

编译原理第二次实验测试用例：目录

1	A 组测试用例	3
1.1	A-1	3
1.2	A-2	3
1.3	A-3	4
1.4	A-4	5
1.5	A-5	6
1.6	A-6	7
1.7	A-7	8
1.8	A-8	9
1.9	A-9	10
1.10	A-10	11
1.11	A-11	12
1.12	A-12	13
1.13	A-13	14
1.14	A-14	15
1.15	A-15	16
1.16	A-16	16
1.17	A-17	17
1.18	A-18	18
1.19	A-19	19
1.20	A-20	20
2	B 组测试用例	21
2.1	B-1	21
2.2	B-2	23
3	C 组测试用例	24
3.1	C-1	25
3.2	C-2	26

4	D 组测试用例	28
4.1	D-1	28
4.2	D-2	29
4.3	D-3	31
5	E 组测试用例	33
5.1	E-1	33
5.2	E-2	34
5.3	E-3	35
6	结束语	37

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 15, 9, 7。每个测试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”类型必须报出来，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 Product {
2     int name;
3     float price;
4 };
5
6 int main(){
7     struct Product burger, fries;
8     burger.name = 1;
9     fries.price = 1.3;
10    n = burger.name - 1;
11    return 0;
12 }
```

输出

```
1 Error type 1 at line 10: Variable undefined "n".
```

说明：n = burger.name - 1 这一句包含未定义的变量 n，这里也可另外报出错误类型 5（= 两边类型不匹配）。

1.2 A-2

输入

```
1 int compare(int a, int b) {
2     if(a == b)
```

```

3         return 0;
4     if(a < b)
5         return -1;
6     if(a > b)
7         return 1;
8 }
9
10 int main(){
11     int x,y,z;
12     x = 1;
13     y = 2;
14     z = 3;
15     compare(x,y);
16     campare(y,z);
17     return z;
18 }

```

输出

```

1 Error type 2 at line 16: Function undefined "campare".

```

说明: campare 函数未定义。

1.3 A-3

输入

```

1 struct FastFood {
2     int yum;
3     float price;
4 };
5
6 struct Burger {
7     int juice;
8     float bucks;
9 };

```

```

10
11 float buyAFood(float money) {
12     struct FastFood cheap;
13     struct Burger expensive;
14     float realMoney, Burger;
15     cheap.price = 1.3;
16     cheap.yum = 3;
17     expensive.bucks = 3.4;
18     if(cheap.yum == 4) {
19         realMoney = money - cheap.price;
20         return realMoney;
21     } else {
22         Burger = money - expensive.bucks;
23         return Burger;
24     }
25 }

```

输出

```

1 Error type 3 at line 14: Variable redefined "Burger".

```

说明：重复定义的变量 **Burger**，这里如果错误位置写为第一行也算对。可能会在 22、23 行报出错误类型 1、5、8。

1.4 A-4

输入

```

1 int compare(int a, int b) {
2     if (a == b)
3         return 0;
4     if (a < b)
5         return -1;
6     else
7         return 1;
8 }

```

```

9
10 int compare(float x, float y) {
11     return -2;
12 }
13
14 int main(){
15     compare(1,2);
16     compare(1.2,3.4);
17     return 0;
18 }

```

输出

```

1 Error type 4 at line 10: Function redefined "compare".

```

说明：重复定义的函数 `compare`。可能同时会在第 15、16 行报错误类型 2 或 16。

1.5 A-5

输入

```

1 struct Burger{
2     int meat;
3     int bread;
4     float saurce;
5 }McDonald;
6
7 struct Whopper{
8     int meat1;
9     int meat2;
10    int bread1;
11    int saurce1;
12 }BurgerKing;
13
14 int main(){
15     McDonald.meat = -1;

```

```

16     McDonald.bread = 1;
17     McDonald.saurce = 3.4;
18     BurgerKing.meat1 = 3;
19     BurgerKing.meat2 = 3;
20     BurgerKing.bread1 = McDonald.bread;
21     BurgerKing.saurcel = McDonald.saurce;
22     return 3;
23 }

```

输出

```

1 Error type 5 at line 21: Type mismatched for assignment.

```

说明：赋值号两边不匹配（float 赋值给 int）。

1.6 A-6

输入

```

1 struct Burger{
2     int meat;
3     int bread;
4     float saurce;
5 }McDonald;
6
7 struct Whopper{
8     int meat1;
9     int meat2;
10    int bread1;
11    int saurcel;
12 }BurgerKing;
13
14 int theSame(int a, int b) {
15     if(a ==b )
16         return 1;
17     else

```

```

18         return 0;
19     }
20
21     int main() {
22         int p;
23         McDonald.meat = -1;
24         McDonald.bread = 1;
25         McDonald.saurce = 3.4;
26         BurgerKing.meat1 = 3;
27         BurgerKing.meat2 = 3;
28         p = theSame(McDonald.meat, BurgerKing.meat1);
29         theSame(McDonald.meat, BurgerKing.meat2) = p;
30         return p;
31     }

```

输出

```

1 Error type 6 at line 29: The left-hand side of an assignment must be
  a variable.

```

说明：赋值号左边是一个不能为左值的类型（函数）。

1.7 A-7

输入

```

1 struct Burger{
2     int meat;
3     int bread;
4     float saurce;
5 }McDonald;
6
7 struct Whopper{
8     int meat1;
9     int meat2;
10    int bread1;

```



```

11         int saurcel;
12     }BurgerKing;
13
14     int main(){
15         int meats;
16         float burgers;
17         McDonald.meat = -1;
18         McDonald.bread = 1;
19         McDonald.saurce = 3.4;
20         BurgerKing.meat1 = 3;
21         BurgerKing.meat2 = 3;
22         BurgerKing.bread1 = McDonald.bread;
23         burgers = McDonald.saurce * BurgerKing.saurcel;
24         meats = McDonald.meat * BurgerKing.meat1;
25         return meats;
26     }

```

输出

```

1 Error type 7 at line 23: Operands type mismatched.

```

说明：乘号操作符两边类型不匹配，这里可以另外报错误类型 5，必须在 23 行。

1.8 A-8

输入

```

1 struct Fries{
2     int number;
3     float flavor;
4 };
5
6 int tasty(){
7     struct Fries McDonald;
8     McDonald.number = 30;
9     McDonald.flavor = -1.2;

```

```

10     while(McDonald.number > 13) {
11         McDonald.flavor = McDonald.flavor + 1.2;
12         McDonald.number = McDonald.number - 3;
13     }
14
15     if(McDonald.number <= 24) {
16         return McDonald.number;
17     } else{
18         return McDonald.flavor;
19     }
20 }

```

输出

```

1 Error type 8 at line 18: Return type mismatched.

```

说明：返回值类型与函数定义不一致，也可以报在第 6 行。

1.9 A-9

输入

```

1 struct Furiosa{
2     int arm;
3     int hair;
4 }theFuriosa;
5
6 struct AquaCola{
7     float amount;
8     struct Furiosa transporter;
9 }cola;
10
11 float transport(struct Furiosa theTransporter, struct AquaCola
12     transportee, int requiredArm) {
13     if (theTransporter.arm == requiredArm) {
14         return transportee.amount;
15     }
16 }

```

```

14         } else
15             return -1.0;
16     }
17
18     int main() {
19         int fullArm;
20         float receivedAmount;
21
22         cola.transporter = theFuriosa;
23         theFuriosa.arm = 1;
24         fullArm = 2;
25         receivedAmount = transport(theFuriosa, cola.transporter,
26                                   fullArm);
27         return 3;
28     }

```

输出

```

1 Error type 9 at Line 25: Function 'transport' is not applicable for
   arguments.

```

说明：函数实参与形参类型不一致。

1.10 A-10

输入

```

1 struct MilkShake{
2     float milk;
3     float cream;
4 }rabbit[10];
5
6 struct Spirit{
7     int bottle;
8     float amount;
9 }dog;

```

```

10
11 int main() {
12     int t;
13     t = 3;
14     while(t < 10) {
15         rabbit[t].milk = 2.3;
16         dog[t].bottle = 2;
17         t = t + t;
18     }
19     return t;
20 }

```

输出

```

1 Error type 10 at line 16: Variable "dog" is not an array.

```

说明：对非数组变量使用 [] 操作符，这里可能会连带报出错误类型 5.

1.11 A-11

输入

```

1 int gcd(int x, int y){
2     if(x == 0) return y;
3     return gcd(y - x, x);
4 }
5
6 int main() {
7     int gdd, i, N, sum;
8     gdd = 14;
9     N = 12;
10    i = 0;
11    sum = 0;
12    while(i < N) {
13        i = i+ 1;
14        sum = sum + gcd(gdd, i);

```

```

15     }
16     return sum + gdd(gdd, N * 2);
17 }

```

输出

```

1 Error type 11 at line 16: Variable "gdd" is not a function.

```

说明：对非函数的标识符使用 () 操作符。可能会连带报出错误类型 8。

1.12 A-12

输入

```

1 struct Burger{
2     int meat;
3     float flavor;
4 }burgers[10];
5
6 int exchange(struct Burger a, struct Burger b) {
7     int tem;
8     tem = a.meat;
9     a.meat = b.meat;
10    b.meat = tem;
11    return tem;
12 }
13
14 int main() {
15     int bb, N;
16     float t;
17     N = 10;
18     bb = 0;
19     t = 1.0;
20
21     while(bb < N) {
22         bb = bb +2;

```

```

23         burgers[bb].meat = N + 3;
24         burgers[bb-1].meat = N + 2;
25     }
26     N = N - 1 ;
27     while (N > 0) {
28         bb = exchange(burgers[N], burgers[N - 1]);
29         burgers[N].flavor = 3.0;
30         N = N - 1;
31     }
32     burgers[burgers[N + 1].flavor].meat = bb;
33     return bb;
34 }

```

输出

```

1 Error type 12 at line 32: Array index is not integer.

```

说明：数组下标非整数。

1.13 A-13

输入

```

1 struct Sith{
2     int force;
3     int darkness;
4 }sithLords[10];
5
6 int jedi[10];
7
8 int battle(){
9     int results[10];
10    int i = 0;
11    while(i < 10){
12        if(sithLords[i].force >= jedi[i].force)
13            results[i] = sithLords[i].darkness;

```

```

14         else
15             results[i] = -1;
16     }
17     return i;
18 }

```

输出

```

1 Error type 13 at Line 12: Illegal use of "."

```

说明：对非结构体变量使用“.”操作符，同时可以报出错误类型 5。

1.14 A-14

输入

```

1 struct Sith{
2     int force;
3     int darkness;
4 }vader;
5
6 struct Jedi{
7     int power;
8     float calm;
9 }yoda;
10
11 int battle() {
12     int result;
13     if (vader.force >= yoda.power)
14         result = vader.darkness;
15     else if (vader.force <= 3)
16         result = vader.force;
17     else
18         result = yoda.darkness;
19     return result;
20 }

```

输出

```
1 Error type 14 at Line 18: Non-existent field 'darkness'.
```

说明：使用了结构体中未定义的域 `darkness`，这里可以报出错误类型 5 和 7。

1.15 A-15

输入

```
1 struct Jedi{
2     int force, power;
3     float might;
4     float force;
5 };
6
7 int main() {
8     struct Jedi obiwon;
9     return obiwon.force;
10 }
```

输出

```
1 Error type 15 at line 4: Field redefined "force".
```

说明：结构体内部有重复定义的域。第 8 行和第 9 行可能会报 Jedi 未定义，不影响得分。

1.16 A-16

输入

```
1 struct Soldier{
2     int name;
3     int power;
4 }platoon[10];
5
6 struct General{
7     int names;
8     struct Soldier soldiers[10];
```



```

9  }McMiller;
10
11
12  int main(){
13      struct McMiller{
14          struct General man;
15      }one;
16      return 0;
17  }

```

输出

```

1  Error type 16 at line 13: Duplicated name "McMiller".

```

说明：结构体的名字域之前定义的变量名重复。

1.17 A-17

输入

```

1  struct Movie{
2      int title;
3      float rate;
4  }django;
5
6  struct French{
7      int male;
8      int age;
9  }sophi;
10
11 struct Drink{
12     int name;
13     float amount;
14 };
15
16 int main(){

```

```

17     struct Movie inglorious = django;
18     struct French marso = sophi;
19     struct Drunk cola;
20     return marso.age;
21 }

```

输出

```

1 Error type 17 at line 19: Structure Undefined "Drunk".

```

说明：使用了未定义的结构体 Drunk。

1.18 A-18

输入

```

1 struct Food{
2     int name;
3     float price;
4     struct Content{
5         int title;
6         float type;
7     }types;
8 }ham;
9
10 struct Burger{
11     struct Food meat = ham;
12     int buck;
13 };
14
15 int main() {
16     struct Burger hamburger;
17     return hamburger.buck;
18 }

```

输出

1 Error type 15 at line 11: Initialize a field of structure "meat".

说明：结构体定义时对域进行初始化。

1.19 A-19

输入

```
1 struct food{
2     int name;
3     float price;
4 }ham, burger;
5
6 int tem;
7 float tem1;
8
9 int swap(struct food a, struct food b, int type){
10     if(type == 1) {
11         tem = a.name;
12         a.name = b.name;
13         b.name = tem;
14     } else {
15         tem1 = a.price;
16         a.price = b.price;
17         b.price = tem1;
18     }
19     return 1;
20 }
21
22 int main() {
23     swap(ham,burger);
24     return 0;
25 }
```

输出

1 Error type 9 at line 23: Argument number mismatched.

说明：函数调用时实参与形参数目不匹配。

1.20 A-20

输入

```
1 int a[20][20];
2
3 struct Type{
4     int name[20];
5     int hair[20];
6 };
7
8 int main() {
9     int i, N, sum;
10    struct Type t;
11    i = 0;
12    N = 10;
13    sum = 0;
14    while(i < N){
15        t.name[i] = a[i][i];
16        t.hair[i] = a[i][N - i - 1];
17        i = i + 1;
18    }
19    while(i >= 0){
20        sum = sum + t.name[i] * t.hair;
21        i = i - 1;
22    }
23    return sum;
24 }
```

输出

```
1 Error type 7 at line 20: Operands type mismatched.
```

说明：操作数类型与操作符不匹配（乘号右边为数组）。

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 Dog{
2     int name,height,kind,owner;
3     float beauty;
4 };
5
6 struct DogHouse{
7     int address;
8     float strong = 2.3;
9     struct Dog dogs[10];
10 };
11
12 struct DogHouse putDogInHouse(struct DogHouse house[10], struct Dog
    wilds[100]){
13     int i,j;
14     i = 0;
15     while(i < 10) {
16         j = i * 10;
17         while (j < (i+1)*10){
18             house[i].dogs[j] = wilds[i*10 + j];
19             j = j;
20     }
```

```

21         i = i + 1;
22     }
23     return house[0];
24 }
25
26 struct Dog letDogOut(struct DogHouse broken[10], int number, int
    brokenAddress, float looseBeauty) {
27     struct Dog de;
28     int x, y;
29     x = 0;
30     while(x < 10) {
31         if (broken[x] == brokenAddress) {
32             de = broken[x].dogs[number];
33             de = de.height;
34         }
35         x = x + 1;
36     }
37     de.beauty = de.beauty - looseBeauty;
38     return de;
39 }
40
41 float main() {
42     struct DogHouse fullHouse[10];
43     struct Dog carols[100];
44     struct DogHouse firstHouse;
45     struct Dog looseDog;
46     firstHouse = putDogInHouse(fullHouse, carols);
47     looseDog = letDogout(fullHouse, 3, 412, 0.4);
48     return looseDog.beauty;
49 }

```

输出

```

1 Error type 15 at line 8: Initialize a field of structure "strong".

```

```
2 Error type 7 at line 31: Operands type mismatched.
3 Error type 5 at line 33: Type mismatched for assignment.
4 Error type 2 at line 47: Function undefined "letDogout".
```

说明：输出中的 4 个错误为本质错误，是必须要报出来的，这些错误可能会有连锁反应，合理即可。

2.2 B-2

输入

```
1 struct Defined{
2     int name;
3     float ty;
4 };
5
6 int sequence[1000];
7 int quickSort(int start, int end) {
8     int i,j, x;
9     i = start;
10    j = end;
11    x = sequence[i];
12    if(i < j) {
13        while(i < j) {
14            while(i < j && sequence[j] > x)
15                j = j - 1;
16            if(i < j)
17                sequence[i] = sequence[j];
18            while(i < j && sequence[i] <= x)
19                i = i + 1;
20            if(i < j)
21                sequence[j] = sequence[i];
22        }
23        sequence[i] = x;
24        quickSort(start, i, -1);
```

```

25         quickSort(i+1, end);
26     }
27     return 1;
28 }
29
30 int main() {
31     struct NotDefined test;
32     struct Defined ttt;
33     int q,p;
34     q = 0;
35     p = 3;
36     ttt.nme = q;
37     while(q < 1000) {
38         sequence[q] = p * 3;
39         p = p - q;
40         q = q + 1;
41     }
42     quickSort(0,999);
43     q(0,999);
44     return q;
45 }

```

输出

```

1 Error type 9 at line 24: Argument number mismatched.
2 Error type 17 at line 31: Structure Undefined "NotDefined".
3 Error type 14 at line 36: Structure field undefined "nme".
4 Error type 11 at line 43: Variable "q" is not a function.

```

说明：输出中的 4 个错误为本质错误，是必须要报出来的，这些错误可能会有连锁反应，合理即可。

3 C 组测试用例

本组测试用例共 2 个，不包含任何错误，不需要任何输出。

3.1 C-1

输入

```
1 struct Dog{
2     int name,height,kind,owner;
3     float beauty;
4 };
5
6 struct DogHouse{
7     int address;
8     float strong;
9     struct Dog dogs[10];
10 };
11
12 struct DogHouse putDogInHouse(struct DogHouse house[10], struct Dog
    wilds[100]){
13     int i,j;
14     i = 0;
15     while(i < 10) {
16         j = i * 10;
17         while (j < (i+1)*10){
18             house[i].dogs[j] = wilds[i*10 + j];
19             j = j;
20         }
21         i = i + 1;
22     }
23     return house[0];
24 }
25
26 struct Dog letDogOut(struct DogHouse broken[10], int number, int
    brokenAddress, float looseBeauty) {
27     struct Dog de;
28     int x, y;
```

```

29     x = 0;
30     while(x < 10) {
31         if (broken[x].address == brokenAddress) {
32             de = broken[x].dogs[number];
33         }
34         x = x + 1;
35     }
36     de.beauty = de.beauty - looseBeauty;
37     return de;
38 }
39
40 float main() {
41     struct DogHouse fullHouse[10];
42     struct Dog carols[100];
43     struct DogHouse firstHouse;
44     struct Dog looseDog;
45     firstHouse = putDogInHouse(fullHouse, carols);
46     looseDog = letDogOut(fullHouse, 3, 412, 0.4);
47     return looseDog.beauty;
48 }

```

输出

```

1 // 正常返回，没有任何输出。

```

说明：测试用例 B_1 的正确版。

3.2 C-2

输入

```

1 struct Defined{
2     int name;
3     float ty;
4 };
5

```

```

6  int sequence[1000];
7  int quickSort(int start, int end) {
8      int i,j, x;
9      i = start;
10     j = end;
11     x = sequence[i];
12     if(i < j) {
13         while(i < j) {
14             while(i < j && sequence[j] > x)
15                 j = j - 1;
16             if(i < j)
17                 sequence[i] = sequence[j];
18             while(i < j && sequence[i] <= x)
19                 i = i + 1;
20             if(i < j)
21                 sequence[j] = sequence[i];
22         }
23         sequence[i] = x;
24         quickSort(start, i-1);
25         quickSort(i+1, end);
26     }
27     return 1;
28 }
29
30 int main() {
31     struct Defined ttt;
32     int q,p;
33     q = 0;
34     p = 3;
35     while(q < 1000) {
36         sequence[q] = p * 3;
37         p = p - q;

```

```

38         q = q + 1;
39     }
40     quickSort(0, 999);
41     return q;
42 }

```

输出

```

1 // 正常返回，没有任何输出。

```

说明：测试用例 B_2 的正确版。

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

```

1 int words[100];
2
3 int defineFirst(int a, int b);
4
5 int typeOut(int word, int position){
6     int results;
7     if(position < 100 && 0 >= position){
8         words[position] = word;
9         results = 1;
10    }
11    if(position >= 100) {
12        words[99] = word;
13        results = 2;
14    }
15    if (position < 0) {

```

```

16         words[0] = word;
17         results = -1;
18     }
19     return results;
20 }
21
22 int defineFirst(int a, int b){
23     return a + b;
24 }
25
26 int main() {
27     int i, N;
28     i = 0;
29     N = 3;
30     while (typeOut(N,i) == 1) {
31         N = -(N - 3);
32         i = i + N + 2;
33     }
34     return N;
35 }

```

输出

```

1 // 正常返回， 没有任何输出。

```

说明：对于 2.1 分组的同学，应该没有任何输出，对于其他分组的同学，应该在第 3 行报出有语法错误（Error type B at line 3）。

4.2 D-2

输入

```

1 int sequences[100][100];
2
3 int addEach(int p) {
4     int i,j;

```

```

5      i = 0;
6      j = 0;
7      while(i < 100){
8          while(j < 100){
9              sequences[i][j] = sequences[i][j] + p;
10             j = j + 1;
11         }
12         i = i + 1;
13     }
14     return p;
15 }
16
17 int mulEach(int p) {
18     int i, j;
19     i = 0;
20     j = 0;
21     while(i < 100){
22         while(j < 100){
23             sequences[i][j] = sequences[i][j] * p;
24             j = j + 1;
25         }
26         i = i + 1;
27     }
28     return p;
29 }
30
31 int main(){
32     int i, j;
33     i = 0;
34     j = 0;
35     while(i < 100){
36         while(j < 100){

```

```

37         sequences[i][j] = i * j;
38         j = j + 1;
39     }
40     i = i + 1;
41 }
42 addEach(i + 3);
43 mulEach(j - 4);
44 return i + j;
45 }

```

输出

```

1 // 正常返回， 没有任何输出。

```

说明：对于 2.2 分组的同学，应该没有任何输出，对于其他分组的同学，应该在第 18 和 32 行报出有语义错误。

4.3 D-3

输入

```

1 struct A{
2     int a;
3     float a_float;
4     struct A_inner{
5         int a_inner_int[20];
6     }a_inner;
7 }aa;
8
9 struct B{
10     int b;
11     float b_float;
12     int b_array[10][3];
13 }bb;
14
15 struct C{

```

```

16     int c;
17     float c_float;
18     int c_array[10][3];
19 }cc;
20
21 struct D{
22     int d;
23     float d_float;
24     struct D_inner{
25         int d_innerr[20];
26     }d_inner;
27 }dd;
28
29 int main(){
30     struct A temA, temA1;
31     struct B temB, temB1;
32     struct C temC, temC1;
33     struct D temD, temD1;
34     temA = aa;
35     temA1 = dd;
36     temB = bb;
37     temB1 = cc;
38     temC = cc;
39     temC1 = bb;
40     temD = dd;
41     temD1 = aa;
42     return 1;
43 }

```

输出

```

1 // 正常返回， 没有任何输出。

```

说明：对于 2.3 分组的同学，应该没有任何输出，对于其他分组的同学，应该在第 35、37、39

和 41 行识别出类型不匹配（函数参数类型 Error type 5）。

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 分组同学。输入

```
1 struct Burger{
2     int meat;
3     float saurce;
4     int bread;
5 }king;
6
7 int addMeat(int addedAmount);
8 int addMeat(int addedAmount, struct Burger addedBurger);
9
10 int addMeat(int addedAmount){
11     king.meat = king.meat + addedAmount;
12     return 0;
13 }
14
15 float addSaurce(float addedSaurce);
16
17 int addSaurce(float addedSaurce) {
18     king.saurce = addedSaurce;
19     return 0;
20 }
21
22 int addBread(int breads, struct Burger bb);
```

输出

```
1 Errot type 19 at Line 8: Conflict between function declarations.
```

```
2 Error type 19 at Line 17: Conflict between function declarations.
3 Error type 18 at Line 22: Undefined but declared function.
```

说明：仅 2.1 分组同学需要测试该用例，需要输出上述错误信息。

5.2 E-2

这组测试用例针对 2.2 分组同学。输入

```
1 struct Node{
2     int value;
3 }start[200];
4
5 struct Node linkNode(struct Node currentNode, struct Node newNode){
6     int i,j;
7     struct Node current = start[0];
8     i = 0;
9     while(current.value != currentNode.value) {
10         int j = i;
11         j = j + 1;
12         current = start[j];
13     }
14     newNode.value = newNode.value + i + p;
15     return start[i];
16 }
17
18 struct Node newOne(int value){
19     struct Node nn;
20     int x = value;
21     int t = 2;
22     float result = 1.5;
23     while(t < 3) {
24         float x = 1.2;
25         result = result * x;
26         t = t + 1;
```

```

27     }
28     nn.value = x;
29     return nn;
30 }
31
32 int main() {
33     int tem,i,j;
34     float tem, result;
35     i = 0;
36     while(i <= 10){
37         start = linkNode(start[0], newOne(i));
38         i = i + 1;
39     }
40     return i;
41 }

```

输出

```

1 Error type 1 at Line 14: Undefined variable 'p'.
2 Error type 4 at Line 34: Redefined variable "tem".

```

说明：仅 2.2 分组同学需要测试该用例，需要输出上述错误信息。

5.3 E-3

这组测试用例针对 2.3 分组同学。输入

```

1 struct A{
2     int a;
3     float a_float;
4     struct A_inner{
5         int a_inner_int[20];
6     }a_inner;
7 }aa;
8
9 struct B{

```

```

10         int b;
11         float b_float;
12         int b_array[10][3];
13     }bb;
14
15     struct C{
16         int c;
17         float c_float;
18         int c_array[5];
19     }cc;
20
21     struct D{
22         int d;
23         struct D_inner{
24             int d_innerr[15];
25         }d_inner;
26     }dd;
27
28     int main(){
29         struct A temA, temA1;
30         struct B temB, temB1;
31         struct C temC, temC1;
32         struct D temD, temD1;
33
34         temA = aa;
35         temA1 = dd;
36         temB = bb;
37         temB1 = cc;
38         temC = cc;
39         temC1 = bb;
40         temD = dd;
41         temD1 = aa;
42         return 1;

```

42 }

输出

```
1 Error type 5 at line 34: Type mismatched for assignment.  
2 Error type 5 at line 36: Type mismatched for assignment.  
3 Error type 5 at line 38: Type mismatched for assignment.  
4 Error type 5 at line 40: Type mismatched for assignment.
```

说明：仅 2.2 分组同学需要测试该用例，需要输出上述错误信息。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与王珏助教联系，注意同时抄送给许老师。