	INT: 0
140	RP
141	Stmt (16)
142	RETURN
143	Exp (16)
144	ID: temp
145	SEMI
146	ELSE
147	Stmt (17)
148	CompSt (17)
149	LC
150	DefList (18)
151	Def (18)
152	Specifier (18)
153	TYPE: int
154	DecList (18)
155	Dec (18)

```

156         VarDec (18)
157         ID: len
158         ASSIGNOP
159         Exp (18)
160         ID: strlen
161         LP
162         Args (18)
163         Exp (18)
164         ID: l
165         RP
166     SEMI
167 StmtList (19)
168     Stmt (19)
169     IF
170     LP
171     Exp (19)
172     Exp (19)
173     ID: len
174     RELOP
175     Exp (19)
176     ID: lr
177     RP
178     Stmt (20)
179     RETURN
180     Exp (20)
181     Exp (20)
182     Exp (20)
183     LP
184     Exp (20)
185     Exp (20)
186     ID: len
187     RELOP

```

188	Exp (20)
189	ID: 11
190	RP
191	MINUS
192	Exp (20)
193	INT: 0
194	PLUS
195	Exp (20)
196	INT: 1
197	SEMI
198	ELSE
199	Stmt (21)
200	IF
201	LP
202	Exp (21)
203	Exp (21)
204	ID: len
205	RELOP
206	Exp (21)
207	ID: 11
208	RP
209	Stmt (22)
210	RETURN
211	Exp (22)
212	MINUS
213	Exp (22)
214	INT: 1
215	SEMI
216	StmtList (24)
217	Stmt (24)
218	Exp (24)
219	Exp (24)

220	ID: len
221	ASSIGNOP
222	Exp (24)
223	Exp (24)
224	ID: len
225	PLUS
226	Exp (24)
227	INT: 1
228	SEMI
229	StmtList (25)
230	Stmt (25)
231	Exp (25)
232	Exp (25)
233	ID: 1
234	ASSIGNOP
235	Exp (25)
236	Exp (25)
237	ID: 1
238	PLUS
239	Exp (25)
240	ID: len
241	SEMI
242	StmtList (25)
243	Stmt (25)
244	Exp (25)
245	Exp (25)
246	ID: 11
247	ASSIGNOP
248	Exp (25)
249	Exp (25)
250	ID: 11
251	MINUS

252			Exp (25)
253			ID: len
254			SEMI
255			StmtList (25)
256			Stmt (25)
257			Exp (25)
258			Exp (25)
259			ID: r
260			ASSIGNOP
261			Exp (25)
262			Exp (25)
263			ID: r
264			PLUS
265			Exp (25)
266			ID: len
267			SEMI
268			StmtList (25)
269			Stmt (25)
270			Exp (25)
271			Exp (25)
272			ID: lr
273			ASSIGNOP
274			Exp (25)
275			Exp (25)
276			ID: lr
277			MINUS
278			Exp (25)
279			ID: len
280			SEMI
281		RC	
282		RC	
283	RC		

```

284 ExtDefList (35)
285     ExtDef (35)
286         Specifier (35)
287             TYPE: int
288         FunDec (35)
289             ID: foo
290             LP
291             RP
292     CompSt (35)
293         LC
294     DefList (36)
295         Def (36)
296             Specifier (36)
297                 TYPE: int
298             DecList (36)
299                 Dec (36)
300                     VarDec (36)
301                         ID: a
302                         ASSIGNOP
303                         Exp (39)
304                             INT: 323
305             SEMI
306     DefList (40)
307         Def (40)
308             Specifier (40)
309                 TYPE: int
310             DecList (41)
311                 Dec (41)
312                     VarDec (41)
313                         ID: b
314                         ASSIGNOP
315                         Exp (41)

```

316		MINUS
317		Exp (41)
318		MINUS
319		Exp (41)
320		MINUS
321		Exp (41)
322		MINUS
323		Exp (41)
324		INT: 1
325	SEMI	
326	RC	

说明：仅 1.3 分组的同学需要测试这个用例，需要输出正确的语法树。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与[屈道涵](#)助教联系，注意同时抄送给[许老师](#)。