编译原理实习　测试用例 1

[Testcase1]

```
#include <stdio.h>

int main()
{
    return 0;
}
```

testcase1 很简单，第 1 行第 1 个字符#并不在我们定义的词法范围之内，因此报错（错误类型 1）。另外，这个错误还有可能连锁地引发一个语法错误，因此如果学生报了一个错误类型 1 和一个错误类型 2 也算对。再多报就不行了。

[Testcase2]

```
int main()
{
    int k;
    if (k % 2 == 0) return 1;
    return 0;
}
```

testcase2 错误在第 4 行的字符%，同样这个错误也能引发另一个语法错，所以报 2 个错误也算对，多报不可。

[Testcase3]

```
struct V_stack test_stack == 1;

int push_addr(struct V_stack stack, struct I_codeList code, int d)
{
    int i = 0;
    struct V_stack st = malloc(sizeof(st));
    struct V_stack original_stack = stack;
    int extra_d = (d * --3) / (4 ++ 2) * (d -= 5);

    if (st == NULL)
    {
        exit(-1;
    }
    st.kind = ADDR;
    st.u.ret_addr = code;

    while (i < extra_d-1)
    {
        stack = stack.next;
        array[4+i-] = array[5 */ 12];
        i = i + 1;
    }
```

```
    st.next = stack.next;
    stack.next = st;

    return original_stack;
}
```

testcase3 是一个比较复杂的例子。这段代码是根据我写的虚拟机小程序的源码修改而来的，其中第 1 行末尾应该是一个赋值号=而不是等号==，此为第一个语法错误；第 8 行中的++和-=都不符合语法规范，因此这一行应该报两个语法错误；第 12 行少了一个括号，此为第四个语法错误；第 20 行的 i-和*/也都不符合语法规范，因此这一行也应该报两个语法错误，其中*/那里可以算作错误类型 2 也可以算作错误类型 3。对于第 8 行和第 20 行来说，它们在同一行内都有 2 个语法错误，这是为了测试学生的 error 产生式写得是否细致。该测试点是所有 14 个测试点中最难的，可以这样算分：这个测试点总共 6 处错误值 5 分，1 个错误 1 分，也就是说能够正确报出其中的 5 处即可得满分，多报则得 0 分。

[Testcase4]

```
int main(int argc, float argv[3])
{

    if (argc != 2)
    {
        exit(0);
    }

    struct FILE f;
    if (!f)
    {
        perror(argv[1]);
        return f++;
    }

    while (i=1;) i = i + 1;
}
```

testcase4 中，由于变量定义必须在语句块开头，因此第 9 行的变量定义属于语法错误。第 13 行的++和 16 行括号内的 i=1;也都是语法错误

[Testcase5]

```
int type_compare
(struct S_type t1, struct S_type        t2)
{
    if (t1 == t2)
return 0;
    if (t1 == NULL ||                   t2 == NULL) return
-1;

    if (t1->kind !=
                                                t2.kind) return -1;
```

```
    if (t1.kind.kind2 == t2.kind.kind1) return -1;

    while (t1.next != NULL) {
        t1 = t1.next;
    }
```

testcase5 中我故意使用空白符将输入文件打散。第 9 行有一个->属于语法错误，而另一个语法错误是该函数最外层语句块的花括号不匹配，有{却没有}与之对应。一般而言我们将这个错误算在最后一行，也就是 18 行上。

[Testcase6]

```
int fibonacci(int n)
{
    int a = 0, b = 1, i = 0;

    while (i < n)
    {
        int c = a + b;
        write(b);
        a = b;
        b = c;
    }
}
```

无错

[Testcase7]

```
int main()
{
    int a[10], i = 0, j = 0;
    while (i < 10)
    {
        a[i] = 50 - i * 2;
        i = i + 1;
    }
    i = 0;
    while (i < 10)
    {
        j = 0;
        while (j < i)
        {
            if (a[i] < a[j])
            {
                int t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
            j = j + 1;
```

```
        }
        i = i + 1;
    }
    i = 0;
    while (i < 10)
    {
        write(a[i]);
        i = i + 1;
    }
}
```

无错

[Testcase8]

```
struct Complex
{
    float real, image;
};

struct Complex multiply(struct Complex x, struct Complex y)
{
    struct Complex z;
    z.real = x.real * y.real - x.image * y.image;
    z.image = x.real * y.image + y.real * x.image;
    return z;
}

struct Complex divide(struct Complex x, struct Complex y)
{
    struct Complex z;
    float abs_sqr = y.real * y.real + y.image * y.image;
    z.real = (x.real * y.real + x.image * y.image) / abs_sqr;
    z.image = (y.real * x.image - x.real * y.image) / abs_sqr;
    return z;
}
```

无错

[Testcase9]

```
int matrix_mul(int a[100][100], int b[100][100])
{
    int c[100][100], i=0;
    while (i < 100)
    {
        int _j = 0;
        while (_j < 100)
        {
            int k = 0;
```

```
            c[i][_j] = 0;
            while (k < 100)
            {
                c[i][_j] = c[i][_j] + a[i][k] * b[k][_j];
                k = k + 1;
            }
            _j = _j + 1;
        }
        i = i + 1;
    }

    i = 0;
    while (i < 100)
    {
        int _j = 0;
        while (_j < 100) write(c[i][_j]);
    }

    return 0;
}
```

无错

[Testcase10]

```
int main()
{
    int a = 0123;
    int b = 123;
    int c = 0765;
    int d = 765;
}
```

testcase10 用来测试学生是否能识别八进制整数。其中 $(123)_8=(83)_{10}$，而 $(765)_8=(501)_{10}$

[Testcase11]

```
int main()
{
    float a = 1.05e-4;
    float b = 012.348E+5;
    float c = 09e007;
}
```

testcase11 用来测试学生是否能识别指数形式的浮点数。

[Testcase12]

```
int main()
{
    int a = 0x3F;
```

```
    int b = 0x05Ab;
    int c = 0X2b;
    int d = 0XcDfE170;
}
```

testcase12 用来测试学生是否能识别十六进制整数。其中 $(3F)_{16}=(63)_{10}$，$(05Ab)_{16}=(1451)_{10}$，$(2b)_{16}=(43)_{10}$，$(cDfE170)_{16}=(215998832)_{10}$

[Testcase13]

```
int fibonacci(int n)
{
    int a = 0, b = 1, i = 0;

    while (i < n)
    {
//      int c = a + b;
        write(b);
//      a = b;
//      b = c;
    }
}
```

testcase13 用来测试学生是否能识别//形式的注释。第 7、第 9 和第 11 行都被注释掉了，因此不能再出现在输出的语法树当中。这个测试点要注意学生输出的行号一定要完全匹配样例输出，否则不得分。

[Testcase14]

```
int main()
{
/*  int a[10], i = 0, j = 0;
    while (i < 10)
    {
        a[i] = 50 - i * 2;
        i = i + 1;
    }*/
/*
    i = 0;
    while (i < 10)
    {
/*      j = 0;
        while (j < i)
        {
            if (a[i] < a[j])
            {
                int t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
            j = j + 1;
```

```
        }
        i = i + 1;
    }
*/
*/
    i = 0;
/*  while (i < 10)
    {
        write(a[i]);
        i = i + 1;
    }
}
```

testcase14 用来测试学生是否能识别/*形式的注释。第 13 行出现了一个嵌套注释，第 27 行出现了一个多余的*/（算作语法错误），另外第 29 行的/*到了文件末尾也没有结束。这个测试点可以报这 3 个错，也可以再多报一个语法错（因为 int main()后面的那个{没有被匹配），但不能少报或者再进一步多报。