

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	27
4	D 组测试用例	43
4.1	D-1	43
4.2	D-2	46
4.3	D-3	48
5	E 组测试用例	54
5.1	E1.1	54
5.2	E1.2	58
5.3	E1.3	60
6	结束语	76

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 struct 1Person {  
2     int id;  
3 };  
4  
5 int main() {  
6     return get_id(x);  
7 }
```

输出

```
1 Error type B at Line 1: Illegal identifier "1Persion"
```

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 1 行。这里有一个非法标识符 1_Person。

1.2 A-2

输入

```
1 struct if {  
2     int cond;  
3 };  
4  
5 int abc() {  
6     return 123;  
7 }
```

输出

```
1 Error type B at Line 1: Syntax error
```

说明：关键词 if 不可以作为标识符。

1.3 A-3

输入

```
1 int a, a, a;  
2 float a;  
3  
4 struct a {};  
5  
6 int a = 1;  
7  
8 int a() {}
```

输出

```
1 Error type B at Line 6: Cannot initialize global variable "a"
```

说明：全局变量的声明不可以初始化。

1.4 A-4

输入

```
1 int main() {  
2     struct A a[10];  
3     a[0].a = 13;  
4     a[1].b = d;  
5     int b = a;  
6     return c;  
7 }
```

输出

```
1 Error type B at Line 5: Illegal declaration
```

说明：变量声明只允许出现在语句块头部。

1.5 A-5

输入

```
1 int a;  
2 int;  
3  
4 struct;  
5 struct A;  
6  
7 float b() {  
8     return 1.0;  
9 }
```

输出

```
1 Error type B at Line 4: Unexpected ";"
```

说明：第 4 行缺少结构体名。

1.6 A-6

输入

```
1 int main() {  
2     float a = 1.3, b = 0.000000003, c = .3;  
3     int b = a, c = b, s;  
4     return 0.4;  
5 }
```

输出

```
1 Error type A at Line 2: Illegal FLOAT ".3"
```

说明：“.3”不是合法的浮点数（附录要求浮点数的小数点前后必须出现数字），这里也可以报成语法错误。

1.7 A-7

输入

```

1 int BEIJING;
2 int NEWYORK;
3 int MOSCOW;
4
5 struct WorldMap {
6     struct {
7         int capital;
8         float lat, lon, hei;
9     } countries[1000];
10 };
11
12 int main() {
13     struct WorldMap map;
14     BEIJING = 1;
15     NEWYORK = 2;
16     MOSCOW = 3;
17     map.countries[1].capital = BEIJING;
18     map.countries[2].capital = NEWYORK;
19     map.countries[3].capital = BEIJING, MOSCOW;
20     return map;
21 }

```

输出

```

1 Error type B at Line 19: Unexpected ", "

```

说明：C-不允许使用“,”操作符。

1.8 A-8

输入

```

1 struct A {} a[100];
2 struct B {};
3

```

```

4 int eq(struct A a, struct B b) {
5     return a == b;
6 }
7
8 int main() {
9     struct B b[100];
10    if (eq(A, B)) {
11        return a[1];
12    } else
13        return b[200 * 30];
14    return A[d[b[(1 * ((5) + (b - a))) / 3 + a[A]]] * d.b[]];
15 }

```

输出

```

1 Error type B at Line 14: Empty dimension

```

说明：第 14 行，数组使用 d.b[] 缺少维度。

1.9 A-9

输入

```

1 int main()
2 {
3     int i = 0, j, t, a[5];
4     while(i < 5)
5     {
6         a[i] = read();
7         i = i + 1;
8     }
9     i = 0;
10    while(i < 4)
11    {
12        j = i + 1;
13        while(j < 5)

```

```

14     {
15         if(a[i] > a[j])
16         {
17             int d[2.2];
18             t = a[i];
19             a[i] = a[j];
20             a[j] = t;
21         }
22         j = j + 1;
23     }
24     i = i + 1;
25 }
26 i = 0;
27 while(i < 5)
28 {
29     write(a[i]);
30     i = i + 1;
31 }
32 return 0;
33 }

```

输出

```

1 Error type B at Line 17: Definition of an array only accepts an INT
  dimension

```

说明：第 17 行数组维度不可使用浮点数。

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 change(int m) {
2      int n[2] = m;
3      int j = 0;
4      while (j < (n) {
5          n = n - 2;
6          j = j + 1;
7      }
8      return n;
9  }
10
11 int main() {
12     int a[5];
13     int i = 0;
14     struct A { int a; };
15     a[i] = 100;
16     while (i < 4) {
17         i = i + 1;
18         a[i] = change(a[i-1]);
19         write(a[i]);
20     }
21     return 0;
22 }

```

输出

```

1 Error type B at Line 4: Missing ")"
2 Error type B at Line 14: Unexpected ";"

```

说明：第 4 行括号不匹配，第 14 行结构体定义后缺少变量名（注意，语句块内定义的结构体需立即使用）。

2.2 B-2

输入


```

1  int big(int x, int y) {
2      if (x >= y ) return 0;
3      return 1;
4  }
5
6  int main(){
7      int a[2], b[2], c[4.4];
8      int ia, ib, ic;
9      ia = 0;
10     while (ia < 2) {
11         a[ia] = read();
12         ia = ia + 1;
13     }
14     ib = 0;
15     while (ib < 2) {
16         b[ib] = read();
17         ib = ib + 1;
18     }
19     int a[4];
20     ic = 0;
21     ia = 0;
22     ib = 0;
23     while (ic <= 4) {
24         if (ia >= 2) {
25             c[ic] = b[ib];
26             ib = ib + 1;
27         } else if (ib >= 2) {
28             c[ic] = a[ia];
29             ia = ia + 1;
30         } else if (big(a[ia], b[ib])==0) {
31             c[ic] = a[ia];
32             ia = ia + 1;

```

```

33     } else {
34         c[ic] = b[ib];
35         ib = ib + 1;
36     }
37     ic = ic + 1;
38 }
39 ic = 0
40 while (ic < 4) {
41     write(c[ic]);
42     ic = ic + 1;
43 }
44 return 0;
45 }

```

输出

```

1 Error type B at Line 7: Defition of an array only accepts an INT
  dimension
2 Error type B at Line 19: Illegal declaration
3 Error type B at Line 23: Unexpected "="
4 Error type B at Line 39: Missing ";"

```

说明：第 7 行维度不允许浮点数；第 19 行变量声明仅允许出现在语句块头部；第 23 行 `< =` 多了一个空格；第 39 行缺少分号。

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 int look_and_say(int x) {
2     int _1[32], i = 0;

```

```

3  int _2[64], j = 0, k = 0;
4  int count = 0;
5
6  if (x < 0) {
7      x = ---x;
8  }
9
10 i = 0;
11 while (x > 0) {
12     _1[i] = lsd(x);
13     x = x / 10;
14     i = i + 1;
15 }
16
17 i = i - 1;
18 j = 0;
19 while (i >= 0) {
20     if (i == 0) {
21         _2[j] = 1;
22         _2[j + 1] = _1[0];
23         j = j + 2;
24         i = -1;
25     } else {
26         count = 1;
27         while (i > 0 && _1[i - 1] == _1[i]) {
28             count = count + 1;
29             i = i - 1;
30         }
31         _2[j] = count;
32         _2[j + 1] = _1[i];
33         i = i - 1;
34         j = j + 2;

```

```

35     }
36 }
37
38 k = 0;
39 while (k < j) {
40     write(_2[k]);
41     k = k + 1;
42 }
43
44 return 0;
45 }

```

输出

```

1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         TYPE: int
6       FunDec (1)
7         ID: look_and_say
8         LP
9         VarList (1)
10          ParamDec (1)
11            Specifier (1)
12              TYPE: int
13            VarDec (1)
14              ID: x
15          RP
16        CompSt (1)
17          LC
18          DefList (2)
19            Def (2)
20              Specifier (2)

```

```

21         TYPE: int
22     DecList (2)
23         Dec (2)
24             VarDec (2)
25                 VarDec (2)
26                     ID: _1
27                     LB
28                     INT: 32
29                     RB
30         COMMA
31     DecList (2)
32         Dec (2)
33             VarDec (2)
34                 ID: i
35             ASSIGNOP
36             Exp (2)
37                 INT: 0
38     SEMI
39 DefList (3)
40     Def (3)
41         Specifier (3)
42             TYPE: int
43         DecList (3)
44             Dec (3)
45                 VarDec (3)
46                     VarDec (3)
47                         ID: _2
48                         LB
49                         INT: 64
50                         RB
51         COMMA
52     DecList (3)

```

```

53         Dec (3)
54         VarDec (3)
55         ID: j
56         ASSIGNOP
57         Exp (3)
58         INT: 0
59         COMMA
60         DecList (3)
61         Dec (3)
62         VarDec (3)
63         ID: k
64         ASSIGNOP
65         Exp (3)
66         INT: 0
67     SEMI
68     DefList (4)
69     Def (4)
70     Specifier (4)
71     TYPE: int
72     DecList (4)
73     Dec (4)
74     VarDec (4)
75     ID: count
76     ASSIGNOP
77     Exp (4)
78     INT: 0
79     SEMI
80     StmtList (6)
81     Stmt (6)
82     IF
83     LP
84     Exp (6)

```

85	Exp (6)
86	ID: x
87	RELOP
88	Exp (6)
89	INT: 0
90	RP
91	Stmt (6)
92	CompSt (6)
93	LC
94	StmtList (7)
95	Stmt (7)
96	Exp (7)
97	Exp (7)
98	ID: x
99	ASSIGNOP
100	Exp (7)
101	MINUS
102	Exp (7)
103	MINUS
104	Exp (7)
105	MINUS
106	Exp (7)
107	ID: x
108	SEMI
109	RC
110	StmtList (10)
111	Stmt (10)
112	Exp (10)
113	Exp (10)
114	ID: i
115	ASSIGNOP
116	Exp (10)

```

117         INT: 0
118     SEMI
119 StmtList (11)
120     Stmt (11)
121         WHILE
122             LP
123             Exp (11)
124                 Exp (11)
125                     ID: x
126                 RELOP
127                 Exp (11)
128                     INT: 0
129             RP
130     Stmt (11)
131         CompSt (11)
132             LC
133             StmtList (12)
134                 Stmt (12)
135                     Exp (12)
136                         Exp (12)
137                             Exp (12)
138                                 ID: _1
139                             LB
140                                 Exp (12)
141                                     ID: i
142                                 RB
143                             ASSIGNOP
144                                 Exp (12)
145                                     ID: lsd
146                                 LP
147                                     Args (12)
148                                         Exp (12)

```


149	ID: x
150	RP
151	SEMI
152	StmtList (13)
153	Stmt (13)
154	Exp (13)
155	Exp (13)
156	ID: x
157	ASSIGNOP
158	Exp (13)
159	Exp (13)
160	ID: x
161	DIV
162	Exp (13)
163	INT: 10
164	SEMI
165	StmtList (14)
166	Stmt (14)
167	Exp (14)
168	Exp (14)
169	ID: i
170	ASSIGNOP
171	Exp (14)
172	Exp (14)
173	ID: i
174	PLUS
175	Exp (14)
176	INT: 1
177	SEMI
178	RC
179	StmtList (17)
180	Stmt (17)

181	Exp (17)
182	Exp (17)
183	ID: i
184	ASSIGNOP
185	Exp (17)
186	Exp (17)
187	ID: i
188	MINUS
189	Exp (17)
190	INT: 1
191	SEMI
192	StmtList (18)
193	Stmt (18)
194	Exp (18)
195	Exp (18)
196	ID: j
197	ASSIGNOP
198	Exp (18)
199	INT: 0
200	SEMI
201	StmtList (19)
202	Stmt (19)
203	WHILE
204	LP
205	Exp (19)
206	Exp (19)
207	ID: i
208	RELOP
209	Exp (19)
210	INT: 0
211	RP
212	Stmt (19)

213	CompSt (19)
214	LC
215	StmtList (20)
216	Stmt (20)
217	IF
218	LP
219	Exp (20)
220	Exp (20)
221	ID: i
222	RELOP
223	Exp (20)
224	INT: 0
225	RP
226	Stmt (20)
227	CompSt (20)
228	LC
229	StmtList (21)
230	Stmt (21)
231	Exp (21)
232	Exp (21)
233	Exp (21)
234	ID: _2
235	LB
236	Exp (21)
237	ID: j
238	RB
239	ASSIGNOP
240	Exp (21)
241	INT: 1
242	SEMI
243	StmtList (22)
244	Stmt (22)

245	Exp (22)
246	Exp (22)
247	Exp (22)
248	ID: _2
249	LB
250	Exp (22)
251	Exp (22)
252	ID: j
253	PLUS
254	Exp (22)
255	INT: 1
256	RB
257	ASSIGNOP
258	Exp (22)
259	Exp (22)
260	ID: _1
261	LB
262	Exp (22)
263	INT: 0
264	RB
265	SEMI
266	StmtList (23)
267	Stmt (23)
268	Exp (23)
269	Exp (23)
270	ID: j
271	ASSIGNOP
272	Exp (23)
273	Exp (23)
274	ID: j
275	PLUS
276	Exp (23)

277	INT: 2
278	SEMI
279	StmtList (24)
280	Stmt (24)
281	Exp (24)
282	Exp (24)
283	ID: i
284	ASSIGNOP
285	Exp (24)
286	MINUS
287	Exp (24)
288	INT: 1
289	SEMI
290	RC
291	ELSE
292	Stmt (25)
293	CompSt (25)
294	LC
295	StmtList (26)
296	Stmt (26)
297	Exp (26)
298	Exp (26)
299	ID: count
300	ASSIGNOP
301	Exp (26)
302	INT: 1
303	SEMI
304	StmtList (27)
305	Stmt (27)
306	WHILE
307	LP
308	Exp (27)

309	Exp (27)
310	Exp (27)
311	ID: i
312	RELOP
313	Exp (27)
314	INT: 0
315	AND
316	Exp (27)
317	Exp (27)
318	Exp (27)
319	ID: _1
320	LB
321	Exp (27)
322	Exp (27)
323	ID: i
324	MINUS
325	Exp (27)
326	INT: 1
327	RB
328	RELOP
329	Exp (27)
330	Exp (27)
331	ID: _1
332	LB
333	Exp (27)
334	ID: i
335	RB
336	RP
337	Stmt (27)
338	CompSt (27)
339	LC
340	StmtList (28)

341	Stmt (28)
342	Exp (28)
343	Exp (28)
344	ID: count
345	ASSIGNOP
346	Exp (28)
347	Exp (28)
348	ID: count
349	PLUS
350	Exp (28)
351	INT: 1
352	SEMI
353	StmtList (29)
354	Stmt (29)
355	Exp (29)
356	Exp (29)
357	ID: i
358	ASSIGNOP
359	Exp (29)
360	Exp (29)
361	ID: i
362	MINUS
363	Exp (29)
364	INT: 1
365	SEMI
366	RC
367	StmtList (31)
368	Stmt (31)
369	Exp (31)
370	Exp (31)
371	Exp (31)
372	ID: _2

373	LB
374	Exp (31)
375	ID: j
376	RB
377	ASSIGNOP
378	Exp (31)
379	ID: count
380	SEMI
381	StmtList (32)
382	Stmt (32)
383	Exp (32)
384	Exp (32)
385	Exp (32)
386	ID: _2
387	LB
388	Exp (32)
389	Exp (32)
390	ID: j
391	PLUS
392	Exp (32)
393	INT: 1
394	RB
395	ASSIGNOP
396	Exp (32)
397	Exp (32)
398	ID: _1
399	LB
400	Exp (32)
401	ID: i
402	RB
403	SEMI
404	StmtList (33)

405		Stmt (33)
406		Exp (33)
407		Exp (33)
408		ID: i
409		ASSIGNOP
410		Exp (33)
411		Exp (33)
412		ID: i
413		MINUS
414		Exp (33)
415		INT: 1
416		SEMI
417		StmtList (34)
418		Stmt (34)
419		Exp (34)
420		Exp (34)
421		ID: j
422		ASSIGNOP
423		Exp (34)
424		Exp (34)
425		ID: j
426		PLUS
427		Exp (34)
428		INT: 2
429		SEMI
430	RC	
431	RC	
432	StmtList (38)	
433	Stmt (38)	
434	Exp (38)	
435	Exp (38)	
436	ID: k	

437	ASSIGNOP
438	Exp (38)
439	INT: 0
440	SEMI
441	StmtList (39)
442	Stmt (39)
443	WHILE
444	LP
445	Exp (39)
446	Exp (39)
447	ID: k
448	RELOP
449	Exp (39)
450	ID: j
451	RP
452	Stmt (39)
453	CompSt (39)
454	LC
455	StmtList (40)
456	Stmt (40)
457	Exp (40)
458	ID: write
459	LP
460	Args (40)
461	Exp (40)
462	Exp (40)
463	ID: _2
464	LB
465	Exp (40)
466	ID: k
467	RB
468	RP

```

469             SEMI
470         StmtList (41)
471             Stmt (41)
472                 Exp (41)
473                     Exp (41)
474                         ID: k
475                     ASSIGNOP
476                 Exp (41)
477                     Exp (41)
478                         ID: k
479                     PLUS
480                 Exp (41)
481                     INT: 1
482             SEMI
483         RC
484     StmtList (44)
485         Stmt (44)
486             RETURN
487             Exp (44)
488                 INT: 0
489         SEMI
490 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 intset;
2 struct uint8_t;
3 struct int64_t;
4
5 struct uint8_t intset_search(struct intset is, struct int64_t value,
    struct uint32_t pos) {

```

```

6      int min = 0, max = intrev32ifbe(is.length)-1, mid = -1;
7      struct int64_t cur = -1;
8
9      if (intrev32ifbe(is.length) == 0) {
10         if (pos) pos = 0;
11         return 0;
12     } else {
13         if (value > intset_get(is,max)) {
14             if (pos) pos = intrev32ifbe(is.length);
15             return 0;
16         } else if (value < intset_get(is,0)) {
17             if (pos) pos = 0;
18             return 0;
19         }
20     }
21
22     while(max >= min) {
23         mid = shr((min + max), 1);
24         cur = intset_get(is, mid);
25         if (value > cur) {
26             min = mid + 1;
27         } else if (value < cur) {
28             max = mid-1;
29         } else {
30             break;
31         }
32     }
33
34     if (value == cur) {
35         if (pos) pos = mid;
36         return 1;
37     } else {

```

```

38     if (pos) pos = min;
39     return 0;
40 }
41 }

```

输出

```

1 Program (1)
2   ExtDefList (1)
3     ExtDef (1)
4       Specifier (1)
5         StructSpecifier (1)
6           STRUCT
7           Tag (1)
8             ID: intset
9     SEMI
10  ExtDefList (2)
11    ExtDef (2)
12      Specifier (2)
13        StructSpecifier (2)
14          STRUCT
15          Tag (2)
16            ID: uint8_t
17    SEMI
18  ExtDefList (3)
19    ExtDef (3)
20      Specifier (3)
21        StructSpecifier (3)
22          STRUCT
23          Tag (3)
24            ID: int64_t
25    SEMI
26  ExtDefList (5)
27    ExtDef (5)

```

```

28     Specifier (5)
29         StructSpecifier (5)
30             STRUCT
31             Tag (5)
32                 ID: uint8_t
33 FunDec (5)
34     ID: intset_search
35     LP
36     VarList (5)
37         ParamDec (5)
38             Specifier (5)
39                 StructSpecifier (5)
40                     STRUCT
41                     Tag (5)
42                         ID: intset
43         VarDec (5)
44             ID: is
45     COMMA
46     VarList (5)
47         ParamDec (5)
48             Specifier (5)
49                 StructSpecifier (5)
50                     STRUCT
51                     Tag (5)
52                         ID: int64_t
53         VarDec (5)
54             ID: value
55     COMMA
56     VarList (5)
57         ParamDec (5)
58             Specifier (5)
59                 StructSpecifier (5)

```

```

60          STRUCT
61          Tag (5)
62          ID: uint32_t
63          VarDec (5)
64          ID: pos
65      RP
66  CompSt (5)
67      LC
68      DefList (6)
69          Def (6)
70              Specifier (6)
71              TYPE: int
72              DecList (6)
73                  Dec (6)
74                      VarDec (6)
75                          ID: min
76                          ASSIGNOP
77                          Exp (6)
78                          INT: 0
79                  COMMA
80                  DecList (6)
81                      Dec (6)
82                          VarDec (6)
83                              ID: max
84                              ASSIGNOP
85                              Exp (6)
86                                  Exp (6)
87                                      ID: intrev32ifbe
88                                      LP
89                                      Args (6)
90                                          Exp (6)
91                                              Exp (6)

```

```

92             ID: is
93             DOT
94             ID: length
95             RP
96             MINUS
97             Exp (6)
98             INT: 1
99         COMMA
100        Declist (6)
101        Dec (6)
102        VarDec (6)
103        ID: mid
104        ASSIGNOP
105        Exp (6)
106        MINUS
107        Exp (6)
108        INT: 1
109    SEMI
110    DefList (7)
111    Def (7)
112    Specifier (7)
113    StructSpecifier (7)
114    STRUCT
115    Tag (7)
116    ID: int64_t
117    Declist (7)
118    Dec (7)
119    VarDec (7)
120    ID: cur
121    ASSIGNOP
122    Exp (7)
123    MINUS

```



```

124             Exp (7)
125             INT: 1
126         SEMI
127     StmtList (9)
128     Stmt (9)
129     IF
130     LP
131     Exp (9)
132     Exp (9)
133     ID: intrev32ifbe
134     LP
135     Args (9)
136     Exp (9)
137     Exp (9)
138     ID: is
139     DOT
140     ID: length
141     RP
142     RELOP
143     Exp (9)
144     INT: 0
145     RP
146     Stmt (9)
147     CompSt (9)
148     LC
149     StmtList (10)
150     Stmt (10)
151     IF
152     LP
153     Exp (10)
154     ID: pos
155     RP

```

```

156         Stmt (10)
157         Exp (10)
158         Exp (10)
159         ID: pos
160         ASSIGNOP
161         Exp (10)
162         INT: 0
163         SEMI
164     StmtList (11)
165     Stmt (11)
166     RETURN
167     Exp (11)
168     INT: 0
169     SEMI
170 RC
171 ELSE
172 Stmt (12)
173     CompSt (12)
174     LC
175     StmtList (13)
176     Stmt (13)
177     IF
178     LP
179     Exp (13)
180     Exp (13)
181     ID: value
182     RELOP
183     Exp (13)
184     ID: intset_get
185     LP
186     Args (13)
187     Exp (13)

```

188	ID: is
189	COMMA
190	Args (13)
191	Exp (13)
192	ID: max
193	RP
194	RP
195	Stmt (13)
196	CompSt (13)
197	LC
198	StmtList (14)
199	Stmt (14)
200	IF
201	LP
202	Exp (14)
203	ID: pos
204	RP
205	Stmt (14)
206	Exp (14)
207	Exp (14)
208	ID: pos
209	ASSIGNOP
210	Exp (14)
211	ID: intrev32ifbe
212	LP
213	Args (14)
214	Exp (14)
215	Exp (14)
216	ID: is
217	DOT
218	ID: length
219	RP

220	SEMI
221	StmtList (15)
222	Stmt (15)
223	RETURN
224	Exp (15)
225	INT: 0
226	SEMI
227	RC
228	ELSE
229	Stmt (16)
230	IF
231	LP
232	Exp (16)
233	Exp (16)
234	ID: value
235	RELOP
236	Exp (16)
237	ID: intset_get
238	LP
239	Args (16)
240	Exp (16)
241	ID: is
242	COMMA
243	Args (16)
244	Exp (16)
245	INT: 0
246	RP
247	RP
248	Stmt (16)
249	CompSt (16)
250	LC
251	StmtList (17)

252	Stmt (17)	
253	IF	
254	LP	
255	Exp (17)	
256	ID: pos	
257	RP	
258	Stmt (17)	
259	Exp (17)	
260	Exp (17)	
261	ID: pos	
262	ASSIGNOP	
263	Exp (17)	
264	INT: 0	
265	SEMI	
266	StmtList (18)	
267	Stmt (18)	
268	RETURN	
269	Exp (18)	
270	INT: 0	
271	SEMI	
272	RC	
273	RC	
274	StmtList (22)	
275	Stmt (22)	
276	WHILE	
277	LP	
278	Exp (22)	
279	Exp (22)	
280	ID: max	
281	RELOP	
282	Exp (22)	
283	ID: min	

284	RP
285	Stmt (22)
286	CompSt (22)
287	LC
288	StmtList (23)
289	Stmt (23)
290	Exp (23)
291	Exp (23)
292	ID: mid
293	ASSIGNOP
294	Exp (23)
295	ID: shr
296	LP
297	Args (23)
298	Exp (23)
299	LP
300	Exp (23)
301	Exp (23)
302	ID: min
303	PLUS
304	Exp (23)
305	ID: max
306	RP
307	COMMA
308	Args (23)
309	Exp (23)
310	INT: 1
311	RP
312	SEMI
313	StmtList (24)
314	Stmt (24)
315	Exp (24)

316	Exp (24)
317	ID: cur
318	ASSIGNOP
319	Exp (24)
320	ID: intset_get
321	LP
322	Args (24)
323	Exp (24)
324	ID: is
325	COMMA
326	Args (24)
327	Exp (24)
328	ID: mid
329	RP
330	SEMI
331	StmtList (25)
332	Stmt (25)
333	IF
334	LP
335	Exp (25)
336	Exp (25)
337	ID: value
338	RELOP
339	Exp (25)
340	ID: cur
341	RP
342	Stmt (25)
343	CompSt (25)
344	LC
345	StmtList (26)
346	Stmt (26)
347	Exp (26)

348	Exp (26)
349	ID: min
350	ASSIGNOP
351	Exp (26)
352	Exp (26)
353	ID: mid
354	PLUS
355	Exp (26)
356	INT: 1
357	SEMI
358	RC
359	ELSE
360	Stmt (27)
361	IF
362	LP
363	Exp (27)
364	Exp (27)
365	ID: value
366	RELOP
367	Exp (27)
368	ID: cur
369	RP
370	Stmt (27)
371	CompSt (27)
372	LC
373	StmtList (28)
374	Stmt (28)
375	Exp (28)
376	Exp (28)
377	ID: max
378	ASSIGNOP
379	Exp (28)


```

380                                     Exp (28)
381                                     ID: mid
382                                     MINUS
383                                     Exp (28)
384                                     INT: 1
385                                     SEMI
386                                     RC
387                                     ELSE
388                                     Stmt (29)
389                                     CompSt (29)
390                                     LC
391                                     StmtList (30)
392                                     Stmt (30)
393                                     Exp (30)
394                                     ID: break
395                                     SEMI
396                                     RC
397                                     RC
398                                     StmtList (34)
399                                     Stmt (34)
400                                     IF
401                                     LP
402                                     Exp (34)
403                                     Exp (34)
404                                     ID: value
405                                     RELOP
406                                     Exp (34)
407                                     ID: cur
408                                     RP
409                                     Stmt (34)
410                                     CompSt (34)
411                                     LC

```

412	StmtList (35)
413	Stmt (35)
414	IF
415	LP
416	Exp (35)
417	ID: pos
418	RP
419	Stmt (35)
420	Exp (35)
421	Exp (35)
422	ID: pos
423	ASSIGNOP
424	Exp (35)
425	ID: mid
426	SEMI
427	StmtList (36)
428	Stmt (36)
429	RETURN
430	Exp (36)
431	INT: 1
432	SEMI
433	RC
434	ELSE
435	Stmt (37)
436	CompSt (37)
437	LC
438	StmtList (38)
439	Stmt (38)
440	IF
441	LP
442	Exp (38)
443	ID: pos

```

444         RP
445         Stmt (38)
446         Exp (38)
447         Exp (38)
448         ID: pos
449         ASSIGNOP
450         Exp (38)
451         ID: min
452         SEMI
453         StmtList (39)
454         Stmt (39)
455         RETURN
456         Exp (39)
457         INT: 0
458         SEMI
459     RC
460 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 x1 = 0703;
3     int x2 = 0642;
4     int x3 = 0x00ABC20f - 0x08048000 * 0XfFffFE - 000256;
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: x1
20                ASSIGNOP
21                Exp (2)
22                  INT: 451
23          SEMI
24        DefList (3)
25          Def (3)
26            Specifier (3)
27              TYPE: int
28            DecList (3)
29              Dec (3)
30                VarDec (3)
31                  ID: x2
```

```

32         ASSIGNOP
33         Exp (3)
34             INT: 418
35     SEMI
36 DefList (4)
37     Def (4)
38         Specifier (4)
39             TYPE: int
40         DeclList (4)
41             Dec (4)
42                 VarDec (4)
43                     ID: x3
44                 ASSIGNOP
45                 Exp (4)
46                     Exp (4)
47                         Exp (4)
48                             INT: 11256335
49                         MINUS
50                         Exp (4)
51                             Exp (4)
52                                 INT: 134512640
53                             STAR
54                             Exp (4)
55                                 INT: 16777214
56                         MINUS
57                         Exp (4)
58                             INT: 174
59         SEMI
60 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 e0 = 0.43244e5;  
3     float e1 = .4e-1;  
4     float e2 = 88.E002;  
5     float e3 = e1 + x2 - x4 * .3e+4;  
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: e0  
20                  ASSIGNOP  
21                  Exp (2)  
22                    FLOAT: 43244.000000
```

```

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: e1
32                         ASSIGNOP
33                         Exp (3)
34                             FLOAT: 0.040000
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: e2
44                         ASSIGNOP
45                         Exp (4)
46                             FLOAT: 8800.000000
47         SEMI
48     DefList (5)
49         Def (5)
50             Specifier (5)
51                 TYPE: float
52             DecList (5)
53                 Dec (5)
54                     VarDec (5)

```

```

55         ID: e3
56     ASSIGNOP
57     Exp (5)
58         Exp (5)
59             Exp (5)
60                 ID: e1
61             PLUS
62             Exp (5)
63                 ID: x2
64             MINUS
65             Exp (5)
66                 Exp (5)
67                     ID: x4
68                 STAR
69                 Exp (5)
70                     FLOAT: 3000.000000
71     SEMI
72 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  /* Pseudo random number generation functions derived from the drand48
   *
   * function obtained from pysam source code.
   *
   * This functions are used in order to replace the default math.
   * random()
   *
   * Lua implementation with something having exactly the same behavior
   * across different systems (by default Lua uses libc's rand() that
   * is not

```



```

7  * required to implement a specific PRNG generating the same sequence
8  * in different systems if seeded with the same integer).
9  *
10 * The original code appears to be under the public domain.
11 * I modified it removing the non needed functions and all the
12 * 1960-style C coding stuff...
13 *
14 *
    -----
15 *
16 * Copyright (c) 2010-2012, Salvatore Sanfilippo <antirez at gmail
    dot com>
17 * All rights reserved.
18 *
19 * Redistribution and use in source and binary forms, with or without
20 * modification, are permitted provided that the following conditions
    are met:
21 *
22 *   * Redistributions of source code must retain the above copyright
    notice,
23 *   this list of conditions and the following disclaimer.
24 *   * Redistributions in binary form must reproduce the above
    copyright
25 *   notice, this list of conditions and the following disclaimer
    in the
26 *   documentation and/or other materials provided with the
    distribution.
27 *   * Neither the name of Redis nor the names of its contributors
    may be used
28 *   to endorse or promote products derived from this software
    without

```

```

29 *      specific prior written permission.
30 *
31 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
    CONTRIBUTORS "AS IS"
32 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
    TO, THE
33 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
    PURPOSE
34 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
    CONTRIBUTORS BE
35 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
    OR
36 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
    OF
37 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
    BUSINESS
38 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
    WHETHER IN
39 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
    OTHERWISE)
40 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    ADVISED OF THE
41 * POSSIBILITY OF SUCH DAMAGE.
42 */
43
44 struct long;
45 struct unsigned;
46 struct int32_t;
47 struct uint32_t;
48
49 struct uint32_t x[3], a[3], c;
50

```

```

51 int N, MASK;
52 int X0, X1, X2, A0, A1, A2, C;
53 struct long HI_BIT;
54
55 struct uint32_t LOW(struct uint32_t x) { return and(x, MASK); }
56 struct uint32_t HIGH(struct uint32_t x) { return LOW(shr((x), N)); }
57 int MUL(struct uint32_t x, struct uint32_t y, struct uint32_t z) {
58     struct uint32_t l = x * y;
59     (z)[0] = LOW(l);
60     (z)[1] = HIGH(l);
61     return 0;
62 }
63
64 int CARRY(struct uint32_t x, struct long y) { return x + y > MASK; }
65 int ADDEQU(struct uint32_t x, struct uint32_t y) { int z = CARRY(x, y
    ); x = LOW(x + y); return z; }
66 int SET3(struct uint32_t x[3], struct uint32_t x0, struct uint32_t x1
    , struct uint32_t x2) { (x)[0] = (x0); (x)[1] = (x1); (x)[2] = (x2
    ); }
67 int SEED(struct uint32_t x0, struct uint32_t x1, struct uint32_t x2)
    { SET3(x, x0, x1, x2); SET3(a, A0, A1, A2); c = C; }
68 int REST(int tv) {
69     int i;
70     while (i < 3) {
71         xsubi[i] = x[i]; x[i] = temp[i];
72         i = i + 1;
73     }
74     return (v);
75 }
76
77 int init() {
78     N = 16;

```

```

79     MASK = (shl(1, (N - 1)) + shl(1, (N - 1)) - 1);
80
81     X0 = 33;
82     X1 = 34;
83     X2 = 234;
84     A0 = 66;
85     A1 = 4;
86     A2 = 5;
87     C  = 8;
88
89     HI_BIT = shl(1, (2 * N - 1));
90
91     x[0] = X0;
92     x[1] = X1;
93     x[2] = X2;
94     a[1] = A0;
95     a[2] = A1;
96     a[3] = A2;
97     c    = C;
98 }
99
100 struct int32_t redisLrand48() {
101     struct int32_t ret;
102     next();
103     ret = (shl(x[2], (N - 1)) + shr(x[1], 1));
104     return ret;
105 }
106
107 int redisSrand48(struct int32_t seedval) {
108     SEED(X0, LOW(seedval), HIGH(seedval));
109     return 0;
110 }

```

```

111
112 int next() {
113     struct uint32_t p[2], q[2], r[2], carry0, carry1;
114
115     MUL(a[0], x[0], p);
116     ADDEQU(p[0], c, carry0);
117     ADDEQU(p[1], carry0, carry1);
118     MUL(a[0], x[1], q);
119     ADDEQU(p[1], q[0], carry0);
120     MUL(a[1], x[0], r);
121     x[2] = LOW(carry0 + carry1 + CARRY(p[1], r[0]) + q[1] + r[1] +
122             a[0] * x[2] + a[1] * x[1] + a[2] * x[0]);
123     x[1] = LOW(p[1] + r[0]);
124     x[0] = LOW(p[0]);
125
126     return 0;
127 }
128
129 /**
130  * Common block comment
131  */
132 int /** break definition */ myFunc() {
133     int abc = 123; // commom line comment
134     int abc = 123 // break line
135     ;
136     int abc = 3 /** break line too
137     */, def = 4;
138     int def = /** wierd block comment /***** *?
139         // recurrsive comments
140         int main() {
141             int abc = 123; // commom line comment
142             int abc = 123 // break line

```

```

143 ;
144 int abc = 3 /** break line too
145 *x/, def = 4;
146 int def = /** wierd block comment /***** */
147 // should recursive, stop it
148 ****x/ 8865;
149 }
150 */ 8865;
151 struct Def def = //\\*//\\*//\\\\//\\*//\\
152 abc;
153 }
```

输出

1 // 太长，暂不放在该文件中。请在群文件里下载。

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 (E1-1)

```
1 int main() {
2     int x1 = 00703;
3     int x2 = 0604209;
4     float e3 = e1 + x2 - x4;
5     return 0000;
6 }
```

输出

```
1 Error type A at Line 3: Illegal oct INT number: "0604209"
```

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 0604209，识别成错误类型 B 也可以。

输入 (E1-2)

```
1 struct charstar;
2 struct uint16_t crc16tab[256];
3
4 struct uint16_t crc16(struct charstar buf, int len) {
5     int counter;
6     struct uint16_t crc;
7
8     crc16tab = (
9         0x0000 && 0x1021 && 0x2042 && 0x3063 && 0x4084 && 0x50a5 && 0
10         x60c6 && 0x70e7 &&
11         0x8108 && 0x9129 && 0xa14a && 0xb16b && 0xc18c && 0xd1ad && 0
12         xe1ce && 0xf1ef &&
13         0x1231 && 0x0210 && 0x3273 && 0x2252 && 0x52b5 && 0x4294 && 0
14         x72f7 && 0x62d6 &&
15         0x9339 && 0x8318 && 0xb37b && 0xa35a && 0xd3bd && 0xc39c && 0
16         xf3ff && 0xe3de &&
17         0x2462 && 0x3443 && 0x0420 && 0x1401 && 0x64e6 && 0x74c7 && 0
18         x44a4 && 0x5485 &&
19         0xa56a && 0xb54b && 0x8528 && 0x9509 && 0xe5ee && 0xf5cf && 0
20         xc5ac && 0xd58d &&
21         0x3653 && 0x2672 && 0x1611 && 0x0630 && 0x76d7 && 0x66f6 && 0
22         x5695 && 0x46b4 &&
23         0xb75b && 0xa77a && 0x9719 && 0x8738 && 0xf7df && 0xe7fe && 0
24         xd79d && 0xc7bc &&
25         0x48c4 && 0x58e5 && 0x6886 && 0x78a7 && 0x0840 && 0x1861 && 0
26         x2802 && 0x3823 &&
27         0xc9cc && 0xd9ed && 0xe98e && 0xf9af && 0x8948 && 0x9969 && 0
```

```

xa90a && 0xb92b &&
19 0x5af5 && 0x4ad4 && 0x7ab7 && 0x6u96 && 0x1a71 && 0x0a50 && 0
    x3a33 && 0x2a12 &&
20 0xdbfd && 0xcbdc && 0xfbbf && 0xeb9e && 0x9b79 && 0x8b58 && 0
    xbb3b && 0xab1a &&
21 0x6ca6 && 0x7c87 && 0x4ce4 && 0x5cc5 && 0x2a22 && 0x3c03 && 0
    x0c60 && 0x1c41 &&
22 0xedaе && 0xfd8f && 0xcdec && 0xddcd && 0xad2a && 0xbd0b && 0
    x8d68 && 0x9d49 &&
23 0x7e97 && 0x6eb6 && 0x5ed5 && 0x4ef4 && 0x3e13 && 0x2e32 && 0
    x1e51 && 0x0e70 &&
24 0xff9f && 0xefbe && 0xdfdd && 0xcffc && 0xbf1b && 0xaf3a && 0
    x9f59 && 0x8f78 &&
25 0x9188 && 0x81a9 && 0xb1ca && 0xa1eb && 0xd10c && 0xc12d && 0
    xf14e && 0xe16f &&
26 0x1080 && 0x00a1 && 0x30c2 && 0x20e3 && 0x5u04 && 0x4025 && 0
    x7046 && 0x6067 &&
27 0x83b9 && 0x9398 && 0xa3fb && 0xb3da && 0xc33d && 0xd31c && 0
    xe37f && 0xf35e &&
28 0x02b1 && 0x1290 && 0x22f3 && 0x32d2 && 0x4235 && 0x5214 && 0
    x6277 && 0x7256 &&
29 0xb5ea && 0xa5cb && 0x95a8 && 0x8589 && 0xf56e && 0xe54f && 0
    xd52c && 0xc50d &&
30 0x34e2 && 0x24c3 && 0x14a0 && 0x0481 && 0x7466 && 0x6447 && 0
    x5424 && 0x4405 &&
31 0xa7db && 0xb7fa && 0x8799 && 0x97b8 && 0xe75f && 0xf77e && 0
    xc71d && 0xd73c &&
32 0x26d3 && 0x36f2 && 0x0691 && 0x16b0 && 0x6657 && 0x7676 && 0
    x4615 && 0x5634 &&
33 0xd94c && 0xc96d && 0xo90e && 0xe92f && 0x99c8 && 0x89e9 && 0
    xb98a && 0xa9ab &&
34 0x5844 && 0x4865 && 0x7806 && 0x6827 && 0x18c0 && 0x08e1 && 0

```



```

        x3882 && 0x28a3 &&
35      0xcb7d && 0xdb5c && 0xeb3f && 0xfb1e && 0x8bf9 && 0x9bd8 && 0
        xabbb && 0xbb9a &&
36      0x4a75 && 0x5a54 && 0x6a37 && 0x7a16 && 0x0af1 && 0x1ad0 && 0
        x2ab3 && 0x3a92 &&
37      0xfd2e && 0xed0f && 0xdd6c && 0xcd4d && 0xbdaa && 0xad8b && 0
        x9de8 && 0x8dc9 &&
38      0x7c26 && 0x6c07 && 0x5c64 && 0x4c45 && 0x3ca2 && 0x2c83 && 0
        x1ce0 && 0x0cc1 &&
39      0xef1f && 0xff3e && 0xcf5d && 0xdf7c && 0xaf9b && 0xbfba && 0
        x8fd9 && 0x9ff8 &&
40      0x6e17 && 0x7e36 && 0x4e55 && 0x5e74 && 0x2e93 && 0x3eb2 && 0
        x0ed1 && 0x1ef0
41    );
42    crc = 0;
43    counter = 0;
44
45    while (counter < len) {
46        crc = xor(shl(crc, 8), crc16tab[and(xor(shr(crc, 8), (star(
            buf) + 1)), 0x00FF)]);
47        counter = counter + 1;
48    }
49
50    return crc;
51 }

```

输出

```

1 Error type A at Line 19: Illegal hex INT number: "0x6u96"
2 Error type A at Line 26: Illegal hex INT number: "0x5u04"
3 Error type A at Line 33: Illegal hex INT number: "0xo90e"

```

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 0x6u96 与 0x5u04 与 0xo90e，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 m() {  
2     struct d3_t d3;  
3     float a = 1.e01;  
4     float b = .1E1;  
5     float c = .1e.1;  
6     float d = 0.1e*3;  
7     d3.e5 = 3;  
8     return a * b.e + .1e1 * a1.e1;  
9 }
```

输出

```
1 Error type A at Line 5: Illegal FLOAT number: ".1e.1"  
2 Error type A at Line 6: Illegal FLOAT number: "0.1e"
```

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.1e.1 和 0.1e，识别成错误类型 B 也可以。

输入 (E2-2)

```
1 float m() {  
2     float c = e1e1;  
3     float f = .1e1;  
4     float d = e.e1;  
5     return e1.e1;  
6 }
```

输出

```
1 Program (1)  
2   ExtDefList (1)  
3     ExtDef (1)  
4       Specifier (1)  
5         TYPE: float
```

```

6      FunDec (1)
7          ID: m
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: c
20                         ASSIGNOP
21                         Exp (2)
22                             ID: e1e1
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
32                                 ASSIGNOP
33                                 Exp (3)
34                                     FLOAT: 1.000000
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: d
44         ASSIGNOP
45         Exp (4)
46         Exp (4)
47         ID: e
48         DOT
49         ID: e1
50         SEMI
51     StmtList (5)
52     Stmt (5)
53     RETURN
54     Exp (5)
55     Exp (5)
56     ID: e1
57     DOT
58     ID: e1
59     SEMI
60 RC

```

说明：仅 1.2 分组的同学需要测试这个用例。

5.3 E1.3

这组测试用例针对 1.3 分组的同学。

输入 (E3-1)

```

1  /* adlist.c - A generic doubly linked list implementation
2  *
3  * Copyright (c) 2006-2010, Salvatore Sanfilippo <antirez at gmail
   dot com>

```

```

4  * All rights reserved.
5  *
6  * Redistribution and use in source and binary forms, with or without
7  * modification, are permitted provided that the following conditions
   * are met:
8  *
9  *   * Redistributions of source code must retain the above copyright
   * notice,
10 *       this list of conditions and the following disclaimer.
11 *   * Redistributions in binary form must reproduce the above
   * copyright
12 *       notice, this list of conditions and the following disclaimer
   * in the
13 *       documentation and/or other materials provided with the
   * distribution.
14 *   * Neither the name of Redis nor the names of its contributors
   * may be used
15 *       to endorse or promote products derived from this software
   * without
16 *       specific prior written permission.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
   * CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
   * TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
   * PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
   * CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
   * OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT

```

```

    OF
24  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
    BUSINESS
25  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
    WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
    OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 /* Add a new node to the list, to tail, containing the specified '
    value'
32  * pointer as value.
33  *
34  * On error, NULL is returned and no operation is performed (i.e. the
35  * list remains unaltered).
36  * On success the 'list' pointer you pass to the function is returned
    . */
37 struct liststar listAddNodeTail(struct liststart list, struct
    voidstart value)
38 {
39     struct listNodestart node;
40
41     if ((node = zmalloc(sizeof(star(node)))) == NULL)
42         return
43         /*\}*\/*
44         }*/
45         NULL;
46     node./*?**/value = value;
47     if (list.len == 0) {

```

```

48     list.head = list.tail = node//;
49     node.prev = node.next = NULL;
50 } else {
51     node.prev = list.tail;
52     node.next = NULL;
53     list.tail.next = node;
54     list.tail = node;
55 }
56 list.len = list.len + 1;
57 return list;
58 }
59
60 /**  unclosed block comment
61  *
62  *
63  *
64  ****x?/

```

输出

```

1 Error type B at Line 48: Missing ";"
2 Error type A at Line 60: Unclosed comment

```

说明：仅 1.3 分组的同学需要测试这个用例。第 48 行也可以报在 49 行；第 60 行也可以报在 61、62、63 或者 64 行，也可以报成语法错误。

输入 (E3-2)

```

1  /* adlist.c - A generic doubly linked list implementation
2   *
3   * Copyright (c) 2006-2010, Salvatore Sanfilippo <antirez at gmail
4   *    dot com>
5   * All rights reserved.
6   *
7   * Redistribution and use in source and binary forms, with or without
8   * modification, are permitted provided that the following conditions

```

```

      are met:
8  *
9  *   * Redistributions of source code must retain the above copyright
      notice,
10 *   this list of conditions and the following disclaimer.
11 *   * Redistributions in binary form must reproduce the above
      copyright
12 *   notice, this list of conditions and the following disclaimer
      in the
13 *   documentation and/or other materials provided with the
      distribution.
14 *   * Neither the name of Redis nor the names of its contributors
      may be used
15 *   to endorse or promote products derived from this software
      without
16 *   specific prior written permission.
17 *
18 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
      CONTRIBUTORS "AS IS"
19 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
      TO, THE
20 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
      PURPOSE
21 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
      CONTRIBUTORS BE
22 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
      OR
23 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
      OF
24 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
      BUSINESS
25 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

```



```

    WHETHER IN
26  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
    OTHERWISE)
27  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    ADVISED OF THE
28  * POSSIBILITY OF SUCH DAMAGE.
29  */
30
31 /* Duplicate the whole list. On out of memory NULL is returned.
32  * On success a copy of the original list is returned.
33  *
34  * The 'Dup' method set with listSetDupMethod() function is used
35  * to copy the node value. Otherwise the same pointer value of
36  * the original node is used as value of the copied node.
37  *
38  * The original list both on success or error is never modified. */
39 struct liststar listDup(struct liststar orig)
40 {
41     struct liststar copy;
42     struct listIter iter;
43     struct listNodestar node;
44
45     if ((copy = listCreate()) == NULL)
46         return NULL;
47     copy.dup = orig.dup;
48     copy.free = orig.free;
49     copy.match = orig.match;
50     listRewind(orig, ref(iter));
51     while((node = listNext(ref(iter))) != NULL) {
52         struct voidstart value;
53
54         if (copy.dup) {

```

```

55     value = dup(copy,
56         /*This is a comment.\
57         /\//\//*****\//\//\//
58         if (value == NULL) {
59             listRelease(copy);
60             return NULL;
61         }
62         */
63         node.value);
64     if (value == NULL) {
65         listRelease(copy);
66         return NULL;
67     }
68 } else
69     value = node.value;
70 if (listAddNodeTail(copy, value) == NULL) {
71     listRelease(copy);
72     return NULL;
73 }
74 }
75 return copy;
76 }

```

输出

```

1 Program (39)
2   ExtDefList (39)
3     ExtDef (39)
4       Specifier (39)
5         StructSpecifier (39)
6           STRUCT
7             Tag (39)
8               ID: liststar
9             FunDec (39)

```

```

10      ID: listDup
11      LP
12      VarList (39)
13          ParamDec (39)
14              Specifier (39)
15                  StructSpecifier (39)
16                      STRUCT
17                          Tag (39)
18                              ID: liststar
19      VarDec (39)
20          ID: orig
21      RP
22      CompSt (40)
23      LC
24      DefList (41)
25          Def (41)
26              Specifier (41)
27                  StructSpecifier (41)
28                      STRUCT
29                          Tag (41)
30                              ID: liststar
31      DecList (41)
32          Dec (41)
33              VarDec (41)
34                  ID: copy
35      SEMI
36      DefList (42)
37          Def (42)
38              Specifier (42)
39                  StructSpecifier (42)
40                      STRUCT
41                          Tag (42)

```

```

42         ID: listIter
43     DecList (42)
44         Dec (42)
45         VarDec (42)
46         ID: iter
47     SEMI
48     DefList (43)
49         Def (43)
50             Specifier (43)
51                 StructSpecifier (43)
52                     STRUCT
53                     Tag (43)
54                         ID: listNodestar
55                     DecList (43)
56                         Dec (43)
57                         VarDec (43)
58                         ID: node
59                     SEMI
60     StmtList (45)
61         Stmt (45)
62             IF
63             LP
64             Exp (45)
65                 Exp (45)
66                     LP
67                     Exp (45)
68                         Exp (45)
69                             ID: copy
70                         ASSIGNOP
71                     Exp (45)
72                         ID: listCreate
73                     LP

```

74	RP
75	RP
76	RELOP
77	Exp (45)
78	ID: NULL
79	RP
80	Stmt (46)
81	RETURN
82	Exp (46)
83	ID: NULL
84	SEMI
85	StmtList (47)
86	Stmt (47)
87	Exp (47)
88	Exp (47)
89	Exp (47)
90	ID: copy
91	DOT
92	ID: dup
93	ASSIGNOP
94	Exp (47)
95	Exp (47)
96	ID: orig
97	DOT
98	ID: dup
99	SEMI
100	StmtList (48)
101	Stmt (48)
102	Exp (48)
103	Exp (48)
104	Exp (48)
105	ID: copy

106	DOT
107	ID: free
108	ASSIGNOP
109	Exp (48)
110	Exp (48)
111	ID: orig
112	DOT
113	ID: free
114	SEMI
115	StmtList (49)
116	Stmt (49)
117	Exp (49)
118	Exp (49)
119	Exp (49)
120	ID: copy
121	DOT
122	ID: match
123	ASSIGNOP
124	Exp (49)
125	Exp (49)
126	ID: orig
127	DOT
128	ID: match
129	SEMI
130	StmtList (50)
131	Stmt (50)
132	Exp (50)
133	ID: listRewind
134	LP
135	Args (50)
136	Exp (50)
137	ID: orig

138	COMMA
139	Args (50)
140	Exp (50)
141	ID: ref
142	LP
143	Args (50)
144	Exp (50)
145	ID: iter
146	RP
147	RP
148	SEMI
149	StmtList (51)
150	Stmt (51)
151	WHILE
152	LP
153	Exp (51)
154	Exp (51)
155	LP
156	Exp (51)
157	Exp (51)
158	ID: node
159	ASSIGNOP
160	Exp (51)
161	ID: listNext
162	LP
163	Args (51)
164	Exp (51)
165	ID: ref
166	LP
167	Args (51)
168	Exp (51)
169	ID: iter

170	RP
171	RP
172	RP
173	RELOP
174	Exp (51)
175	ID: NULL
176	RP
177	Stmt (51)
178	CompSt (51)
179	LC
180	DefList (52)
181	Def (52)
182	Specifier (52)
183	StructSpecifier (52)
184	STRUCT
185	Tag (52)
186	ID: voidstart
187	DecList (52)
188	Dec (52)
189	VarDec (52)
190	ID: value
191	SEMI
192	StmtList (54)
193	Stmt (54)
194	IF
195	LP
196	Exp (54)
197	Exp (54)
198	ID: copy
199	DOT
200	ID: dup
201	RP

202	Stmt (54)
203	CompSt (54)
204	LC
205	StmtList (55)
206	Stmt (55)
207	Exp (55)
208	Exp (55)
209	ID: value
210	ASSIGNOP
211	Exp (55)
212	ID: dup
213	LP
214	Args (55)
215	Exp (55)
216	ID: copy
217	COMMA
218	Args (63)
219	Exp (63)
220	Exp (63)
221	ID: node
222	DOT
223	ID: value
224	RP
225	SEMI
226	StmtList (64)
227	Stmt (64)
228	IF
229	LP
230	Exp (64)
231	Exp (64)
232	ID: value
233	RELOP

234	Exp (64)
235	ID: NULL
236	RP
237	Stmt (64)
238	CompSt (64)
239	LC
240	StmtList (65)
241	Stmt (65)
242	Exp (65)
243	ID: listRelease
244	LP
245	Args (65)
246	Exp (65)
247	ID: copy
248	RP
249	SEMI
250	StmtList (66)
251	Stmt (66)
252	RETURN
253	Exp (66)
254	ID: NULL
255	SEMI
256	RC
257	RC
258	ELSE
259	Stmt (69)
260	Exp (69)
261	Exp (69)
262	ID: value
263	ASSIGNOP
264	Exp (69)
265	Exp (69)

266	ID: node
267	DOT
268	ID: value
269	SEMI
270	StmtList (70)
271	Stmt (70)
272	IF
273	LP
274	Exp (70)
275	Exp (70)
276	ID: listAddNodeTail
277	LP
278	Args (70)
279	Exp (70)
280	ID: copy
281	COMMA
282	Args (70)
283	Exp (70)
284	ID: value
285	RP
286	RELOP
287	Exp (70)
288	ID: NULL
289	RP
290	Stmt (70)
291	CompSt (70)
292	LC
293	StmtList (71)
294	Stmt (71)
295	Exp (71)
296	ID: listRelease
297	LP

298		Args (71)
299		Exp (71)
300		ID: copy
301		RP
302		SEMI
303		StmtList (72)
304		Stmt (72)
305		RETURN
306		Exp (72)
307		ID: NULL
308		SEMI
309		RC
310		RC
311		StmtList (75)
312		Stmt (75)
313		RETURN
314		Exp (75)
315		ID: copy
316		SEMI
317	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或者李聪助教联系，注意同时抄送给许老师。