2. 黑盒测试：针对课本P293,求一元二次方程根的程序，仅仅设计针对A、B、C的“正数、零、负数、非数值、舍入误差(精度）”的测试用例，充分吗？ 还需要哪些测试用例？参考P302

白盒测试：针对课本P304图8-7的的所有路径测试有多少条？所有路径都测试通过了，能保证程序不会出错？

为了对方程求解程序进行测试，我们需要考虑以下测试用例：

1. 异常情况测试用例：

- 输入 A=0，B=0，C=0，预期输出为无数解。

- 输入 A=0，B=0，C='c'，预期输出为输入有误。

- 输入 A=1，B=2，C=3，预期输出为无实数解。

- 输入 A=1，B=2，C=3，方程不再是一元二次方程，预期输出一个解。

2. 边界测试用例：

- 输入 A=1，B=1，C=0，预期输出为无解。

- 输入 A=1，B=1，C=0，预期输出为无解。

- 输入 A=1，B=1，C=0，预期输出为无解。

- 输入 A=1，B=1，C=0，预期输出为-1和-1。

- 输入 A=1，B=1，C=0，预期输出为-1和-1。

- 输入 A=1，B=1，C=0，预期输出为-1和-1。

对于白盒测试，我们可以考虑以下路径：

1-2-3-4-5-6-7

1-2-4-5-6-7

1-2-3-4-5-6-1

1-2-4-5-6-1

需要注意的是，虽然所有路径都测试通过了，但这并不能保证程序不会出错。白盒测试只能覆盖程序内部的代码路径，但仍然可能存在以下情况：

1. 测试用例未覆盖所有可能的输入，导致某些情况未被测试到。

2. 程序存在逻辑或算法错误，即使所有路径都通过测试，仍然可能存在某些输入导致错误。

3. 程序可能没有处理异常情况，例如除以零或越界，这可能导致程序崩溃或产生错误结果。

因此，除了白盒测试，我们还需要进行更全面的测试，包括边界测试、异常情况测试和性能测试，以确保程序的正确性。

3. 课本P324 习题7, 给出所描述的6种方法对构件进行集成测试的顺序。

自底向上方法：

1)Test E,G,H,J,K,L,M,N

2)Test (F,L),(I,M,N)

3)Test (B,F,L,G),(C,H),(D,I,J,K,M,N)

4)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)

自顶向下方法：

1)Test A

2)Test (A,B,C,D,E)

3)Test (A,B,C,D,E,F,G,H,I,J,K)

4)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)

改进的自顶向下方法：

1)Test A

2)Test B,C,D,E

3)Test (A,B,C,D,E)

4)Test F,G,H,I,J,K

5)Test (A,B,C,D,E,F,G,H,I,J,K)

6)Test L,M,N

7)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)

一次性集成方法：

1)Test A,B,C,D,E,F,G,H,I,J,K,L,M,N

2)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)

三明治方法：

1)Test A,L,M,N

2)Test (A,B),(A,C),(A,D),(A,E),(F,L),(I,M,N)

3)Test (B,F,G),(C,H),(D,I,J,K)

4)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)

改进的三明治方法：

1)Test A,B,C,D,E,F,I,L,M,N

2)Test G,H,J,K,(A,B),(A,C),(A,D),(A,E),(F,L),(I,M,N)

3)Test (B,F,G),(C,H),(D,I,J,K)

4)Test (A,B,C,D,E,F,G,H,I,J,K,L,M,N)