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 7，9，15。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

```
struct Person {
  float height;
  float weight;
  int id;
};

int main(){
  struct Person person;
  int a,b,c,d;
  float e,f,g,h;
  float hei = 170.0;
  float wei = 75.0;
  person.height = hei;
  person.weight = wei;
  person.id = p;
  return 0;
}
```

输出

```
Error type 1 at Line 15: Undefined variable.
```

说明：第 15 行中，p 这个变量没有定义过。这里可以多报一个 5 型错误。

1.2 A-2

输入

```
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1
      .y - pp2.y);
}

float main(){
  struct Position p1, p2;
  dis(p1, p2);
  return 1.11;
}
```

输出

```
Error type 2 at Line 12: Undefined function 'dis'.
```

说明：第 12 行中，函数 dis 没有没有定义过。

1.3 A-3

输入

```
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1
      .y - pp2.y);
```

```
 8   }
 9
10   float main(){
11     struct Position p1, p2;
12     float p1 = cal(p1, p2);
13     return 0.1;
14   }
```

输出

```
Error type 3 at Line 12: Redefined variable 'p1'.
```

说明：第 12 行局部变量的名称 p1 和第 11 行的重复了。错误也可以报在第 11 行。

1.4 A-4

输入

```
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1
      .y - pp2.y);
}

float equal(struct Position pp3, struct Position pp4){
  return pp3.x - pp4.x;
}

float equal(struct Position pp5, struct Position pp6){
  return pp5.y - pp6.y;
}

```

```
float main(){
  struct Position p1, p2;
  return cal(p1, p2);
}
```

输出

```
Error type 4 at Line 14: Redefined function 'equal'.
```

说明：第 14 行定义的函数 equal 和第 10 行定义的函数重名了。错误也可以报在第 10 行。

1.5 A-5

输入

```
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1.y - pp2.y);
}

float xDis(struct Position pp3, struct Position pp4){
  return pp3.x - pp4.x;
}

float yDis(struct Position pp5, struct Position pp6){
  return pp5.y - pp6.y;
}

struct {
  int _i;
  int _j;
  int _ads;
} persons;

float main(){
  struct Position p1, p2;
  p1.x = persons._j;
  return cal(p1, p2);
}
```

输出

```
Error type 5 at Line 26: Different types at both side of =.
```

说明：第 26 行中，赋值表达式两边的变量类型不一致，不能把一个浮点数变量赋值给一个整型变量。

1.6 A-6

输入

```
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1
      .y - pp2.y);
}

float xDis(struct Position pp3, struct Position pp4){
  return pp3.x - pp4.x;
}

float yDis(struct Position pp5, struct Position pp6){
  return pp5.y - pp6.y;
16  }
17
18  struct {
19    int _i;
20    int _j;
21    int _ads;
22  } persons;
23
24  struct tempStruct{
25    float _f;
26    float _g;
27  } structures;
28
29  float main(){
30    struct Position p1, p2;
31    yDis(p1, p2) = structures._g;
32    return cal(p1, p2);
33  }
```

输出

```
Error type 6 at Line 31: Invalid left value.
```

说明：第 31 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 Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
    return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1
        .y - pp2.y);
}

float xDis(struct Position pp3, struct Position pp4){
    return pp3.x - pp4.x;
}

float yDis(struct Position pp5, struct Position pp6){
    return pp5.y - pp6.y;
}

struct {
    int _i;
    int _j;
    int _ads;
} persons;

struct tempStruct{
    float _f;
    float _g;
} structures[100];

float main(){
    struct Position p1, p2;
    structures[0]._g = yDis(p1, p2);
    persons + structures;
    return cal(p1, p2);
}
```

输出

```
Error type 7 at Line 32: Bad type(s) for '+' operation.
```

说明：第 32 行中，不能把一个数组和一个结构体相加。

1.8 A-8

输入

```c
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1.y - pp2.y);
}

float xDis(struct Position pp3, struct Position pp4){
  return pp3.x - pp4.x;
}

float yDis(struct Position pp5, struct Position pp6){
  return pp5.y - pp6.y;
}

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

float main(){
  struct Position p1, p2;
  inner_product(p1, p2);
  return p1;
}
```

输出

```
Error type 8 at Line 25: Return type mismatch.
```

说明：第 25 行中，实际的返回值类型 struct Position 和声明的返回值类型 float 不一致。

1.9 A-9

输入

```c
struct Position {
  float x;
  float y;
};

float cal(struct Position pp1, struct Position pp2){
  return (pp1.x - pp2.x) * (pp1.x - pp2.x) + (pp1.y - pp2.y) * (pp1.y - pp2.y);
}

float xDis(struct Position pp3, struct Position pp4){
  return pp3.x - pp4.x;
}

float yDis(struct Position pp5, struct Position pp6){
  return pp5.y - pp6.y;
}

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

float main(){
  struct Position p1, p2;
  struct Position p[100];
  inner_product(p[0], p1, p2);
  return cal(p1, p2);
}
```

输出

```
Error type 9 at Line 25: Funtion args mismatch.
```

说明：第 25 行中，函数 inner_product 的实参数量与形参数量不符。

1.10　A-10

输入

```
struct ClassRoom {
  int cid;
  struct Position {
    int bid;
    int rid;
  } position;

  struct Teacher {
    int tid;
    int tgender;
    int course;
  } teacher;

  struct Student {
    int sid;
    int sgender;
    int grade;
  } students[100];

} classRoom1, classRoom2, classRoom3;

int ave_grade(struct ClassRoom cr){
  int sum = 0;
  int index = 0;
  while(index < 100){
    sum = sum + cr.students[index].grade;
  }
  return sum / 100;
}

int main() {
  int ave1 = ave_grade(classRoom1);
  int ave2 = ave_grade(classRoom2);
  int ave3 = ave_grade(classRoom3);

  if(ave1 > ave2 && ave1 > ave3[0]){
    return 1;
  }
  else{
    return 0;
  }
}
```

输出

```
Error type 10 at Line 36: Apply [] to non-array variable.
```

说明：第 36 行中，对非数组类型的变量 ave3 使用了数组索引符号 “[]”。这里可以多报一个 7 型错误。

1.11 A-11

输入

```
struct ClassRoom {
  int cid;
  struct Position {
    int bid;
    int rid;
  } position;

  struct Teacher {
    int tid;
    int tgender;
    int course;
  } teacher;

  struct Student {
    int sid;
    int sgender;
    float grade;
  } students[100];

} classRoom1, classRoom2, classRoom3;

float ave_grade(struct ClassRoom cr){
  float sum = 0.0;
  int index = 0;
  while(index < 100){
    sum = sum + cr.students[index].grade;
  }
  return sum / 100.0;
}

int main() {
  float ave1 = ave_grade(classRoom1);
  float ave2 = ave_grade(classRoom2);
  float ave3 = ave_grade(classRoom3);

  ave1(ave2, ave3);
}
```

输出

```
Error type 11 at Line 36: This is not a function.
```

说明：第 36 行中，对非函数类型的变量 av1 使用了函数调用符号 “()”。

1.12 A-12

输入

```
struct ClassRoom {
  int cid;
  struct Position {
    int bid;
    int rid;
  } position;

  struct Teacher {
    int tid;
    int tgender;
    int course;
  } teacher;

  struct Student {
    int sid;
    int sgender;
    float grade;
  } students[100];

} classRoom1, classRoom2, classRoom3;

float ave_grade(struct ClassRoom cr1){
  float sum = 0.0;
  int index = 0;
  while(index < 100){
    sum = sum + cr1.students[index].grade;
27      index = index + 1;
28    }
29    return sum / 100.0;
30  }
31
32  float sumUp(float g1, float g2, float g3){
33    return g1 + g2 + g3;
34  }
35
36  int unique_sid_check(struct ClassRoom cr2, float ind){
37     int ssid = cr2.students[ind].sid;
38     int i = 0;
39     while(i < 100){
40        if(cr2.students[i].sid == ssid){
41          return 0;
42        }
43        i = i + 1;
44     }
45     return 1;
46  }
47
48  float main() {
49    float ave1 = ave_grade(classRoom1);
50    float ave2 = ave_grade(classRoom2);
51    float ave3 = ave_grade(classRoom3);
52
53    sumUp(ave1, ave2, ave3);
54
55    unique_sid_check(classRoom1, 1.0);
56  }
```

输出

```
Error type 12 at Line 37: Non-integer index of array.
```

说明：第 37 行中，不能使用 float 类型的变量作为数组的索引，可以多报一个 13 型错误和 5 型错误

1.13 A-13

输入

```
struct ClassRoom {
  int cid;
  struct Position {
    int bid;
    int rid;
  } position;

  struct Teacher {
    int tid;
    int tgender;
    int course;
  } teacher;

  struct Student {
    int sid;
    int sgender;
    float grade;
  } students[100];

} classRoom1, classRoom2, classRoom3;

float ave_grade(struct ClassRoom cr1){
  float sum = 0.0;
  int index = 0;
  while(index < 100){
    sum = sum + cr1.students[index].grade;
27      index = index + 1;
28    }
29    return sum / 100.0;
30  }
31
32  float sumUp(float g1, float g2, float g3){
33    return g1 + g2 + g3;
34  }
35
36  int unique_sid_check(struct ClassRoom cr2, int ind){
37     int ssid = cr2.students[ind].sid.id;
38     int i = 0;
39     while(i < 100){
40        if(cr2.students[i].sid == ssid){
41          return 0;
42        }
43        i = i + 1;
44     }
45     return 1;
46  }
47
48  float main() {
49    float ave1 = ave_grade(classRoom1);
50    float ave2 = ave_grade(classRoom2);
51    float ave3 = ave_grade(classRoom3);
52
53    sumUp(ave1, ave2, ave3);
54
55    unique_sid_check(classRoom1, 1);
56  }
```

输出

```
Error type 13 at Line 37: Applying . to non-structure variable 0.
```

说明：第 37 行中，对整型变量使用了 “.” 操作符。这里可以多报一个 5 型错误.

1.14 A-14

输入

```
 1  int a;
 2  float b;
 3
 4  struct Car {
 5    int c_id;
 6    float c_speed;
 7    struct Location {
 8      float c_longitude;
 9      float c_latitue;
10    } c_location;
11  };
12
13  int foo(){
14    struct Car c;
15    c.c_id = 0;
16    c.c_speed = 1.0;
17    c.c_location.c_latitue = 100.9;
18    c.c_location.c_longitude = 123.3;
19    return c.c_i;
20  }
```

输出

```
Error type 14 at Line 19: Non-existent field.
```

说明：第 19 行中，使用了未定义的域 c_i。这里可以多报一个 8 型错误

1.15 A-15

输入

```
1  int a;
2  float b;
3
4  struct Car {
5    int c_id;
6    float c_speed;
7    struct Location {
8      float c_longitude;
9      float c_latitue;
10   } c_location;
11
12   struct {
13     int c_x;
14     int c_y;
15   } c_location;
16 };
17
18 int foo(){
19   struct Car c;
20   c.c_id = 0;
21   c.c_speed = 1.0;
22   return 0;
23 }
```

输出

```
Error type 15 at Line 15: Redefined field.
```

说明：第 15 行中，c_location 与第 10 行重复。该错误可以报在第 10 行.

1.16 A-16

输入

```
struct ClassRoom {
  int cid;
  struct Position {
    int bid;
    int rid;
  } position;

  struct Teacher {
    int tid;
    int tgender;
    int course;
  } teacher;

  struct Student {
    int sid;
    int sgender;
    float grade;
  } students[100];

} classRoom1, classRoom2, classRoom3;

float ave_grade(struct ClassRoom cr){
  float sum = 0.0;
  int index = 0;
  while(index < 100){
    sum = sum + cr.students[index].grade;
  }
  return sum / 100.0;
}

struct Teacher {
  struct ClassRoom classRoom;
};
```

```

int main() {
  float ave1 = ave_grade(classRoom1);
  float ave2 = ave_grade(classRoom2);
  float ave3 = ave_grade(classRoom3);
}
```

输出

```
Error type 16 at Line 31: Redefined structure 'Teacher'.
```

说明：第 31 行中，定义的结构体 Teacher 和已经定义过的结构体重名了，也可以报在第 8 行。可以多报与 struct Teacher 相关的 17 型错误和 1 型错误。

1.17 A-17

输入

```
struct Node {
  int id;
  int next;
  int prev;
};

struct Edge edges[100];

int add_next(struct Node curNode, struct Node nextNode){
  curNode.next = nextNode.id;
  nextNode.prev = curNode.id;
}

int main(){
  struct Node node1, node2;
  add_next(node1, node2);
}
```

输出

```
Error type 17 at Line 7:  Undefined struct type 'Edge'
```

说明：第 7 行中，使用了未定义的结构体类型 Edge。

1.18 A-18

输入

```
struct Node{
  int id;
};

struct Edge {
  struct Node from;
  struct Node to;
} edges[100];


int main(){
  int a = 100;
  return edges + a;
}
```

输出

```
Error type 7 at Line 13: Bad type(s) for '+' operation.
```

说明：第 13 行中，数组变量不能和整型变量相加。可以多报一个 8 型错误。

1.19 A-19

输入

```
struct Point_int{
  int i_x;
  int i_y;
};
```

```c
5
6   struct Point_float{
7     float f_x;
8     float f_y;
9   };
10
11  int distance_int(struct Point_int ip1, struct Point_int ip2){
12    return (ip1.i_x - ip2.i_x) * (ip1.i_x - ip2.i_x) + (ip1.i_y - ip2.i_y) * (ip1.i_y - ip2.i_y);
13  }
14
15  float distance_float(struct Point_float fp1, struct Point_float fp2){
16    return (fp1.f_x - fp2.f_x) * (fp1.f_x - fp2.f_x) + (fp1.f_y - fp2.f_y) * (fp1.f_y - fp2.f_y);
17  }
18
19  int main(){
20    struct Point_int pi1, pi2;
21    struct Point_float pf1, pf2;
22    distance_float(pi1, pi2);
23    distance_float(pf1, pf2);
24  }
```

输出

```
Error type 9 at Line 22: Funtion args mismatch.
```

说明：第 22 行中，函数的实参类型与形参类型不匹配。

1.20 A-20

输入

```c
struct Student {
  int sid;
3      int age = 10;
4    };
5
6    struct Teacher {
7      int tid;
8      int course;
9    };
10
11   struct ClassRoom{
12     struct Student students[100];
13     struct Teacher teacher;
14   };
15
16   int main(){
17     struct ClassRoom classRoom1;
18     classRoom1.teacher.course = 10;
19   }
```

输出

```
Error type 15 at Line 3: Illegal use of assignment.
```

说明：第 3 行中，全局变量或结构体中域不能初始化。

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 Rectangle{
  float r_length = 100.0;
```

```
 3    float r_width;
 4  };
 5
 6  struct Triangle{
 7    float t_height;
 8    float t_width;
 9  };
10
11  float area_rec(struct Rectangle rec){
12    return rec.r_length * rec.r_width;
13  }
14
15  int area_tri(struct Triangle tri1){
16    return tri1.t_height * tri1.t_width;
17  }
18
19  float area_tri(struct Triangle tri2){
20    return tri2.t_height * tri2.t_width;
21  }
22
23  int main(){
24    struct Rectangle rectangles[100];
25    struct Triangle triangles[50];
26    float area;
27    area_rec(rectangles[0]) = area;
28  }
```

输出

```
Error type 15 at Line 2: Illegal use of assignment.
Error type 8 at Line 16: Return type mismatch.
Error type 4 at Line 19: Redifined function 'area_tri'.
Error type 6 at Line 27: LHS are a right-value-only Expression.
```

说明：第 2 行中，初始化了全局结构体中的域；在第 16 行中，函数实际返回值与定义的类型不匹配；在第 19 行中，重复定义了函数’area_tri’，也可报在第 15 行; 在 27 行中，赋值符号左侧是一个右值表达式，这里可以多报一个 5 型错误。

2.2 B-2

输入

```
1  struct Rectangle {
2    int tlx, tly;
3    int w, h;
4  };
5
6  struct Circle {
7    int cx, cy;
8    int cr;
9  };
10
11 struct Rectangle makeRect(int etlx, int etly, int ew, int eh) {
12   struct Rectangle erect;
13   erect.tlx = etlx;
14   erect.tly = etly;
15   erect.rw = ew;
16   erect.h = eh;
17   return erect;
18 }
19
20 struct Circle makeCirc(int fcx, int fcy, int fcr) {
21   struct Circle fcirc;
22   fcirc.cx = fcx;
23   fcirc.cy = fcy;
24   fcirc.cr = fcr;
25   return fcirc(12);
26 }
```

```c

int calArea(struct Rectangle arect) {
  return arect.w * arect.h;
}

int calArea(struct Circle bcirc) {
  return 3 * bcirc.cr * bcirc.cr;
}

int isRCover(struct Rectangle drect, int dx, int dy) {
  int dtop = drect.tly;
  int dleft = drect.tlx;
  int dbottom = dtop + drect.h;
  int dright = dleft + drect.w;

  if (dleft <= dx && dx <= dright) {
    if (dtop <= dy && dy <= dbottom) {
      return 1;
    }
  }

  return 0;
}

int main() {
  struct Rectangle mr = makeRect(1, 4, 32, 53);
  struct Circle mc = makeCirc(12.1, 21, 4.3);
  int mx = 12, my = mc.cx * mc.cy / mc.cr;
  return isRCover(mr, mx, my);
}
```

输出

```
Error type 14 at Line 15: Non-existent field "rw".
Error type 11 at Line 25: Function required but get "fcirc".
Error type 4 at Line 32: Redefined function "calArea".
Error type 9 at Line 53: Arguments types mismatch for function "
   makeCirc".
```

说明：第 15 行中，Rectangle 结构体中未定义域 rw, 此处可以多报一个 5 型错误。在第 25 行中，错误地对非数组变量使用了’[]’ 符号，此处可以多报一个 8 型错误；在第 32 行，重复定义了函数’calArea’; 在第 53 行中，函数的实参类型与形参类型不匹配。

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 Combination {
  int c_base;
  int c_num;
  int c_answer;
};

struct Permutation{
  int p_base;
  int p_num;
  int p_answer;
};

int factorial(int n){
  int f_sum = 1;
  int index = n;
  while(index > 1){
    f_sum = f_sum * index;
    index = index - 1;
```

```c
    }
  return f_sum;
}

int calculation_combination(struct Combination com){
  int cc_sum = 1;
  cc_sum = cc_sum * factorial(com.c_base);
  cc_sum = cc_sum / factorial(com.c_num);
  cc_sum = cc_sum / factorial(com.c_base - com.c_num);
  return cc_sum;
}


int calculation_permutation(struct Permutation per){
  int cp_sum = 1;
  cp_sum = cp_sum * factorial(per.p_base);
  cp_sum = cp_sum / factorial(per.p_base - per.p_answer);
  return cp_sum;
}

int main(){
  struct Combination com1;
  struct Permutation per1;
  per1.p_base = com1.c_base = 4;
  per1.p_num = com1.c_num = 2;
  com1.c_answer = calculation_combination(com1);
  per1.p_answer = calculation_permutation(per1);
  return 0;
}
```

输出

```
// 正常返回，没有任何输出
```

3.2 C-2

输入

```c
struct ArithmeticalSequence{
  int as_a0;
  int as_d;
  int as_num;
};

struct GeometricSequence{
  int gs_a0;
  int gs_q;
  int gs_num;
};


int arithmeticalSequenceSum(struct ArithmeticalSequence as){
  int as_last = as.as_a0 + (as.as_num - 1) * as.as_d;
  return (as.as_a0 + as_last) * as.as_num / 2;
}

int geometricSequenceSum(struct GeometricSequence gs){
  int gs_sum = gs.gs_a0;
  int gs_temp = gs.gs_a0;
  int gs_index = 1;
  while(gs_index < gs.gs_num){
    gs_index = gs_index + 1;
    gs_temp = gs_temp * gs.gs_q;
    gs_sum = gs_sum + gs_temp;
  }
  return gs_sum;
}

```

```
int main(){
  struct ArithmeticalSequence as1;
  struct GeometricSequence gs1;
  int sum = arithmeticalSequenceSum(as1) + geometricSequenceSum(gs1
     );
  return sum;
}
```

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 Combination {
  int c_base;
  int c_num;
  int c_answer;
};

struct Permutation{
  int p_base;
  int p_num;
  int p_answer;
};

int factorial(int n);

```

```c
int factorial(int n){
  int f_sum = 1;
  int index = n;
  while(index > 1){
    f_sum = f_sum * index;
    index = index - 1;
  }
  return f_sum;
}

int calculation_combination(struct Combination com){
  int cc_sum = 1;
  cc_sum = cc_sum * factorial(com.c_base);
  cc_sum = cc_sum / factorial(com.c_num);
  cc_sum = cc_sum / factorial(com.c_base - com.c_num);
  return cc_sum;
}


int calculation_permutation(struct Permutation per){
  int cp_sum = 1;
  cp_sum = cp_sum * factorial(per.p_base);
  cp_sum = cp_sum / factorial(per.p_base - per.p_answer);
  return cp_sum;
}

int calculation_combination(struct Combination com);


int main(){
  struct Combination com1;
  struct Permutation per1;
47    per1.p_base = com1.c_base = 4;
48    per1.p_num = com1.c_num = 2;
49    com1.c_answer = calculation_combination(com1);
50    per1.p_answer = calculation_permutation(per1);
51    return 0;
52  }
53
54  int calculation_permutation(struct Permutation per);
```

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 13，41，54 行报语法错误。

4.2 D-2

输入

```
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];
};

```

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

struct Result {
    int   rType;
    int   valid;
    int   iRes;
    float fRes;
};

int initArith() {
    O_ADD = 0;
    O_PRD = 1;
    O_SUB = 2;
    O_DIV = 3;
    T_INT = 4;
    T_FLT = 5;
    MLEN  = 100;
    return 0;
}

int cnt;

struct Result binOperator(struct Operation operation, struct
   BinData binData) {
    struct Result result;
    result.valid = 1;
    if (operation.opt == O_ADD) {
        result.iRes = binData.bdIData[0] + binData.bdIData[1];
        result.fRes = binData.bdFData[0] + binData.bdFData[1];
    } else if (operation.opt == O_PRD) {
        result.iRes = binData.bdIData[0] * binData.bdIData[1];
        result.fRes = binData.bdFData[0] * binData.bdFData[1];
    } else if (operation.opt == O_SUB) {
        result.iRes = binData.bdIData[0] - binData.bdIData[1];
        result.fRes = binData.bdFData[0] - binData.bdFData[1];
    } else if (operation.opt == O_DIV) {
        result.iRes = binData.bdIData[0] / binData.bdIData[1];
        result.fRes = binData.bdFData[0] / binData.bdFData[1];
    } else {
        result.valid = 0;
    }
    result.valid = result.valid && (operation.oType == binData.
       bdType);
    result.rType = operation.oType;
    return result;
}

struct Result mulOperation(struct Operation operation, struct
   MulData mulData) {
    struct Result result;
    int cnt = 0;
    result.valid = 1;
    if (operation.opt == O_ADD) {
        result.iRes = 0;
        result.fRes = 0.0;
    } else if (operation.opt == O_PRD) {
        result.iRes = 1;
        result.fRes = 1.0;
    } else {
        result.valid = 0;
    }
    while (cnt < MLEN) {
        if (operation.opt == O_ADD) {
            result.iRes = result.iRes + mulData.mdIData[cnt];
            result.fRes = result.fRes + mulData.mdFData[cnt];
        } else if (operation.opt == O_PRD) {
            result.iRes = result.iRes * mulData.mdIData[cnt];
            result.fRes = result.fRes * mulData.mdFData[cnt];
        }
        cnt = cnt + 1;
    }
    result.valid = result.valid && (operation.oType == mulData.
       mdType);
    result.rType = operation.oType;
    return result;
}
```

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 operation，result，cnt 的重复定义。

4.3 D-3

输入

```
struct S1 {
  int a1,b1;
  float c1,d1;
  int iarray1[100];
  float farray1[50];
  struct {
    int aa1,bb1;
```

```c
    float cc1, dd1;
  } sd1;
  struct SS1 {
    int ssiarray1[100];
  } ss1[100];
};


struct S2 {
  int a2,b2;
  float c2,d2;
  int iarray2[100];
  float farray2[50];
  struct {
    int aa2,bb2;
    float cc2, dd2;
  } sd2;
  struct SS2 {
    int ssiarray2[100];
  } ss2[100];
};

int compare(struct SS1 tss1, struct SS1 tss2){
  int i = 0;
  while(i < 100){
    if(tss1.ssiarray1[i] != tss2.ssiarray1[i]){
      return 0;
    }
    i = i + 1;
  }
  return 1;
}
```

```c

int equal(struct S1 ts1, struct S1 ts2){
  int index;
  int j;
  if(ts1.a1 != ts2.a1 || ts1.b1 != ts2.b1){
    return 0;
  }

  index = 0;
  while(index < 100){
    if(ts1.iarray1[index] != ts2.iarray1[index]){
      return 0;
    }
    index = index + 1;
  }

  if(ts1.sd1.aa1 != ts2.sd1.aa1 || ts1.sd1.bb1 != ts2.sd1.bb1){
    return 0;
  }

  index = 0;
  while(index < 100){
    if(compare(ts1.ss1[index], ts2.ss1[index]) == 0){
      return 0;
    }
    index = index + 1;
  }

  return 1;
}

int main() {
```

```
72    struct S1 myS1;
73    struct S2 myS2;
74    equal(myS1, myS2);
75  }
```

输出

```
// 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 74 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 E-1

这组测试用例针对 2.1 分组的同学。

输入

```
struct Combination {
  int c_base;
  int c_num;
  int c_answer;
};

struct Permutation{
  int p_base;
  int p_num;
  int p_answer;
};

int factorial(int n);

int factorial(int n){
  int f_sum = 1;
  int index = n;
  while(index > 1){
    f_sum = f_sum * index;
    index = index - 1;
  }
  return f_sum;
}

int calculation_combination(struct Combination com){
  int cc_sum = 1;
  cc_sum = cc_sum * factorial(com.c_base);
  cc_sum = cc_sum / factorial(com.c_num);
  cc_sum = cc_sum / factorial(com.c_base - com.c_num);
  return cc_sum;
}


int calculation_permutation(struct Permutation per){
  int cp_sum = 1;
  cp_sum = cp_sum * factorial(per.p_base);
  cp_sum = cp_sum / factorial(per.p_base - per.p_answer);
  return cp_sum;
}

int calculation_combination(struct Combination com);


int main(){
  struct Combination com1;
  struct Permutation per1;
  per1.p_base = com1.c_base = 4;
  per1.p_num = com1.c_num = 2;
  com1.c_answer = calculation_combination(com1);
  per1.p_answer = calculation_permutation(per1);
  return 0;
}

int calculation_permutation(struct Permutation per);

int calculation(struct Combination com, struct Permutation per);

int calculation_combination(struct Permutation per);
```

输出（基础班）

```
Error type 18 at Line 56: Undefined function "calculation".
```

输出（普通班）

```
Error type 19 at Line 56: Undefined function "calculation".
Error type 19 at Line 58: Inconsistent declaration of function "
    calculation_combination".
```

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。

5.2 E-2

这组测试用例针对 2.2 分组的同学。

输入

```
struct Combination {
  int c_base;
  int c_num;
  int c_answer;
};

struct Permutation{
  int p_base;
  int p_num;
```

```c
  int p_answer;
};

int factorial(int n){
  int sum = 1;
  int index = n;
  int tempArray;
  tempArray[0] = 1;
  while(index > 1){
    sum = sum * index;
    index = index - 1;
  }
  return sum;
}

int calculation_combination(struct Combination com){
  int sum = 1;
  sum = sum * factorial(com.c_base) * 1.0;
  sum = sum / factorial(com.c_num);
  sum = sum / factorial(com.c_base - com.c_num);
  return sum;
}


int calculation_permutation(struct Permutation per){
  int sum = 1;
  sum = sum * factorial(per.p_base);
  sum = sum / factorial(per.p_base - per.p_answer);
  return sum;
}

int main(){
  struct Combination com1;
  struct Permutation per1;
  per1.p_base = com1.c_base = 4;
  per1.p_num = com1.c_num = 2;
  calculation_combination(com1) = com1.c_answer;
  per1.p_answer = calculation_permutation(per1);
  return 0;
}
```

输出

```
Error type 10 at Line 17: Apply [] to non-array variable.
Error type 7 at Line 27: Bad type(s) for operation.
Error type 6 at Line 46: Invalid left value.
```

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。17 行可以多报一个 5 型错误。

5.3 E-3

这组测试用例针对 2.3 分组的同学。

输入

```
struct Combinations {
  struct Combination{
    int c_base;
    int c_num;
  } combinations[10];
  int c_answer = 100;
};

struct Permutations {
  int p_answer;
  struct Permutation{
    int p_base;
    int p_num;
```

```c
  } permutations[10];
};


int factorial(int n){
  int f_sum = 1;
  int f_index = n;
  while(f_index > 1){
    f_sum = f_sum * f_index;
    f_index = f_index - 1;
  }
  return f_sum;
}

int calculation_combination(struct Combination com){
  int cc_sum = 1.0;
  cc_sum = cc_sum * factorial(com.c_base);
  cc_sum = cc_sum / factorial(com.c_num);
  cc_sum = cc_sum / factorial(com.c_base - com.c_num);
  return cc_sum;
}


int calculation_permutation(struct Permutation per){
  int cp_sum = 1;
  cp_sum = cp_sum * factorial(per.p_base);
  cp_sum = cp_sum / factorial(per.p_base - per.p_num);
  return cp_sum;
}

int calculation(struct Combinations coms){
  int c_index = 0;
  int c_sum = 0;
  while(c_index < 10){
    c_sum = c_sum + calculation_combination(coms.combinations[
        c_index]);
    c_index = c_index + 1;
  }
  return c_sum;
}

int main(){
  struct Combinations com1;
  struct Permutations per1;
  com1.c_answer = calculation(com1);
  per1.p_answer = calculation(per1);
  return 0;
}
```

输出

```
Error type 15 at Line 6: Illegal use of assignment.
Error type 5 at Line 29: Type mismatched for assignment.
Error type 9 at Line 58: Arguments types mismatch for Function "
    calculation(Combinations)".
```

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，可以写邮件与张灵毓助教联系，注意同时抄送给许老师。