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 3，9，15。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 Hammer {
2    int h_id;
3    int h_no;
4    float h_weight;
5  };
6
7  int main() {
8    struct Hammer h;
9    h.h_id = 0;
10   h.h_no = 2;
11   h.h_weight = 2.2;
12   return _i;
13 }
```

输出

```
1  Error type 1 at Line 12: Use undefined variable
```

说明：第 12 行中，_i 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 Spoon {
2    int s_id;
```

```
  int s_no;
  int s_weight;
};

int mul(struct Spoon m_s) {
  return m_s.s_no * m_s.s_weight;
}

int main() {
  struct Spoon s;
  s.s_id = 3;
  s.s_no = 10;
  add(s.s_id, s._no);
  return 0;
}
```

输出

```
Error type 2 at Line 15: Undefined function 'add'
```

说明：第 15 行中，函数 add 没有没有定义过。

1.3 A-3

输入

```
struct Hammer {
  int h_id;
  int h_no;
  float h_weight;
};

int is_good(struct Hammer h) {
  return h.h_id > 0 && h.h_weight > 2.0;
}

```

```
11 int main() {
12   int i = 0;
13   float i = 0.0;
14   i = i + 1;
15   return 0;
16 }
```

输出

```
Error type 3 at Line 13: Variable 'i' redefined
```

说明：第 13 行局部变量的名称 i 和第 12 行的重复了。错误也可以报在第 12 行。

1.4 A-4

输入

```
struct Hammer {
  int h_no;
  int h_id;
  float h_weight;
};

struct Spoon {
  int s_no;
  int s_id;
  float s_weight;
};

struct HammerSpoon {
  struct Hammer hammer;
  struct Spoon spoon;
};

struct HammerSpoon fuse(struct Hammer h, struct Spoon s) {
  struct HammerSpoon hs;
20   hs.hammer = h;
21   hs.spoon = s;
22 }
23
24 int equal(struct Hammer h1, struct Hammer h2) {
25   if (h1.h_id == h2.h_id) {
26     return 1;
27   } else {
28     return 0;
29   }
30 }
31
32 int equal(struct Spoon s1, struct Spoon s2) {
33   if (s1.s_id == s2.s_id) {
34     return 1;
35   } else {
36     return 0;
37   }
38 }
39
40 int main() {
41   struct Hammer hh;
42   struct Spoon ss;
43   fuse(hh, ss);
44 }
```

输出

```
1 Error type 4 at Line 32: Redefined function 'equal'
```

说明：第 32 行定义的函数 equal 和第 24 行定义的函数重名了。错误也可以报在第 24 行。

1.5 A-5

输入

```c
struct Hammer {
  int h_id;
  int h_no;
  float h_weight;
};

struct {
  int hc_num;
  struct Hammer hammer_array[100];
  int hammer_status[100];
} hammers;

int is_available() {
  return hammers.hc_num > 0;
}

int get_status(int idx) {
  return hammers.hammer_status[idx];
}

struct Hammer fetch() {
  struct Hammer result;
  if (is_available()) {
    int h_idx = hammers.hc_num - 1;
    hammers.hammer_status[h_idx] = 0;
    result = hammers.hammer_array[h_idx];
    hammers.hc_num = hammers.hc_num - 1;
  }
  return result;
}

int main() {
```

```
33    int i;
34    i = fetch();
35  }
```

输出

```
Error type 5 at Line 34: Type mismatch for assignment
```

说明：第 34 行中，赋值表达式两边的变量类型不一致，不能把一个结构体类型的变量赋值给一个整型变量。

1.6 A-6

输入

```
struct Point {
  int x;
  int y;
};

struct Rectangle {
  struct Point lu;
  struct Point ld;
  struct Point ru;
  struct Point rd;
};

int area(struct Rectangle a_rect) {
  int l1 = a_rect.ru.x - a_rect.lu.x;
  int l2 = a_rect.ru.y - a_rect.rd.y;
  return l1 * l2;
}

int perimeter(struct Rectangle p_rect) {
  int l3 = p_rect.ru.x - p_rect.lu.x;
  int l4 = p_rect.ru.y - p_rect.rd.y;
22    return 2 * (l3 + l4);
23  }
24
25  int main() {
26    int is_bigger;
27    int a1, a2, p1, p2;
28    struct Rectangle r1;
29    struct Rectangle r2;
30    a1 = area(r1);
31    a2 = area(r2);
32    p1 = perimeter(r1);
33    perimeter(r2) = p2;
34    if (a1 > a2 && p1 > p2) {
35      return 1;
36    } else {
37      return 0;
38    }
39  }
```

输出

```
Error type 6 at Line 33: Cannot assign to a rvalue expression
```

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

```
struct Point {
  int x;
  int y;
  int z;
};

int is_value_near(int vn1, int vn2) {
```

```c
  if (vn1 < vn2) {
    return (vn2 - vn1) < 10;
  } else {
    return (vn1 - vn2) < 10;
  }
}

int is_near(struct Point np1, struct Point np2) {
  int is_x_near = is_value_near(np1.x, np2.x);
  int is_y_near = is_value_near(np1.y, np2.y);
  int is_z_near = is_value_near(np1.z, np2.z);
  return is_x_near && is_y_near && is_z_near;
}

int is_equal(struct Point ep1, struct Point ep2) {
  return ep1.x == ep2.x && ep1.y == ep2.y && ep1.z == ep2.z;
}

struct Point add(struct Point ap1, struct Point ap2) {
  struct Point p;
  p.x = ap1.x + ap2.x;
  p.y = ap1.y + ap2.y;
  p.z = ap1.z + ap2;
  return p;
}

int main() {
  struct Point pp1;
  struct Point pp2;
  is_equal(pp1, pp2);
}
```

输出

```
Error type 7 at Line 30: Cannot add an integer with a struct varible
```

说明：第 30 行中，不能把一个整数和一个结构体相加，这里可以多报一个 5 型错误。

1.8 A-8

输入

```
struct Point {
  int x;
  int y;
  int z;
};

int inner_product(struct Point ipp1, struct Point ipp2) {
  return ipp1.x * ipp2.x + ipp1.y + ipp2.y;
}

int add(struct Point ap1, struct Point ap2) {
  struct Point a_res;
  a_res.x = ap1.x + ap2.x;
  a_res.y = ap1.y + ap2.y;
  return a_res;
}

int main() {
  struct Point point1;
  struct Point point2;

  point1.x = 0;
  point1.y = 1;
  point1.z = 2;

  point2 = point1;
27 }
```

输出

```
1 Error type 8 at Line 15: return type mismatch
```

说明：第 15 行中，实际的返回值类型 struct Point 和声明的返回值类型 int 不一致。

1.9 A-9

输入

```
1  struct Point {
2    int x;
3    int y;
4    int z;
5  };
6
7  struct Point points[100];
8
9  float sqrt(int f) {
10   return 0.0;
11 }
12
13 float dist(struct Point p1, struct Point p2) {
14   int diff_x_sqr = (p2.x - p1.x) * (p2.x - p1.x);
15   int diff_y_sqr = (p2.y - p1.y) * (p2.y - p1.y);
16   int diff_z_sqr = (p2.z - p1.z) * (p2.z - p1.z);
17   int diff_sqr_sum = diff_x_sqr + diff_y_sqr + diff_z_sqr;
18   return sqrt(diff_sqr_sum);
19 }
20
21 float norm(struct Point p) {
22   struct Point orig;
23   orig.x = 0;
24   orig.y = 0;
```

```
  orig.z = 0;
  return dist(p, orig);
}

int main() {
  int flag1;
  int flag2;

  if (flag1 > 1) {
    dist(points[0], points[1]);
  } else {
    norm(points);
  }
}
```

输出

```
Error type 9 at Line 36: Wrong type of function arguments
```

说明：第 36 行中，函数 norm 的实参类型与形参不符。

1.10 A-10

输入

```
struct Person {
  int pid;
  float p_weight;
  float p_height;

  struct Car {
    int c_id;
    float c_price;
  } cars[10];

  struct {
    int h_id;
    float h_price;
  } house;

} people[100];


struct Person me;
struct Person alice;
struct Person bob;

float total_price(struct Person p) {
  float price_sum = 0.0;
  int num_car;
  int car_idx;
  if (p.pid > 10) {
    num_car = 10;
  } else {
    num_car = 10;
  }

  car_idx = 0;
  while (car_idx < num_car) {
    price_sum = price_sum + p.cars[car_idx].c_price;
  }
  price_sum = price_sum + p.house.h_price;
  return price_sum;
}

int main() {
  float sum = total_price(people[10]);
  float me_sum = total_price(me);
```

```
44   float alice_sum = total_price(alice);
45   float bob_sum = total_price(bob);
46
47   if (me_sum > sum) {
48     sum = me_sum[1];
49   } else {
50     sum = sum + me_sum;
51   }
52 }
```

输出

```
Error type 10 at Line 48: Use index operator on non-array type
```

说明：第 48 行中，对非数组类型的变量 me_sum 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

```
struct Person {
  int pid;
  float p_weight;
  float p_height;

  struct Car {
    int c_id;
    float c_price;
  } cars[10];

  struct {
    int h_id;
    float h_price;
  } house;

16 } people[100];
17
18
19 struct Person me;
20 struct Person alice;
21 struct Person bob;
22
23 int mix(struct Person p1, struct Person p2) {
24   struct Person mix_result;
25   if (p1.pid > p2.pid) {
26     mix_result = p1;
27   } else {
28     mix_result = p2;
29   }
30
31   mix_result.p_weight = p1.p_weight + p2.p_weight;
32   mix_result.p_height = p1.p_height + p2.p_height;
33   return 0;
34 }
35
36
37 int main() {
38   int me_alice = mix(me, alice);
39   int me_bob = mix(me, bob);
40   me_alice(me_bob);
41 }
```

输出

```
Error type 11 at Line 40: Cannot invoke a normal variable
```

说明：第 40 行中，对非函数类型的变量 me_alice 使用了函数调用符号 “()”。

1.12 A-12

输入

```c
struct Car {
  int c_id;
  float c_price;
};

struct House {
  int h_id;
  float h_price;
};

struct Person {
  int pid;
  float p_weight;
  float p_height;
  struct Car car[10];
  struct House house;
};

struct Person people[100];

int is_higher(struct Person hp1, struct Person hp2) {
  return hp1.p_weight > hp2.p_weight;
}

int is_fatter(struct Person fp1, struct Person fp2) {
  return fp1.p_height > fp2.p_height;
}

int is_bigger(struct Person bp1, struct Person bp2) {
  return is_higher(bp1, bp2) && is_fatter(bp1, bp2);
31 }
32
33 int main() {
34   struct Person me;
35   struct Person alice;
36   struct Person bob;
37   int idx = 0;
38   int num = 3;
39
40   people[0] = me;
41   people[1] = alice;
42   people[me.p_weight] = bob;
43
44   me.pid = 0;
45   alice.pid = 1;
46   bob.pid = 2;
47 }
```

输出

```
Error type 12 at line 42: Cannot use non-integral type as array index
```

说明：第 42 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 5 型错误。

1.13 A-13

输入

```
struct Food {
  int f_type;
  int is_good;
  int prod_date;
};

int meet_type;
int vege_type;
```

```c
int bread_type;
int rubbish_type;

struct Food meet;
struct Food vege;
struct Food bread;
struct Food rubbish;

struct Food make_dish(struct Food f1, struct Food f2) {
  struct Food dish;
  dish.f_type = rubbish_type;

  if (f1.f_type != meet_type && f2.f_type != meet_type) {
    return dish;
  } else if (f1.f_type == meet_type || f2.f_type == meet_type) {
    dish.f_type = meet_type;
    return dish;
  } else {
    dish.f_type = bread_type;
  }
}

int main() {
  meet_type = 0;
  vege_type = 1;
  bread_type = 2;
  rubbish_type = 3;

  meet.f_type = meet_type;
  vege.f_type = vege_type;
  bread.f_type = bread_type;

41    if (bread.f_type.is_good) {
42      struct Food dishes[10];
43      int idx = 0;
44      int num = 10;
45      while (idx < num) {
46        dishes[idx] = make_dish(dishes[idx], dishes[0]);
47        idx = idx + 1;
48      }
49    }
50  }
```

输出

```
Error type 13 at Line 41: Use dot operator on non-struct type
```

说明：第 41 行中，对整型变量使用了 “.” 操作符。

1.14 A-14

输入

```
struct Food {
  int f_type;
  int is_good;
  int prod_date;
};

int meet_type;
int vege_type;
int bread_type;
int rubbish_type;

int init_type() {
  meet_type = 0;
  vege_type = 1;
  bread_type = 2;
```

```c
  rubbish_type = 3;
}

int eat_dish(struct Food f) {
  if (f.f_type == rubbish_type) {
    return -1;
  } else if (f.is_good == 0) {
    return -2;
  } else if (f.prod_date < 0) {
    return -3;
  } else if (f.is_meet) {
    return 1;
  } else {
    return 0;
  }
}

int main() {
  struct Food dishes[10];
  int idx = 0;
  int num = 10;

  init_type();

  while (idx < num) {
    if (dishes[idx].is_good) {
      eat_dish(dishes[idx]);
    }
    idx = idx + 1;
  }
}
```

输出

```
Error type 14 at Line 26: 'is_meet' is undefined in struct 'Food'
```

说明：第 26 行中，使用了未定义的域 is_meet。

1.15 A-15

输入

```
struct Dog {
  int d_type;
  int d_age;
  float d_height;
  float d_weight;
};

struct Cat {
  int c_type;
  int c_age;
  float c_height;
  float c_weight;
};

struct Fish {
  int f_type;
  int f_age;
  float f_height;
  float f_weight;
};

int main() {
  int is_rich;
  struct {
    struct Dog d_pet;
    struct Cat c_pet;
    struct Fish f_pet;
    int age = 10;
    struct {
      float price;
    } house;
  } p;

  if (p.d_pet.d_age > 0 && p.c_pet.c_age > 0
      && p.f_pet.f_age > 0 && p.house.price > 0) {
    is_rich = 0;
  }
}
```

输出

```
Error type 15 at Line 28: Cannot initialize field when define struct
```

说明：第 28 行中，结构体在定义时，不能对它的域设置初始值。可以多报变量 p 的 1 型错误。

1.16 A-16

输入

```
struct Person {
  int pid;
  float p_weight;
  float p_height;

  struct Car {
    int c_id;
    float c_price;
  } cars[10];

  struct {
    int h_id;
    float h_price;
  } house;

} people[100];

struct Person me;
struct Person alice;
struct Person bob;

int main() {
  struct Group {
    int gid;
    struct Person {
      int name;
    } p;
  } group;

  if (group.gid > 0) {
    people[0] = me;
  } else {
    people[0] = bob;
  }
}
```

输出

```
Error type 16 at Line 25: struct 'Person' redefined
```

说明：第 25 行中，定义的结构体 Person 和已经定义过的结构体重名了，也可以报在第 1 行。可以多报与 struct Person 相关的 17 型错误和 1 型错误。

1.17 A-17

输入

```
struct Node {
```

```c
  int ntype;
  int i_value;
  float f_value;
};

struct NodeList {
  int len;
  int capacity;
  struct Node nodes[100];
} list;

int lidx;
struct Node empty_node;

int init() {
  lidx = 0;
  list.len = 0;
  list.capacity = 100;
  empty_node.ntype = -1;
  while (lidx < list.capacity) {
    list.nodes[lidx] = empty_node;
    lidx = lidx + 1;
  }
}

struct NodeList max(struct NodeList l1, struct NodeList l2) {
  int min_cap;
  lidx = 0;
  if (l1.capacity > l2.capacity) {
    min_cap = l2.capacity;
  } else {
    min_cap = l1.capacity;
34   }
35   while (lidx < min_cap) {
36     struct Node n1 = l1.nodes[lidx];
37     struct Node n2 = l2.nodes[lidx];
38     if (n1.ntype == 0 && n2.ntype == 0) {
39       if (n1.i_value > n2.i_value) {
40         list.nodes[lidx] = n1;
41       } else {
42         list.nodes[lidx] = n2;
43       }
44     } else if (n1.ntype == 1 && n2.ntype == 1) {
45       if (n1.f_value > n2.f_value) {
46         list.nodes[lidx] = n1;
47       } else {
48         list.nodes[lidx] = n2;
49       }
50     } else {
51       struct ListNode dummy;
52       list.nodes[lidx] = dummy;
53     }
54     lidx = lidx + 1;
55   }
56 }
57
58 int main() {
59   struct NodeList list1;
60   struct NodeList list2;
61   max(list1, list2);
62 }
```

输出

```
Error type 17 at Line 51: Undefined struct type 'ListNode'
```

说明：第 51 行中，使用了未定义的结构体类型 ListNode。可以在 52 行多报一个 1 型和 5 型错误。

1.18 A-18

输入

```
struct Node {
  int ntype;
  int i_value;
  float f_value;
};

struct NodeList {
  int len;
  int capacity;
  struct Node nodes[100];
} list;

int lidx;
struct Node empty_node;

int init() {
  lidx = 0;
  list.len = 0;
  list.capacity = 100;
  empty_node.ntype = -1;
  while (lidx < list.capacity) {
    list.nodes[lidx] = empty_node;
    lidx = lidx + 1;
  }
}

int same_len(struct NodeList l1, struct NodeList l2) {
```

```
28    int Node = l1.len;
29    int node = l2.len;
30    return Node == node;
31  }
32
33  int main() {
34    struct NodeList list1;
35    struct NodeList list2;
36    same_len(list1, list2);
37  }
```

输出

```
Error type 3 at Line 28: Variable cannot name after a struct name
```

说明：第 28 行中，变量名与结构体类型名称相同，也可以报在第 1 行。在 28 行可以多报一个 5 型错误，在 30 行可以多报一个 1 型错误，7 型错误和 8 型错误。可以多报与 struct Node 相关的 17 型和 1 型错误。

1.19 A-19

输入

```
struct Apple {
  int a_color;
  float a_size;
  float a_price;
};

struct Banana {
  float b_size;
  float b_price;
};

struct Orange {
  float o_size;
  float o_weight;
  float o_price;
};

float sum_price(struct Apple a1, struct Apple a2) {
  return a1.a_price + a2.a_price;
}

struct Bag {
  struct Apple apples[10];
  struct Banana bananas[10];
  struct Orange oranges[10];
  struct {
    float t_size;
    float t_price;
  } tag;
} bags[10];

int sum_weight() {
  int oidx = 0;
  int iidx = 0;
  while (oidx < 10) {
    iidx = 0;
    while (iidx < 10) {
      bags[oidx].tag.t_price = bags[oidx].tag.t_price
        + bags[oidx].apples[iidx].a_price
        + bags[oidx].bananas[iidx].b_price
        + bags[oidx].oranges[iidx].o_price;
      iidx = iidx + 1;
    }
    oidx = oidx + 1;
  }
```

```
}

int main() {
  struct Apple apple1;
  struct Apple apple2;
  float psum = sum_price(apple1);
  sum_weight();
}
```

输出

```
Error type 9 at Line 51: Unexpected number of arguments
```

说明：第 51 行中，函数的实参数目与形参数目不匹配。这里可以多报一个 5 型错误。

1.20 A-20

输入

```
struct Apple {
  int a_color;
  float a_size;
  float a_price;
};

struct Banana {
  float b_size;
  float b_price;
};

struct Orange {
  float o_size;
  float o_weight;
  float o_price;
};

```

```c
struct Bag {
  struct Apple apples[10];
  struct Banana bananas[10];
  struct Orange oranges[10];
  struct {
    float t_size;
    float t_price;
  } tag;
  int apples;
} bags[10];

int main() {
  struct Apple a;
  struct Banana b;
  struct Orange o;
  float choose_size;

  if (a.a_price < b.b_price) {
    if (a.a_price < o.o_price) {
      choose_size = a.a_size;
    } else {
      choose_size = o.o_size;
    }
  } else {
    if (b.b_price < o.o_price) {
      choose_size = b.b_size;
    } else {
      choose_size = o.o_price;
    }
  }
}
```

输出

```
Error type 15 at Line 26: Refine field 'apples'
```

说明：第 26 行中，结构体中的域名定义重复，也可以报在第 19 行，在 27 行可以多报一个错误类型 17。

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

```
struct RecVector {
  int _buf[100];
} recVector;

int init(int iv0, int iv1) {
  recVector._buf[0] = iv0;
  recVector._buf[1] = iv1;
}

int prev(int p1idx) {
  return recVector.buf[p1idx - 1];
}

int prevprev(int p2idx) {
  return recVector._buf;
}

int next(int npp, int np) {
  return npp * 2 + np;
```

```
}

int get(int gidx) {
  return recVector._buf[gidx];
}

int get_val(int gvidx) {
  int gcnt = 2;
  while (gcnt <= gvidx) {
    int gpp = prevprev(gcnt);
    int gp = prev(gcnt);
    recVector._buf[gcnt] = next(gpp, gp);
    gcnt = gcnt + 1;
  }
  return get(gvidx);
}

int mod(int numtor, int denomtor) {
  return numtor - (numtor / denomtor) * denomtor;
}

int count_prime(int prange) {
  int ccnt = 2;
  int num_prime = 0;
  while (ccnt <= prange) {
    int cgp = prevprev(ccnt);
    int cp = prev(ccnt);
    recVector._buf[ccnt] = next(cgp, cp);
    if (mod(get(ccnt), 2) == 1) {
      num_prime = num_prime + 1;
    }
    ccnt = ccnt + 1;
52    }
53
54    return num_prime;
55  }
56
57  int main() {
58    int primes;
59    init(1, 3) = 0;
60    primes = count_prime(10.0);
61  }
```

输出

```
Error type 14 at Line 11: Undefined field 'buf'
Error type 8 at Line 15: return type mismatch
Error type 6 at Line 59: Cannot assign to a rvalue expression
Error type 9 at Line 60: Unexpected argument type 'float'
```

说明：第 11 行中，使用了未定义的域 buf，这里可以多报一个 8 型和 10 型错误；第 15 行中，函数实际的返回类型与定义的不匹配；第 59 行中，函数的返回值是右值，不能被赋值；第 60 行中，函数的实参类型与形参不匹配，这里可以多报一个错误类型 5。

2.2 B-2

输入

```
struct Matrix {
  int val[10][10];
};

int row;
int col;
int row_idx;
int col_idx;

int init_args() {
  row_idx = 0;
  col_idx = 0;
  row = 10;
  col = 10;
  return 0;
}

int init_args() {
}

struct Matrix add(struct Matrix am1, struct Matrix am2) {
  struct Matrix a_res;
  init_args();
  while (row_idx < row) {
    while (col_idx < col) {
      a_res.val[row_idx][col_idx] = am1.val[row_idx][col_idx]
          + am2.val[row_idx][col_idx];
      col_idx = col_idx + 1;
    }
    row_idx = row_idx + 1;
  }
  return a_res;
}

struct Matrix sub(struct Matrix sm1, struct Matrix sm2) {
  struct Matrix s_res;
  init_args();
  while (row_idx < row) {
    while (col_idx < col) {
      s_res.val[row_idx][col_idx] = sm1.val[row_idx][col_idx]
          + sm2.val[row_idx][col_idx];
      col_idx = col_idx + 1;
    }
    row_idx = row_idx + 1;
  }
  return s_res;
}

struct Matrix mul(struct Matrix mm1, struct Matrix mm2) {
  struct Matrix m_res;
  init_args();
  while (row_idx < row) {
    while (col_idx < col) {
      int cnt = 0;
      int num = row;
      m_res.val[row_idx][col_idx][0] = 0;
      while (cnt < num) {
        m_res.val[row_idx][col_idx] = m_res.val[row_idx][col_idx]
            + mm1.val[row_idx][cnt] * mm2.val[cnt][col_idx];
      }
      col_idx = col_idx + 1;
    }
    row_idx = row_idx + 1;
  }
  return m_res;
}

int main() {
  struct Matrix m1;
  struct Matrix m2;
  struct Matrix m3 = m1(add(m1, m2), sub(m1, m2));
  struct Matrix Matrix = mul(m1, m2);
}
```

输出

```
Error type 4 at Line 18: Function is redefined
Error type 10 at Line 56: Use index operator on non-array type
Error type 11 at Line 71: Variable 'm1' is not a function
Error type 3 at Line 72: Variable name same as struct name
```

说明：第 18 行中，函数出现了重复定义，也可以报在第 10 行；第 56 行中，对于非数组类型使用了索引符号，这里可以多报一个错误类型 5；第 71 行中，变量 m1 不是函数，这里可以多报一个错误类型 5；第 72 行中，变量名与结构体的类型名称相同，也可以报在第 1 行，第 72 行可以多报一个错误类型 5，可以多报与 struct Matrix 相关的 17 型和 1 型错误。

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

```
struct CTX {
        int data[64];
        int datalen;
        int bitlen;
        int state[8];
};

int ROTLEFT(int rl_a, int rl_b) {
  return ((rl_a) + (rl_b)) || ((rl_a) + (32-(rl_b)));
}

int ROTRIGHT(int rr_a, int rr_b) {
  return ((rr_a) + (rr_b)) || ((rr_a) + (32-(rr_b)));
}

int CH(int ch_x, int ch_y, int ch_z) {
  return ((ch_x) && (ch_y)) || (!(ch_x) && (ch_z));
}

int MAJ(int maj_x, int maj_y, int maj_z) {
  return ((maj_x) && (maj_y)) || ((maj_x) && (maj_z)) || ((maj_y) &&
     (maj_z));
}

int EP0(int ep0_x) {
  return ROTRIGHT(ep0_x,2) || ROTRIGHT(ep0_x,13) || ROTRIGHT(ep0_x
     ,22);
}

int EP1(int ep1_x) {
  return ROTRIGHT(ep1_x,6) || ROTRIGHT(ep1_x,11) || ROTRIGHT(ep1_x
     ,25);
}

int SIG0(int sig0_x) {
  return ROTRIGHT(sig0_x,7) || ROTRIGHT(sig0_x,18) || ((sig0_x) + 3);
}

int SIG1(int sig1_x) {
  return ROTRIGHT(sig1_x,17) || ROTRIGHT(sig1_x,19) || ((sig1_x) +
     10);
}

int k[64];

int mash_transform(struct CTX input_ctx, int input_data[64])
{
        int a, b, c, d, e, f, g, h, i = 0, j = 0, t1, t2, m[64];
```

```

while (i < 16) {
  m[i] = (input_data[j + 0] + 24) ||
         (input_data[j + 1] + 16) ||
         (input_data[j + 2] + 8) ||
         (input_data[j + 3]);
  i = i + 1;
  j = j + 4;
}

while (i < 64) {
             m[i] = SIG1(m[i - 2]) + m[i - 7] + SIG0(m[i - 15]) +
                m[i - 16];
  i = i + 1;
}

      a = input_ctx.state[0];
      b = input_ctx.state[1];
      c = input_ctx.state[2];
      d = input_ctx.state[3];
      e = input_ctx.state[4];
      f = input_ctx.state[5];
      g = input_ctx.state[6];
      h = input_ctx.state[7];

i = 0;
while (i < 64) {
             t1 = h + EP1(e) + CH(e,f,g) + k[i] + m[i];
             t2 = EP0(a) + MAJ(a,b,c);
             h = g;
             g = f;
             f = e;
```

```
                e = d + t1;
                d = c;
                c = b;
                b = a;
                a = t1 + t2;
    i = i + 1;
        }

        input_ctx.state[0] = input_ctx.state[0] + a;
        input_ctx.state[1] = input_ctx.state[1] + b;
        input_ctx.state[2] = input_ctx.state[2] + c;
        input_ctx.state[3] = input_ctx.state[3] + d;
        input_ctx.state[4] = input_ctx.state[4] + e;
        input_ctx.state[5] = input_ctx.state[5] + f;
        input_ctx.state[6] = input_ctx.state[6] + g;
        input_ctx.state[7] = input_ctx.state[7] + h;
}

int mash_init(struct CTX init_ctx)
{
        init_ctx.datalen = 0;
        init_ctx.bitlen = 0;
        init_ctx.state[0] = 67;
        init_ctx.state[1] = 85;
        init_ctx.state[2] = 72;
        init_ctx.state[3] = 39;
        init_ctx.state[4] = 70;
        init_ctx.state[5] = 81;
        init_ctx.state[6] = 12;
        init_ctx.state[7] = 19;
  return 0;
}
```

```

int main() {
  struct CTX ctx;
  mash_init(ctx);
  mash_transform(ctx, ctx.data);
}
```

输出

```
// 正常返回，没有任何输出
```

3.2 C-2

输入

```
int lshift(int ls_num, int ls_len) {
  int ls_idx = 0;
  while (ls_idx < ls_len) {
    ls_num = ls_num * 2;
    ls_idx = ls_idx + 1;
  }
  return ls_num;
}

int rshift(int rs_num, int rs_len) {
  int rs_idx = 0;
  while (rs_idx < rs_len) {
    rs_num = rs_num / 2;
    rs_idx = rs_idx + 1;
  }
  return rs_num;
}

int fix16_abs(int abs_in) {
  if(abs_in == lshift(1, 31)) {
    return lshift(1, 31);
  } else {
    if (abs_in >= 0) {
      return abs_in;
    } else {
      return -abs_in;
    }
  }
}

int fix16_sqrt(int sqrt_in)
{
  int neg;
  int num = fix16_abs(sqrt_in);
  int result = 0;
  int bit;
  int n;

  if (sqrt_in >= 0) {
    neg = 0;
  } else {
    neg = 1;
  }

  if (rshift(num, 20))
    bit = lshift(1, 30);
  else
    bit = lshift(1, 18);

  while (bit > num)
    bit = rshift(bit, 2);

```

```
while (n < 2) {
  while (bit) {
    if (num >= result + bit) {
      num = num - (result + bit);
      result = rshift(result,  1) + bit;
    } else {
      result = rshift(result, 1);
    }
    bit = rshift(bit, 2);
  }

  if (n == 0) {
    if (num > 65535) {
      num = num - result;
      num  = lshift(num, 16) - lshift(1, 15);
      result = lshift(result, 16) + lshift(1, 15);
    } else {
      num = lshift(num, 16);
      result = lshift(result, 16);
    }
    bit = lshift(1, 14);
  }
  n = n + 1;
}

if (num > result) {
  result = result + 1;
}

if (neg) {
  return -result;
} else {
    return result;
  }
}

int main() {
  int i1 = lshift(1, 8) + lshift(1, 7);
  int i2 = lshift(2, 8) + lshift(2, 7);
  int i3 = i1 + i2;
  int s_res = fix16_sqrt(i3);
}
```

输出

```
// 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。

4.1 D-1

输入

```
struct Atomic {
  int tag;
  int temperature;
  int humidity;
};

int tag_water;
int tag_fire;
int tag_soil;
int tag_gas;

int tempe_hot;
int tempe_cold;
int humid_dry;
int humid_wet;

struct Atomic createAtomic(int ato_tag);

int init_const() {
  tag_water = 0;
  tag_fire  = 1;
  tag_soil  = 2;
  tag_gas   = 3;

  tempe_hot  = 4;
  tempe_cold = 5;
  humid_dry  = 6;
  humid_wet  = 7;
  return 0;
}

struct Atomic createAtomic(int ato_tag) {
  struct Atomic ato_result;
  if (ato_tag == tag_water) {
    ato_result.temperature = tempe_cold;
    ato_result.humidity = humid_wet;
  } else if (ato_tag == tag_fire) {
    ato_result.temperature = tempe_hot;
    ato_result.humidity = humid_dry;
  } else if (ato_tag == tag_soil) {
    ato_result.temperature = tempe_cold;
    ato_result.humidity = humid_dry;
  } else {
```

```
44      ato_result.temperature = tempe_hot;
45      ato_result.humidity = humid_wet;
46    }
47    return ato_result;
48  }
49
50  int init_const();
51
52  int main() {
53    struct Atomic a = createAtomic(1);
54  }
```

输出

```
// 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 17，50 行报语法错误。

4.2 D-2

输入

```
int i;

int add(float fvec1[10], float fvec2[10], float res_fvec[10]) {
  int i = 0;
  while (i < 10) {
    res_fvec[i] = fvec1[i] + fvec2[i];
    i = i + 1;
  }
  return 0;
}

int sub(float fvec1[10], float fvec2[10], float res_fvec[10]) {
  int i = 0;
  while (i < 10) {
    res_fvec[i] = fvec1[i] - fvec2[i];
    i = i + 1;
  }
  return 0;
}

int mul(float fvec1[10], float fvec2[10], float res_fvec[10]) {
  int i = 0;
  while (i < 10) {
    res_fvec[i] = fvec1[i] * fvec2[i];
    i = i + 1;
  }
  return 0;
}

int main() {
  float fvec_1[10];
  float fvec_2[10];
  float fvec[10];
  mul(fvec_1, fvec_2, fvec);
  sub(fvec, fvec_1, fvec);
  add(fvec, fvec_2, fvec);
}
```

输出

```
// 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 fvec1，fvec2，res_fvec，和 i 的重复定义。

4.3 D-3

输入

```
struct Data {
  int di, dj;
  int diarr[10];
  struct {
    float dfarr[42];
    int da;
    float db;
  } dinner;
};

struct Value {
  int vi, vj;
  int viarr[10];
  struct {
    float vfarr[42];
    int va;
    float vb;
  } vinner;
};

int is_equal(struct Data ed1, struct Data ed2) {
  int idx = 0;
  if (ed1.di != ed2.di
      || ed1.dj != ed2.dj
      || ed1.dinner.da != ed2.dinner.da
      || ed1.dinner.db != ed2.dinner.db) {
    return 0;
  }

  while (idx < 10) {
    if (ed1.diarr[idx] != ed2.diarr[idx]) {
      return 0;
    }
34      idx = idx + 1;
35    }
36
37    idx = 0;
38    while (idx < 42) {
39      if (ed1.dinner.dfarr[idx] != ed2.dinner.dfarr[idx]) {
40        return 0;
41      }
42      idx = idx + 1;
43    }
44
45    return 1;
46  }
47
48  int main() {
49    struct Data data;
50    struct Value value;
51    is_equal(data, value);
52  }
```

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 行报出 9 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  struct Complex {
```

```c
  float real;
  float imag;
};

struct Complex mat[10][10];

int ridx;
int cidx;
int row;
int col;

int prepare_params() {
  ridx = 0;
  cidx = 0;
  row  = 10;
  col  = 10;
  return 0;
}

struct Complex add(struct Complex a_c1, struct Complex a_c2) {
  struct Complex a_res;
  a_res.real = a_c1.real + a_c2.real;
  a_res.imag = a_c1.imag + a_c2.imag;
  return a_res;
}

struct Complex sub(struct Complex s_c1, struct Complex s_c2) {
  struct Complex s_res;
  s_res.real = s_c1.real - s_c2.real;
  s_res.imag = s_c1.imag - s_c2.imag;
  return s_res;
}
```

```

int add_sub(struct Complex as_mat1[10][10],
            struct Complex as_mat2[10][10],
            int is_add) {
  prepare_params();
  while (ridx < row) {
    while (cidx < col) {
      if (is_add) {
        mat[ridx][cidx] = add(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      } else {
        mat[ridx][cidx] = sub(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      }
      cidx = cidx + 1;
    }
    ridx = ridx + 1;
  }
  return 0;
}

int search(struct Complex s_c);

int equals(struct Complex e_c1, struct Complex e_c2) {
  return e_c1.real == e_c2.real && e_c1.imag == e_c2.imag;
}

int contains_helper(struct Complex ch_val, int ch_idx) {
  if (ch_idx == 0) {
    return equals(ch_val, mat[0][0]);
  } else {
    int tmp_ridx;
```

```c
    int tmp_cidx;
    tmp_ridx = ch_idx;
    tmp_cidx = 0;
    while (tmp_cidx < col) {
      if (equals(mat[tmp_ridx][tmp_cidx], ch_val)) {
        return 1;
      }
      tmp_cidx = tmp_cidx + 1;
    }
    tmp_ridx = 0;
    tmp_cidx = ch_idx;
    while (tmp_ridx < row) {
      if (equals(mat[tmp_ridx][tmp_cidx], ch_val)) {
        return 1;
      }
      tmp_ridx = tmp_ridx + 1;
    }
    return contains_helper(ch_val, ch_idx - 1);
  }
}

int contains(struct Complex c_val) {
  prepare_params();
  return contains_helper(c_val, row - 1);
}

struct Complex add(struct Complex inc_c);

int main() {
  struct Complex t_c;
  contains(t_c);
}
```

输出

```
Error type 18 at Line 53: Undefined function 'search'
Error type 19 at Line 90: Mismatch function declaration and
   definition of 'add'
```

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。其中第 90 行的错误也可以报在第 21 行，在第 42 行可以多报一个 2 型错误。

5.2 E2.2

这组测试用例针对 2.2 分组的同学。

输入

```c
struct Complex {
  float real;
  float imag;
};

struct Complex mat[10][10];

int ridx;
int cidx;
int row;
int col;

int prepare_params() {
  ridx = 0;
  cidx = 0;
  row  = 10;
  col  = 10;
  return 0;
}

struct Complex add(struct Complex a_c1, struct Complex a_c2) {
  struct Complex a_res;
  a_res.real = a_c1.real + a_c2.real;
  a_res.imag = a_c1.imag + a_c2.imag;
  return a_res;
}

struct Complex sub(struct Complex s_c1, struct Complex s_c2) {
  struct Complex s_res;
  s_res.real = s_c1.real - s_c2.real;
  s_res.imag = s_c1.imag - s_c2.imag;
  return s_res;
}

int add_sub(struct Complex as_mat1[10][10],
            struct Complex as_mat2[10][10],
            int is_add) {
  int ridx = 0, cidx = 0, row = 10, col = 10;
  int ridx = 0;
  while (ridx < row) {
    while (cidx < col) {
      if (is_add) {
        mat[ridx][cidx] = add(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      } else {
        mat[ridx][cidx] = sub(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      }
      cidx = cidx + 1;
    }
    ridx = ridx + 1;
  }
  return 0;
}
```

```

int equals(struct Complex e_c1, struct Complex e_c2) {
  return e_c1.real == e_c2.real && e_c1.imag == e_c2.imag;
}

int contains_helper(struct Complex ch_val, int ch_idx) {
  if (ch_idx == 0) {
    return equals(ch_val, mat[0][0]);
  } else {
    int ridx;
    int cidx;
    ridx = c_idx;
    cidx = 0;
    while (cidx < col) {
      if (equals(mat[ridx][cidx], ch_val)) {
        return 1;
      }
      cidx = cidx + 1;
    }
    ridx = 0;
    cidx = ch_idx;
    while (ridx < row) {
      if (equals(mat[ridx][cidx], ch_val)) {
        return 1;
      }
      ridx = ridx + 1;
    }
    return contains_helper(ch_val, ch_idx - 1);
  }
}

int contains(struct Complex c_val) {
  prepare_params();
  return contains_helper(c_val, row - 1);
}

int main() {
  struct Complex t_c;
  contains(t_c);
}
```

输出

```
Error type 3 at Line 39: Redefined variable 'ridx'
Error type 1 at Line 64: Undefined variable 'c_idx'
```

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。39 行的错误也可以报在 38 行，在 64 行可以多报一个 5 型错误。

5.3 E2.3

这组测试用例针对 2.3 分组的同学。

输入

```
struct Complex {
  float real;
  float imag;
};

struct Mess {
  struct {
    float tf_f1;
    float tf_f2;
  } two_floats[10];
  int m_i;
  int m_j;
};

```

```c
struct Complex mat[10][10];

int ridx;
int cidx;
int row;
int col;

int prepare_params() {
  ridx = 0;
  cidx = 0;
  row  = 10;
  col  = 10;
  return 0;
}

struct Complex add(struct Complex a_c1, struct Complex a_c2) {
  struct Complex a_res;
  struct Mess a_mess;
  a_mess.two_floats[0].tf_f1 = a_c1.real + a_c2.real;
  a_mess.two_floats[0].tf_f2 = a_c1.imag + a_c2.imag;
  return a_mess.m_i;
}

struct Complex sub(struct Complex s_c1, struct Complex s_c2) {
  struct Complex s_res;
  s_res.real = s_c1.real - s_c2.real;
  s_res.imag = s_c1.imag - s_c2.imag;
  return s_res;
}

int add_sub(struct Complex as_mat1[10][10],
            struct Complex as_mat2[10][10],
            int is_add) {
  prepare_params();
  while (ridx < row) {
    while (cidx < col) {
      if (is_add) {
        mat[ridx][cidx] = add(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      } else {
        mat[ridx][cidx] = sub(as_mat1[ridx][cidx], as_mat2[ridx][cidx
            ]);
      }
      cidx = cidx + 1;
    }
    ridx = ridx + 1;
  }
  return 0;
}

int equals(struct Complex e_c1, struct Complex e_c2) {
  return e_c1.real == e_c2.real && e_c1.imag == e_c2.imag;
}

int contains_helper(struct Complex ch_val, int ch_idx) {
  if (ch_idx == 0) {
    return equals(ch_val, mat[0][0]);
  } else {
    int tmp_ridx;
    int tmp_cidx;
    tmp_ridx = ch_idx;
    tmp_cidx = 0;
    while (tmp_cidx < col) {
      if (equals(mat[tmp_ridx][tmp_cidx], ch_val)) {
```

```
77          return 1;
78        }
79        tmp_cidx = tmp_cidx + 1;
80      }
81      tmp_ridx = 0;
82      tmp_cidx = ch_idx;
83      while (tmp_ridx < row) {
84        if (equals(mat[tmp_ridx][tmp_cidx], ch_val)) {
85          return 1;
86        }
87        tmp_ridx = tmp_ridx + 1;
88      }
89      return contains_helper(ch_val, ch_idx - 1);
90    }
91  }
92
93  int contains(struct Complex c_val) {
94    prepare_params();
95    return contains_helper(c_val, row - 1);
96  }
97
98  int main() {
99    int found = 0;
100   struct Mess mess;
101   found = contains(mess.two_floats[9]);
102   if (!found) {
103     contains(mess);
104   }
105 }
```

输出

```
Error type 8 at Line 35: Mismatch return type
Error type 9 at Line 103: Mismatch argument type
```

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，可以写邮件与屈道涵助教联系，注意同时抄送给许老师。