编译原理第二次实验测试用例：目录

1 A 组测试用例

本组测试用例共 20 个，测试用例 1-17 分别对应语义错误 1-17，之后三个测试用例对应于语义错误 7，12，15。每个用例仅在其中一行含有语义错误。某些语义错误可能会产生连锁反应。测试用例 A-i 对应的"本质错误"的错误类型是必须报出来的，如果报出其他错误，只要是由本质错误连带引发的（包括但不限于下面明确给出的情况），我们都不会扣分。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 Point {
2      float _x;
3      float _y;
4  };
5
6  struct Point createPoint(float x, float y) {
7      struct Point p;
8      p._x = x;
9      p._y = y;
10     return p;
11 }
12
13 float getX(struct Point p1) {
14     return p1._x;
15 }
16
17 float getY(struct Point p2) {
18     return p2_y;
19 }
```

输出

```
1  Error type 1 at line 18 : undefined variable 'p2_y'
```

说明：第 18 行中，p2_y 这个变量没有定义过。这里可以多报一个 8 型错误。

1.2 A-2

输入

```
1  struct Point {
2      int    _mode;
3      float _x;
4      float _y;
5      float _r;
6      float _theta;
7  };
8
9  struct Point createCartPoint(float x, float y) {
10     struct Point cartPoint;
11     cartPoint._mode  = 0;
12     cartPoint._x     = x;
13     cartPoint._y     = y;
14     cartPoint._r     = 0.0;
15     cartPoint._theta = 0.0;
16     return cartPoint;
17 }
18
19 struct Point createPolarPoint(float r, float theta) {
20     struct Point polarPoint;
21     polarPoint._mode  = 1;
22     polarPoint._x     = 0.0;
23     polarPoint._y     = 0.0;
24     polarPoint._r     = r;
25     polarPoint._theta = theta;
26     return polarPoint;
27 }
28
```

```
29  float getX(struct Point p1) {
30      if (p1._mode == 0) {
31          return p1._x;
32      } else {
33          return p1._r * cos(p1._theta);
34      }
35  }
```

输出

```
1  Error type 2 at line 33: undefined function 'cos'
```

说明：第 33 行中，函数 cos 没有没有定义过。这里可以多报一个 7 型和 8 型错误。

1.3 A-3

输入

```
1   struct Stack {
2       int _capacity;
3       int _size;
4       int _content[100];
5   };
6
7   struct Stack initStack(int tryCapacity) {
8       struct Stack stk;
9       stk._capacity = 100;
10      stk._size    = 0;
11      if (tryCapacity > 0 && tryCapacity < 100) {
12          stk._capacity = tryCapacity;
13      }
14      return stk;
15  }
16
17  int size(struct Stack Stack) {
18      return Stack._size;
```

```
19  }
20
21  int empty(struct Stack stk2) {
22      return stk2._size == 0;
23  }
24
25  int top(struct Stack stk3) {
26      if (stk3._size == 0) {
27          return -1;
28      }
29      return stk3._content[stk3._size - 1];
30  }
```

输出

```
1  Error type 3 at line 17: duplicate variable definition 'Stack'
```

说明：第 17 行函数的形参的名称 Stack 和第 1 行的结构体的名称重复了。这里可以在 18 行多报一个 1 型和 13 型错误。

1.4 A-4

输入

```
1  struct Stack {
2      int _capacity;
3      int _size;
4      int _content[100];
5  };
6
7  struct Stack initStack(int tryCapacity) {
8      struct Stack stk;
9      stk._capacity = 100;
10     stk._size    = 0;
11     if (tryCapacity > 0 && tryCapacity < 100) {
12         stk._capacity = tryCapacity;
```

```
13        }
14        return stk;
15   }
16
17   int size(struct Stack stk1) {
18        return stk1._size;
19   }
20
21   int empty(struct Stack stk2) {
22        return stk2._size == 0;
23   }
24
25   int top(struct Stack stk3) {
26        if (stk3._size == 0) {
27            return -1;
28        }
29        return stk3._content[stk3._size - 1];
30   }
31
32   int top(struct Stack stk4) {
33        if (stk4._size == 0) {
34            return -1;
35        }
36        return stk4._content[stk4._size - 1];
37   }
```

输出

```
1   Error type 4 at line 32: duplicate function definition 'top'
```

说明：第 32 行定义的函数 top 和第 25 行定义的函数重名了。错误也可以报在 25 行。

1.5 A-5

输入

```c
struct Stack {
    int _capacity;
    int _size;
    int _content[100];
};

struct Stack initStack(int tryCapacity) {
    struct Stack stk;
    stk._capacity = 100.0;
    stk._size    = 0;
    if (tryCapacity > 0 && tryCapacity < 100) {
        stk._capacity = tryCapacity;
    }
    return stk;
}

int size(struct Stack stk1) {
    return stk1._size;
}

int empty(struct Stack stk2) {
    return stk2._size == 0;
}

int top(struct Stack stk3) {
    if (stk3._size == 0) {
        return -1;
    }
    return stk3._content[stk3._size - 1];
}
```

输出

```
Error type 5 at line 9: type mismatch for assignment statement
```

说明：第 9 行中，赋值表达式两边的变量类型不一致，不能把一个浮点数赋值给一个整型变量。

1.6 A-6

输入

```
struct Stack {
    int _capacity;
    int _size;
    int _content[100];
};

struct Stack initStack(int tryCapacity) {
    struct Stack stk;
    stk._capacity = 100;
    stk._size    = 0;
    if (tryCapacity > 0 && tryCapacity < 100) {
        stk._capacity = tryCapacity;
    }
    return stk;
}

int size(struct Stack stk1) {
    return stk1._size;
}

int empty(struct Stack stk2) {
    return stk2._size == 0;
}

int top(struct Stack stk3) {
    if (stk3._size == 0) {
```

```
27          return -1;
28      }
29      return stk3._content[stk3._size - 1];
30  }
31
32  struct Stack pop(struct Stack stk4) {
33      if (empty(stk4) = 1) {
34          return stk4;
35      }
36      stk4._size = stk4._size - 1;
37      return stk4;
38  }
```

输出

```
1  Error type 6 at line 33: cannot assign to a rvalue
```

说明：第 33 行中，函数的返回值是右值，不能放在赋值表达式的左边。

1.7 A-7

输入

```
1  struct Stack {
2      int _capacity;
3      int _size;
4      int _content[100];
5  };
6
7  struct Stack initStack(int tryCapacity) {
8      struct Stack stk;
9      stk._capacity = 100;
10     stk._size    = 0;
11     if (tryCapacity > 0 && tryCapacity < 100) {
12         stk._capacity = tryCapacity;
13     }
```

```
14      return stk;
15  }
16
17  int size(struct Stack stk1) {
18      return stk1._size;
19  }
20
21  int empty(struct Stack stk2) {
22      return stk2._size == 0;
23  }
24
25  int top(struct Stack stk3) {
26      if (stk3._size == 0) {
27          return -1;
28      }
29      return stk3._content[stk3._size - 1];
30  }
31
32  struct Stack push(struct Stack stk5, int value) {
33      if (stk5._size < stk5._capacity) {
34          stk5._content[stk5._size] = value;
35          stk5._size = stk5 + 1;
36      }
37      return stk5;
38  }
```

输出

```
1  Error type 7 at line 35: cannot add a struct type and an integer
```

说明：第 35 行中，不能把一个结构体和一个整数相加。这里可以多报一个 5 型错误。

1.8 A-8

输入

11

```c
struct Stack {
    int _capacity;
    int _size;
    int _content[100];
};

struct Stack initStack(int tryCapacity) {
    struct Stack stk;
    stk._capacity = 100;
    stk._size    = 0;
    if (tryCapacity > 0 && tryCapacity < 100) {
        stk._capacity = tryCapacity;
    }
    return stk;
}

int size(struct Stack stk1) {
    return stk1._size;
}

int empty(struct Stack stk2) {
    return stk2._size == 0;
}

int top(struct Stack stk3) {
    if (stk3._size == 0) {
        return -1;
    }
    return stk3;
}
```

输出

```
Error type 8 at line 29: type mismatch for return
```

说明：第 29 行中，实际的返回值类型 struct Stack 和声明的返回值类型 int 不一致。

1.9　A-9

输入

```
struct CheckData {
    int _base;
    int _data;
};

struct CheckData createCheckData(int base, int data) {
    struct CheckData cd1;
    cd1._base = base;
    cd1._data = data;
    return cd1;
}

int mod(int p, int q) {
    return p - p / q;
}

int isGeneralPalindrome(struct CheckData cd) {
    if (cd._data < 0 || cd._base < 2) {
        return 0;
    }
    if (cd._data < cd._base) {
        return 1;
    }
    {
        int tail = mod(cd._data, cd._base);
        int head = cd._data;
        int len  = 0;
```

```
28        while (head >= cd._base) {
29            head = head / cd._base;
30            len = len + 1;
31        }
32        if (head == tail) {
33            while (len > 0) {
34                head = head * cd._base;
35                len = len - 1;
36            }
37            cd._data = cd._data - head - tail;
38            return isGeneralPalindrome(cd);
39        } else {
40            return 0;
41        }
42    }
43 }
44
45 int main() {
46     int b = 10;
47     int d = 100;
48     isGeneralPalindrome(createCheckData(b, d));
49     isGeneralPalindrome(b, d);
50 }
```

输出

```
Error type 9 at line 49: function call arguments mismatch with
    function parameters
```

说明：第 49 行中，函数 isGeneralPalindrome 的实参数目和类型与形参不符。

1.10　A-10

输入

```
struct Point {
```

```
2      int   _mode;

3      float _x;

4      float _y;

5      float _r;

6      float _theta;

7  };

8

9  struct Point createCartPoint(float x, float y) {

10     struct Point cartPoint;

11     cartPoint._mode  = 0;

12     cartPoint._x     = x;

13     cartPoint._y     = y;

14     cartPoint._r     = 0.0;

15     cartPoint._theta = 0.0;

16     return cartPoint;

17 }

18

19 struct Point createPolarPoint(float r, float theta) {

20     struct Point polarPoint;

21     polarPoint._mode  = 1;

22     polarPoint._x     = 0.0;

23     polarPoint._y     = 0.0;

24     polarPoint._r     = r;

25     polarPoint._theta = theta;

26     return polarPoint;

27 }

28

29 int main() {

30     struct Point p, points[10];

31     int cnt = 0;

32     while (cnt < 10) {

33         points[cnt] = createPolarPoint(2.2, 3.5);
```

```
34        cnt = cnt + 1;
35      }
36    p = p[0];
37    return 0;
38 }
```

输出

```
1 Error type 10 at line 36: subscripted value is not an array
```

说明：第 36 行中，对非数组类型的变量 p 使用了数组索引符号 "[]"。这里可以多报一个 5 型错误。

1.11 A-11

输入

```
1  struct Point {
2      int    _mode;
3      float _x;
4      float _y;
5      float _r;
6      float _theta;
7  };
8
9  struct Point createCartPoint(float x, float y) {
10     struct Point cartPoint;
11     cartPoint._mode  = 0;
12     cartPoint._x     = x;
13     cartPoint._y     = y;
14     cartPoint._r     = 0.0;
15     cartPoint._theta = 0.0;
16     return cartPoint;
17 }
18
19 struct Point createPolarPoint(float r, float theta) {
```

```
20      struct Point polarPoint;

21      polarPoint._mode  = 1;

22      polarPoint._x     = 0.0;

23      polarPoint._y     = 0.0;

24      polarPoint._r     = r;

25      polarPoint._theta = theta;

26      return polarPoint;

27  }

28

29  float cos(float cosv) {

30      return -1.0;

31  }

32

33  float getX(struct Point p1) {

34      if (p1._mode == 0) {

35          return p1._x;

36      } else {

37          return p1._r * cos(p1._theta);

38      }

39  }

40

41  int main() {

42      struct Point ps[10];

43      int cnt = 0;

44      float px = 0.0;

45      while (cnt < 10) {

46          ps[cnt] = createPolarPoint(px, px);

47          px = getX(ps[cnt]);

48          px();

49          cnt = cnt + 1;

50      }

51      return 0;
```

```
52  }
```

输出

```
1  Error type 11 at line 48: called object type 'float' is not a
       function
```

说明：第 48 行中，对非函数类型的变量 px 使用了函数调用符号 "()"。

1.12　A-12

输入

```
1   struct Point {
2       int   _mode;
3       float _x;
4       float _y;
5       float _r;
6       float _theta;
7   };
8
9   struct Point createCartPoint(float x, float y) {
10      struct Point cartPoint;
11      cartPoint._mode  = 0;
12      cartPoint._x     = x;
13      cartPoint._y     = y;
14      cartPoint._r     = 0.0;
15      cartPoint._theta = 0.0;
16      return cartPoint;
17  }
18
19  struct Point createPolarPoint(float r, float theta) {
20      struct Point polarPoint;
21      polarPoint._mode  = 1;
22      polarPoint._x     = 0.0;
23      polarPoint._y     = 0.0;
```

```c
24       polarPoint._r     = r;
25       polarPoint._theta = theta;
26       return polarPoint;
27   }

28

29   float sqrt(float sqrtv) {
30       return -1.0;
31   }

32

33   float getX(struct Point px) {
34       return -1.0;
35   }

36

37   float getY(struct Point py) {
38       return -1.0;
39   }

40

41   float calDist(struct Point p1, struct Point p2) {
42       float p1x = getX(p1);
43       float p1y = getY(p1);
44       float p2x = getX(p2);
45       float p2y = getY(p2);
46       float xdist = p1x - p2x;
47       float ydist = p1y - p2y;
48       return sqrt(xdist * xdist + ydist * ydist);
49   }

50

51   int main() {
52       struct Point ps1[10];
53       struct Point ps2[10];
54       int cnt = 0;
55       float dist = 0.0;
```

```
56    while (cnt < 10) {

57        ps1[cnt] = createCartPoint(dist, dist);

58        ps2[cnt] = createPolarPoint(dist, dist);

59        dist = calDist(ps1[cnt], ps2[cnt]);

60        cnt = cnt + 1;

61    }

62    {

63        struct Point p = ps1[dist];

64    }

65    return 0;

66 }
```

输出

```
1 Error type 12 at line 63: array subscript is not an integer
```

说明：第 63 行中，不能使用 float 类型的变量 dist 作为数组的索引。

1.13   A-13

输入

```
1  struct Point {

2      int    _mode;

3      float _x;

4      float _y;

5      float _r;

6      float _theta;

7  };

8

9  struct Point createCartPoint(float x, float y) {

10     struct Point cartPoint;

11     cartPoint._mode  = 0;

12     cartPoint._x     = x;

13     cartPoint._y     = y;

14     cartPoint._r     = 0.0;
```

```c
15      cartPoint._theta = 0.0;

16      return cartPoint;

17  }

18

19  struct Point createPolarPoint(float r, float theta) {

20      struct Point polarPoint;

21      polarPoint._mode  = 1;

22      polarPoint._x     = 0.0;

23      polarPoint._y     = 0.0;

24      polarPoint._r     = r;

25      polarPoint._theta = theta;

26      return polarPoint;

27  }

28

29  float sqrt(float sqrtv) {

30      return -1.0;

31  }

32

33  float getX(struct Point px) {

34      return -1.0;

35  }

36

37  float getY(struct Point py) {

38      return -1.0;

39  }

40

41  float calDist(struct Point p1, struct Point p2) {

42      float p1x = getX(p1);

43      float p1y = getY(p1);

44      float p2x = getX(p2);

45      float p2y = getY(p2);

46      float xdist = p1x - p2x;
```

```
47    float ydist = p1y - p2y;

48    return sqrt(xdist * xdist + ydist * ydist);

49 }

50

51 int main() {

52    struct Point u;

53    struct Point v;

54    u = createCartPoint(1.0, 1.0);

55    v = createPolarPoint(0.0, 2.0);

56    {

57       float dist = calDist(u, v);

58       float dx = dist.x;

59    }

60    return 0;

61 }
```

输出

```
1 Error type 13 at line 58: member reference base type 'float' is not a
     structure
```

说明：第 58 行中，对 float 类型的变量 dist 使用了 "." 操作符。这里可以多报一个 5 型错误。

1.14 A-14

输入

```
1 struct PersonInfo {

2    int id;

3    int age;

4    int phoneNo;

5 };

6

7 int BLEN;

8 int CLEN;

9 struct AddressBook {
```

```
10        int volNo;
11        struct PersonInfo contacts[100];
12    } addrBooks[10];
13
14    int initAddrBooks() {
15        int bcnt = 0;
16        int ccnt = 0;
17        BLEN = 10;
18        CLEN = 100;
19        while (bcnt < BLEN) {
20            addrBooks[bcnt].volNo = bcnt;
21            ccnt = 0;
22            while (ccnt < CLEN) {
23                addrBooks[bcnt].contacts[ccnt].id = -1;
24                ccnt = ccnt + 1;
25            }
26            bcnt = bcnt + 1;
27        }
28        return 0;
29    }
30
31    int main() {
32        int h;
33        initAddrBooks();
34        h = addrBooks[0].contacts[0].hight;
35    }
```

输出

```
1   Error type 14 at line 34: no member named 'hight' in 'struct
        PersonInfo'
```

说明：第 34 行中，使用了未定义的域 hight。这里可以多报一个 5 型错误。

1.15 A-15

输入

```
struct PersonInfo {
    int id = -1;
    int age;
    int phoneNo;
};

int BLEN;
int CLEN;
struct AddressBook {
    int volNo;
    struct PersonInfo contacts[100];
} addrBooks[10];

int initAddrBooks() {
    int bcnt = 0;
    BLEN = 10;
    CLEN = 100;
    while (bcnt < BLEN) {
        addrBooks[bcnt].volNo = bcnt;
        bcnt = bcnt + 1;
    }
    return 0;
}

int main() {
    initAddrBooks();
}
```

输出

```
Error type 15 at line 2: cannot assign default value to structure
```

```
member at definition
```

说明：第 2 行中，结构体在定义时，不能对它的域设置初始值。

1.16   A-16

输入

```
1  struct PersonInfo {
2      int id;
3      int age;
4      int phoneNo;
5  };
6
7  int BLEN;
8  int CLEN;
9  struct AddressBook {
10     int volNo;
11     struct PersonInfo contacts[100];
12 } addrBooks[10];
13
14 int initAddrBooks() {
15     int bcnt = 0;
16     int ccnt = 0;
17     BLEN = 10;
18     CLEN = 100;
19     while (bcnt < BLEN) {
20         addrBooks[bcnt].volNo = bcnt;
21         ccnt = 0;
22         while (ccnt < CLEN) {
23             addrBooks[bcnt].contacts[ccnt].id = -1;
24             ccnt = ccnt + 1;
25         }
26         bcnt = bcnt + 1;
27     }
```

```
28    return 0;

29  }

30

31  int main() {

32      struct PersonInfo {

33          float hight;

34          float weight;

35      } ps;

36      initAddrBooks();

37  }
```

输出

```
1  Error type 16 at line 32: duplicate structure name 'PersonInfo'
```

说明：第 32 行中，定义的结构体 PersonInfo 和已经定义过的结构体重名了。

1.17 A-17

输入

```
1  int BLEN;

2  int CLEN;

3  struct AddressBook {

4      int volNo;

5      struct PersonInfo contacts[100];

6  } addrBooks[10];

7

8  int initAddrBooks() {

9      int bcnt = 0;

10     BLEN = 10;

11     CLEN = 100;

12     while (bcnt < BLEN) {

13         addrBooks[bcnt].volNo = bcnt;

14         bcnt = bcnt + 1;

15     }
```

```
16    return 0;

17  }

18

19  int main() {

20      initAddrBooks();

21  }
```

输出

```
1  Error type 17 at line 5: use undefined structure type 'PersonInfo'
```

说明：第 5 行中，使用了未定义的结构体类型 PersonInfo。

1.18 A-18

输入

```
1   struct S {

2       int   si;

3       float sf;

4       int   sia[10];

5       float sfa[10];

6   } sa[10];

7

8   int foo(int fi, float ff, struct S fs) {

9       return 0;

10  }

11

12  int main() {

13      foo(sa[0].si - sa[0].sia, sa[0].sf, sa[0]);

14  }
```

输出

```
1  Error type 7 at line 13: invalid operands to binary operator '-'
```

说明：第 13 行中，数组类型的变量不能作为减号的操作数。这里可以多报一个 9 型错误。

27

1.19 A-19

输入

```
1  struct S {
2      int   si;
3      float sf;
4      int   sia[10];
5      float sfa[10];
6  } sa[10];
7
8  int foo(int fi, float ff, struct S fs) {
9      return 0;
10 }
11
12 int main() {
13     foo(sa[sa[0].sf].si, sa[0].sf, sa[0]);
14 }
```

输出

```
1  Error type 12 at line 13: array subscript is not an integer
```

说明：第 13 行中，不能使用 float 类型的变量作为数组的索引。这里可以多报一个 9 型错误。

1.20 A-20

输入

```
1  struct S {
2      int   si;
3      float sf;
4      int   sia[10];
5      float sia[10];
6  } sa[10];
7
8  int foo(int fi, float ff, struct S fs) {
```

```
 9        return 0;
10    }
11
12    int main() {
13        foo(sa[0].si, sa[0].sf, sa[0]);
14    }
```

输出

```
1    Error type 15 at line 5: duplicate member 'sia'
```

说明：第 5 行中，结构体的域的名字和之前的域的名称重复。错误也可以报在第 4 行。

2 B 组测试用例

本组测试用例共 2 个，其中包含多个语义错误。每一行的语义错误会分别算分，同一个语义错误可能会有连锁反应，其处理方式与 A 类用例相同，只要是合理的（包括但不限于下面明确给出的情况），都不会影响得分。

2.1 B-1

输入

```
 1    struct {
 2        int _stk[100];
 3        int _top = -1;
 4    } stkVM;
 5    int STK_MDEP;
 6    int errno;
 7
 8    int init() {
 9        STK_MDEP   = 100.0;
10        errno      = 0;
11        stkVM._top = -1;
12        return stkVM;
13    }
```

29

```
14
15  int pop() {
16      if (errno == -1) {
17          return -1;
18      }
19      if (stkVM._top == -1) {
20          errno = -1;
21          return -1;
22      } else {
23          int ret = stkVM._stk[stkVM._top];
24          stkVM._top = stkVM._top - 1;
25          return ret;
26      }
27  }
28
29  int push(int i) {
30      if (errno == -1) {
31          return -1;
32      }
33      if (stkVM._top == STK_MDEP - 1) {
34          errno = -1;
35          return -1;
36      } else {
37          stkVM._top = stkVM._top + 1;
38          stkVM._stk[stkVM._top] = i;
39          return 0;
40      }
41  }
42
43  int add(int a1, int a2) {
44      if (errno == -1) {
45          return -1;
```

```
46        }
47        a1 = pop();
48        a2 = pop();
49        push(a1 + a2);
50        return 0;
51    }
52
53    int sub(int s1, int s2) {
54        if (errno == -1) {
55            return -1;
56        }
57        s1 = pop();
58        s2 = pop();
59        push(s1 - s1);
60        return 0;
61    }
62
63    int mul(int m1, int m2) {
64        if (errno == -1) {
65            return -1;
66        }
67        m1 = pop();
68        m2 = pop();
69        push(m1 * m2);
70        return 0;
71    }
72
73    int div(int d1, int d2) {
74        if (errno == -1) {
75            return -1;
76        }
77        d1 = pop();
```

```
78      d2 = pop();
79      if (d2 == 0) {
80          errno = -1;
81          return -1;
82      } else {
83          push(d1 / d2);
84          return 0;
85      }
86  }
87
88  int main() {
89      int res = -1;
90      init();
91      add(add(push(2), push(3)));
92      if (errno == 0) {
93          res = pop();
94          res = res();
95      }
96      return res;
97  }
```

输出

```
1  Error type 15 at line 3: cannot assign default value to structure
      member at definition
2  Error type 5 at line 9: type mismatch for assignment statement
3  Error type 8 at line 12: type mismatch for return
4  Error type 9 at line 91: function call arguments mismatch with
      function parameters
5  Error type 11 at line 94: called object type 'int' is not a function
```

说明：第 3 行中，不能在定义结构体时对域 _top 进行初始化；第 9 行中，不能将浮点数赋值给整型变量 STK_MDEP；第 12 行中，函数实际的返回值 stkVM 类型和声明的返回值类型 int 不符；第 91 行中，函数 add 的实参数目与形参不符；第 94 行中，不能对 int 类型使用函数调用

操作符，这里可以多报一个 5 型错误。多报了没有'_top' 这个域的 14 型错误也算对。

2.2 B-2

输入

```
1  struct Vector {
2      int x;
3      int y;
4      int z;
5  };
6
7  int equal(struct Vector v1e, struct Vector v2e) {
8      if (v1e.x == v2e._x && v1e.y == v2e.y && v1e.z == v2e.z) {
9          return 1;
10     } else {
11         return 0;
12     }
13 }
14
15 struct Vector add(struct Vector v1a, struct Vector v2a) {
16     struct Vector va;
17     va.x = v1a + v2a.x;
18     va.y = v1a.y + v2a.y;
19     va.z = v1a.z + v2a.z;
20     return va;
21 }
22
23 struct Vector sub(struct Vector v1s, struct Vector v2s) {
24     struct Vector vs;
25     vs.x = v1s.x - v2s.x;
26     vs.y = v1s.y - v2s.y;
27     vs.z = v1s.z - v2s.z;
28     return vs;
```

```
29   }

30

31   struct Particle {
32       struct Mass mass;
33       struct Vector position;
34       struct Vector velocity;
35   };

36

37   int willCollide(struct Particle particles[10], int time) {
38       while (time > 0) {
39           int cnt = 0;
40           int i = 0;
41           int j = 0;
42           while (cnt < 10) {
43               particles[cnt].position = add(particles[cnt].position,
                     particles[cnt].velocity);
44               cnt = cnt + 1;
45           }
46           while (i < 10) {
47               while (j < 10) {
48                   if (equal(particles[i].position, particles[j].
                         position)) {
49                       return 1;
50                   }
51                   j = j + 1;
52               }
53               i = i + 1;
54           }
55           time = time - 1;
56       }
57       return 0;
58   }
```

```
59
60  int main() {
61      struct Particle ps[10];
62      willCollide(ps, 100);
63  }
```

输出

```
1  Error type 14 at line 8: no member named '_x' in 'struct Vector'
2  Error type 7 at line 17: cannot add a struct type and an integer
3  Error type 17 at line 32: use undefined structure type 'Mass'
```

说明：第 8 行中，使用了未定义的域 _x，这里可以多报一个 7 型错误；第 17 行中，不能把一个结构体和一个整数相加，这里可以多报一个 5 型错误；第 32 行中，使用了未定义的结构体类型 Mass。

3 C 组测试用例

本组测试用例共 2 个，不包含任何错误。

3.1 C-1

输入

```
1   struct {
2       int _stk[100];
3       int _top;
4   } stkVM;
5   int STK_MDEP;
6   int errno;
7
8   int init() {
9       STK_MDEP   = 100;
10      errno      = 0;
11      stkVM._top = -1;
12      return 0;
```

```c
13    }

15    int pop() {
16        if (errno == -1) {
17            return -1;
18        }
19        if (stkVM._top == -1) {
20            errno = -1;
21            return -1;
22        } else {
23            int ret = stkVM._stk[stkVM._top];
24            stkVM._top = stkVM._top - 1;
25            return ret;
26        }
27    }

29    int push(int i) {
30        if (errno == -1) {
31            return -1;
32        }
33        if (stkVM._top == STK_MDEP - 1) {
34            errno = -1;
35            return -1;
36        } else {
37            stkVM._top = stkVM._top + 1;
38            stkVM._stk[stkVM._top] = i;
39            return 0;
40        }
41    }

43    int add(int a1, int a2) {
44        if (errno == -1) {
```

```c
        return -1;
    }
    a1 = pop();
    a2 = pop();
    push(a1 + a2);
    return 0;
}

int sub(int s1, int s2) {
    if (errno == -1) {
        return -1;
    }
    s1 = pop();
    s2 = pop();
    push(s1 - s1);
    return 0;
}

int mul(int m1, int m2) {
    if (errno == -1) {
        return -1;
    }
    m1 = pop();
    m2 = pop();
    push(m1 * m2);
    return 0;
}

int div(int d1, int d2) {
    if (errno == -1) {
        return -1;
    }
```

```
77    d1 = pop();
78    d2 = pop();
79    if (d2 == 0) {
80        errno = -1;
81        return -1;
82    } else {
83        push(d1 / d2);
84        return 0;
85    }
86 }
87
88 int main() {
89    int res = -1;
90    init();
91    add(push(1), add(push(2), push(3)));
92    if (errno == 0) {
93        res = pop();
94    }
95    return res;
96 }
```

输出

```
1  // 正常返回，没有任何输出
```

3.2   C-2

输入

```
1  struct Vector {
2      int x;
3      int y;
4      int z;
5  };
6
```

```c
int equal(struct Vector v1e, struct Vector v2e) {
    if (v1e.x == v2e.x && v1e.y == v2e.y && v1e.z == v2e.z) {
        return 1;
    } else {
        return 0;
    }
}


struct Vector add(struct Vector v1a, struct Vector v2a) {
    struct Vector va;
    va.x = v1a.x + v2a.x;
    va.y = v1a.y + v2a.y;
    va.z = v1a.z + v2a.z;
    return va;
}


struct Vector sub(struct Vector v1s, struct Vector v2s) {
    struct Vector vs;
    vs.x = v1s.x - v2s.x;
    vs.y = v1s.y - v2s.y;
    vs.z = v1s.z - v2s.z;
    return vs;
}


struct Particle {
    int mass;
    struct Vector position;
    struct Vector velocity;
};


int willCollide(struct Particle particles[10], int time) {
    while (time > 0) {
```

```
39        int cnt = 0;

40        int i = 0;

41        int j = 0;

42        while (cnt < 10) {

43            particles[cnt].position = add(particles[cnt].position,
                  particles[cnt].velocity);

44            cnt = cnt + 1;

45        }

46        while (i < 10) {

47            while (j < 10) {

48                if (equal(particles[i].position, particles[j].
                      position)) {

49                    return 1;

50                }

51                j = j + 1;

52            }

53            i = i + 1;

54        }

55        time = time - 1;

56    }

57    return 0;

58 }

59

60 int main() {

61    struct Particle ps[10];

62    willCollide(ps, 100);

63 }
```

输出

```
1 // 正常返回，没有任何输出
```

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。需要能够识别其语言特性，如果提示错误则不得分；其他分组的同学需要识别出其中的错误，如果没有报错，则将视为违规，将会倒扣分。

输入

```c
int O_ADD;
int O_PRD;
int O_SUB;
int O_DIV;
struct Operation {
    int oType;
    int opt;
};

int T_INT;
int T_FLT;
struct BinData {
    int   bdType;
    int   bdIData[2];
    float bdFData[2];
};

int MLEN;
struct MulData {
    int   mdType;
    int   mdIData[100];
    float mdFData[100];
};

struct Result {
```

```
26      int    rType;
27      int    valid;
28      int    iRes;
29      float fRes;
30  };

31

32  int initArith() {
33      O_ADD = 0;
34      O_PRD = 1;
35      O_SUB = 2;
36      O_DIV = 3;
37      T_INT = 4;
38      T_FLT = 5;
39      MLEN  = 100;
40      return 0;
41  }

42

43  struct Result binOperator(struct Operation bOperation, struct BinData
        binData);

44

45  struct Result binOperator(struct Operation bOperation, struct BinData
        binData) {
46      struct Result bOpRes;
47      bOpRes.valid = 1;
48      if (bOperation.opt == O_ADD) {
49          bOpRes.iRes = binData.bdIData[0] + binData.bdIData[1];
50          bOpRes.fRes = binData.bdFData[0] + binData.bdFData[1];
51      } else if (bOperation.opt == O_PRD) {
52          bOpRes.iRes = binData.bdIData[0] * binData.bdIData[1];
53          bOpRes.fRes = binData.bdFData[0] * binData.bdFData[1];
54      } else if (bOperation.opt == O_SUB) {
55          bOpRes.iRes = binData.bdIData[0] - binData.bdIData[1];
```

```
56          bOpRes.fRes = binData.bdFData[0] - binData.bdFData[1];
57      } else if (bOperation.opt == O_DIV) {
58          bOpRes.iRes = binData.bdIData[0] / binData.bdIData[1];
59          bOpRes.fRes = binData.bdFData[0] / binData.bdFData[1];
60      } else {
61          bOpRes.valid = 0;
62      }
63      bOpRes.valid = bOpRes.valid && (bOperation.oType == binData.
            bdType);
64      bOpRes.rType = bOperation.oType;
65      return bOpRes;
66  }
67
68  struct Result mulOperation(struct Operation mOperation, struct
        MulData mulData) {
69      struct Result mOpRes;
70      int cnt = 0;
71      mOpRes.valid = 1;
72      if (mOperation.opt == O_ADD) {
73          mOpRes.iRes = 0;
74          mOpRes.fRes = 0.0;
75      } else if (mOperation.opt == O_PRD) {
76          mOpRes.iRes = 1;
77          mOpRes.fRes = 1.0;
78      } else {
79          mOpRes.valid = 0;
80      }
81      while (cnt < MLEN) {
82          if (mOperation.opt == O_ADD) {
83              mOpRes.iRes = mOpRes.iRes + mulData.mdIData[cnt];
84              mOpRes.fRes = mOpRes.fRes + mulData.mdFData[cnt];
85          } else if (mOperation.opt == O_PRD) {
```

```
86        mOpRes.iRes = mOpRes.iRes * mulData.mdIData[cnt];
87        mOpRes.fRes = mOpRes.fRes * mulData.mdFData[cnt];
88      }
89      cnt = cnt + 1;
90    }
91    mOpRes.valid = mOpRes.valid && (mOperation.oType == mulData.
         mdType);
92    mOpRes.rType = mOperation.oType;
93    return mOpRes;
94 }
95
96 struct Result mulOperation(struct Operation mOperation, struct
      MulData mulData);
97 struct Result binOperator(struct Operation bOperation, struct BinData
       binData);
```

输出

```
1 // 正常返回，没有任何输出。
```

说明：2.1 分组的同学没有任何输出，其他同学在第 43，96，97 行报语法错误。

4.2 D-2

输入

```
1  int O_ADD;
2  int O_PRD;
3  int O_SUB;
4  int O_DIV;
5  struct Operation {
6      int oType;
7      int opt;
8  };
9
10 int T_INT;
```

```c
11   int T_FLT;
12   struct BinData {
13       int   bdType;
14       int   bdIData[2];
15       float bdFData[2];
16   };
17
18   int MLEN;
19   struct MulData {
20       int   mdType;
21       int   mdIData[100];
22       float mdFData[100];
23   };
24
25   struct Result {
26       int   rType;
27       int   valid;
28       int   iRes;
29       float fRes;
30   };
31
32   int initArith() {
33       O_ADD = 0;
34       O_PRD = 1;
35       O_SUB = 2;
36       O_DIV = 3;
37       T_INT = 4;
38       T_FLT = 5;
39       MLEN  = 100;
40       return 0;
41   }
42
```

```
43  int cnt;

44

45  struct Result binOperator(struct Operation operation, struct BinData
        binData) {
46      struct Result result;
47      result.valid = 1;
48      if (operation.opt == O_ADD) {
49          result.iRes = binData.bdIData[0] + binData.bdIData[1];
50          result.fRes = binData.bdFData[0] + binData.bdFData[1];
51      } else if (operation.opt == O_PRD) {
52          result.iRes = binData.bdIData[0] * binData.bdIData[1];
53          result.fRes = binData.bdFData[0] * binData.bdFData[1];
54      } else if (operation.opt == O_SUB) {
55          result.iRes = binData.bdIData[0] - binData.bdIData[1];
56          result.fRes = binData.bdFData[0] - binData.bdFData[1];
57      } else if (operation.opt == O_DIV) {
58          result.iRes = binData.bdIData[0] / binData.bdIData[1];
59          result.fRes = binData.bdFData[0] / binData.bdFData[1];
60      } else {
61          result.valid = 0;
62      }
63      result.valid = result.valid && (operation.oType == binData.bdType
            );
64      result.rType = operation.oType;
65      return result;
66  }

67

68  struct Result mulOperation(struct Operation operation, struct MulData
        mulData) {
69      struct Result result;
70      int cnt = 0;
71      result.valid = 1;
```

```
72    if (operation.opt == O_ADD) {
73        result.iRes = 0;
74        result.fRes = 0.0;
75    } else if (operation.opt == O_PRD) {
76        result.iRes = 1;
77        result.fRes = 1.0;
78    } else {
79        result.valid = 0;
80    }
81    while (cnt < MLEN) {
82        if (operation.opt == O_ADD) {
83            result.iRes = result.iRes + mulData.mdIData[cnt];
84            result.fRes = result.fRes + mulData.mdFData[cnt];
85        } else if (operation.opt == O_PRD) {
86            result.iRes = result.iRes * mulData.mdIData[cnt];
87            result.fRes = result.fRes * mulData.mdFData[cnt];
88        }
89        cnt = cnt + 1;
90    }
91    result.valid = result.valid && (operation.oType == mulData.mdType
          );
92    result.rType = operation.oType;
93    return result;
94 }
```

输出

```
1  // 正 常 返 回 ， 没 有 任 何 输 出 。
```

说明：2.2 分组的同学没有任何输出。其他同学应该识别出对于变量 operation，result，cnt 的重复定义。这里可以放松对于行号的要求，只要报出它们的重复定义就行。

4.3　D-3

输入

```c
struct S {
    int   si;
    float sf;
    int   sia[10];
    float sfa[10];
    struct {
        float ssfa[10];
        int   ssia[10];
        float ssf;
        int   ssi;
    } ssa[10];
};

struct T {
    int   ti;
    float tf;
    int   tia[10];
    float tfa[10];
    struct {
        float ttfa[10];
        int   ttia[10];
        float ttf;
        int   tti;
    } tta[10];
};

int equal(struct S s1, struct S s2) {
    int i = 0;
    int j = 0;
    if (s1.si != s2.si || s1.sf != s2.sf) {
        return 0;
    }
```

```
33    while (i < 10) {
34        if (s1.sia[i] != s2.sia[i] || s1.sfa[i] != s2.sfa[i]
35                || s1.ssa[i].ssi != s2.ssa[i].ssi || s1.ssa[i].ssf !=
                    s2.ssa[i].ssf) {
36            return 0;
37        }
38        j = 0;
39        while (j < 10) {
40            if (s1.ssa[i].ssfa[j] != s2.ssa[i].ssfa[j] || s1.ssa[i].
                ssia[j] != s2.ssa[i].ssia[j]) {
41                return 0;
42            }
43            j = j + 1;
44        }
45        i = i + 1;
46    }
47    return 1;
48 }
49
50 struct S copy(struct S s, struct T t) {
51    if (equal(s, t) == 1) {
52        return s;
53    } else {
54        return t;
55    }
56 }
```

输出

```
1  // 正常返回，没有任何输出
```

说明：2.3 分组的同学没有任何输出。其他同学应该在 51 行和 54 行分别报出 9 型和 8 型错误。

5 E 组测试用例

本组测试用例共 3 个，针对不同分组进行测试。

5.1 E2.1

这组测试用例针对 2.1 分组的同学。

输入

```
1  int O_ADD;
2  int O_PRD;
3  int O_SUB;
4  int O_DIV;
5  struct Operation {
6      int oType;
7      int opt;
8  };
9
10 int T_INT;
11 int T_FLT;
12 struct BinData {
13     int   bdType;
14     int   bdIData[2];
15     float bdFData[2];
16 };
17
18 int MLEN;
19 struct MulData {
20     int   mdType;
21     int   mdIData[100];
22     float mdFData[100];
23 };
24
25 struct Result {
```

```
26     int    rType;
27     int    valid;
28     int    iRes;
29     float fRes;
30 };
31
32 int initArith() {
33     O_ADD = 0;
34     O_PRD = 1;
35     O_SUB = 2;
36     O_DIV = 3;
37     T_INT = 4;
38     T_FLT = 5;
39     MLEN  = 100;
40     return 0;
41 }
42
43 struct Result binOperator(int bOperation, int binData[2]);
44
45 struct Result binOperator(struct Operation bOperation, struct BinData
       binData) {
46     struct Result bOpRes;
47     bOpRes.valid = 1;
48     if (bOperation.opt == O_ADD) {
49         bOpRes.iRes = binData.bdIData[0] + binData.bdIData[1];
50         bOpRes.fRes = binData.bdFData[0] + binData.bdFData[1];
51     } else if (bOperation.opt == O_PRD) {
52         bOpRes.iRes = binData.bdIData[0] * binData.bdIData[1];
53         bOpRes.fRes = binData.bdFData[0] * binData.bdFData[1];
54     } else if (bOperation.opt == O_SUB) {
55         bOpRes.iRes = binData.bdIData[0] - binData.bdIData[1];
56         bOpRes.fRes = binData.bdFData[0] - binData.bdFData[1];
```

```
57    } else if (bOperation.opt == O_DIV) {
58        bOpRes.iRes = binData.bdIData[0] / binData.bdIData[1];
59        bOpRes.fRes = binData.bdFData[0] / binData.bdFData[1];
60    } else {
61        bOpRes.valid = 0;
62    }
63    bOpRes.valid = bOpRes.valid && (bOperation.oType == binData.
          bdType);
64    bOpRes.rType = bOperation.oType;
65    return bOpRes;
66 }
67
68 struct Result mulOperation(struct Operation mOperation, struct
       MulData mulData);
69
70 int equal(struct MulData mulData1, struct MulData mulData2);
71
72 int main() {
73     struct Operation opn;
74     struct MulData data1;
75     struct MulData data2;
76     mulOperation(opn, data1);
77     mulOperation(opn, data2);
78     return equal(data1, data2);
79 }
```

输出

```
1 Error type 18 at line 43: undefined function 'binOperator'
2 Error type 19 at line 45: conflicting types for 'binOperator'
3 Error type 18 at line 68: undefined function 'mulOperation'
4 Error type 18 at line 70: undefined function 'equal'
```

说明：仅 2.1 分组的同学需要测试这个用例，并且报出以上错误。其中与函数 binOperator

相关的 19 型错误也可以报在第 43 行，那么与之相关的 18 型错误就不用报了。

5.2 E2.2

这组测试用例针对 2.2 分组的同学。

输入

```
1  int O_ADD;
2  int O_PRD;
3  int O_SUB;
4  int O_DIV;
5  struct Operation {
6      int oType;
7      int opt;
8  };
9
10 int T_INT;
11 int T_FLT;
12 struct BinData {
13     int   bdType;
14     int   bdIData[2];
15     float bdFData[2];
16 };
17
18 int MLEN;
19 struct MulData {
20     int   mdType;
21     int   mdIData[100];
22     float mdFData[100];
23 };
24
25 struct Result {
26     int   rType;
27     int   valid;
```

```
28      int    iRes;

29      float  fRes;

30   };

31

32   int initArith() {

33       O_ADD = 0;

34       O_PRD = 1;

35       O_SUB = 2;

36       O_DIV = 3;

37       T_INT = 4;

38       T_FLT = 5;

39       MLEN  = 100;

40       return 0;

41   }

42

43   int cnt;

44

45   struct Result binOperator(struct Operation operation, struct BinData
        binData) {

46       struct Result result;

47       result.valid = 1;

48       if (operation.opt == O_ADD) {

49           result.iRes = binData.bdIData[0] + binData.bdIData[1];

50           result.fRes = binData.bdFData[0] + binData.bdFData[1];

51       } else if (operation.opt == O_PRD) {

52           result.iRes = binData.bdIData[0] * binData.bdIData[1];

53           result.fRes = binData.bdFData[0] * binData.bdFData[1];

54       } else if (operation.opt == O_SUB) {

55           result.iRes = binData.bdIData[0] - binData.bdIData[1];

56           result.fRes = binData.bdFData[0] - binData.bdFData[1];

57       } else if (operation.opt == O_DIV) {

58           result.iRes = binData.bdIData[0] / binData.bdIData[1];
```

```c
59        result.fRes = binData.bdFData[0] / binData.bdFData[1];
60    } else {
61        result.valid = 0;
62    }
63    result.valid = result.valid && (operation.oType == binData.bdType
        );
64    result.rType = operation.oType;
65    return result;
66 }
67
68 struct Result mulOperation(struct Operation operation, struct MulData
       mulData) {
69    struct Result result;
70    int cnt = 0;
71    result.valid = 1;
72    if (operation.opt == O_ADD) {
73        result.iRes = 0;
74        result.fRes = 0.0;
75    } else if (operation.opt == O_PRD) {
76        result.iRes = 1;
77        result.fRes = 1.0;
78    } else {
79        result.valid = 0;
80    }
81    while (cnt < MLEN) {
82        if (operation.opt == O_ADD) {
83            result.iRes = result.iRes + mulData.mdIData[cnt];
84            result.fRes = result.fRes + mulData.mdFData[cnt];
85        } else if (operation.opt == O_PRD) {
86            result.iRes = result.iRes * mulData.mdIData[cnt];
87            result.fRes = result.fRes * mulData.mdFData[cnt];
88        }
```

```c
89          cnt = cnt + 1;
90      }
91      result.valid = result.valid && (operation.oType == mulData.mdType
            );
92      result.rType = operation.oType;
93      return result;
94  }
95
96  int dummy;
97
98  int main() {
99      struct Operation operation;
100     struct MulData mulData;
101     float dummy = 0.0;
102     int cnt = 0;
103     {
104         int dummy[10];
105         dummy[0] = dummy[0] + 1;
106     }
107     {
108         int i = 0;
109         int i = 0;
110         operation.oType = T_INT;
111         operation.opt   = O_ADD;
112         mulData.mdType   = T_INT;
113         while (i < MLEN) {
114             mulData.mdIData[i] = i + 1;
115             i = i + 1;
116         }
117         mulOperation(operation, mulData);
118     }
119     return i;
```

```
120  }
```

输出

```
1  Error type 3 at line 109: duplicate variable definition 'i'
2  Error type 1 at line 119: undefined variable 'i'
```

说明：仅 2.2 分组的同学需要测试这个用例，并且报出以上错误。这里的 3 型错误也可以报在 108 行。

5.3　E2.3

这组测试用例针对 2.3 分组的同学。

输入

```
1   struct S {
2       int   si;
3       float sf;
4       int   sia[10];
5       float sfa[10];
6       struct {
7           float ssfa[10];
8           int   ssia[10];
9           float ssf;
10          int   ssi;
11      } ssa[10];
12  };
13
14  struct T {
15      int   ti;
16      float tf;
17      int   tia[10];
18      float tfa[10];
19      struct {
20          float ttfa[10];
21          int   ttia[10];
```

```
22          float ttf;

23          int    tti;

24      } tta[10];

25  };

26

27  int equal(struct S s1, struct S s2) {

28      int i = 0;

29      int j = 0;

30      if (s1.si != s2.si || s1.sf != s2.sf) {

31          return 0;

32      }

33      while (i < 10) {

34          if (s1.sia[i] != s2.sia[i] || s1.sfa[i] != s2.sfa[i]

35                  || s1.ssa[i].ssi != s2.ssa[i].ssi || s1.ssa[i].ssf !=

36                      s2.ssa[i].ssf) {

36              return 0;

37          }

38          j = 0;

39          while (j < 10) {

40              if (s1.ssa[i].ssfa[j] != s2.ssa[i].ssfa[j] || s1.ssa[i].

41                  ssia[j] != s2.ssa[i].ssia[j]) {

41                  return 0;

42              }

43              j = j + 1;

44          }

45          i = i + 1;

46      }

47      return 1;

48  }

49

50  struct S copy(struct S s, struct T t) {

51      if (equal(s, t) == 1) {
```

```
52        return s;
53    } else {
54        return t;
55    }
56 }
57
58 int main () {
59    struct {
60        int   tsi;
61        float tsf;
62        int   tsia[10];
63        float tsfa[10];
64        struct {
65            float tssfa[10];
66            int   tssia[10];
67            float tssf;
68            int   tssi;
69        } tss;
70    } fakeS;
71    struct S tmpS;
72    struct T tmpT;
73    copy(fakeS, tmpT);
74    copy(tmpT, tmpS);
75 }
```

输出

```
1 Error type 9 at line 73: function call arguments mismatch with
    function parameters
```

说明：仅 2.3 分组的同学需要测试这个用例，并且报出以上错误。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与李聪助教或屈道涵助教联系，注意同时抄送给许老师。