Lab 2 Report

Name 張家齊
Student ID 110598109
Date 2022/04/13

1 Test Plan

1.1 Test requirements

The Lab 2 requires to (1) select 15 methods from 6 classes of the SUT (GeoProject), (2) design Unit test cases by using **input space partitioning (ISP)** technique for the selected methods, (3) develop test scripts to implement the test cases, (4) execute the test scripts on the selected methods, (5) report the test results, and (6) specify your experiences of designing test cases systematically using the ISP technique.

In particular, based on the statement coverage criterion, the **test requirements** for Lab 2 are to design test cases *with **ISP*** for each selected method so that "*each statement of the method will be covered by at least one test case* and *the minimum statement coverage is 70% (greater than Lab 1)*".

1.2 Test Strategy

To satisfy the test requirements listed in Section 1, a proposed strategy is to

(1) select **those 10 methods that were chosen in Lab1** and **5 new methods** that are NOT selected previously. If possible, some of the methods do NOT have primitive types of input or output parameters (if possible).

(2) set the objective of the minimum statement coverage to be greater than that of Lab 1 and adjust the test objective based on the time available (if necessary).

(3) design the test cases for those selected methods by using the **input space partitioning (ISP)** technique.

1.3 Test activities

To implement the proposed strategy, the following activities are planned to perform.

No.	Activity Name	Plan hours	Schedule Date
1	Study GeoProject	2	2022/4/6
2	Learn **ISP** and JUnit	2	2022/4/7
3	Design test cases for the selected methods	10	2022/4/8-2022/4/10
4	Implement test cases	3	2022/4/11

5	Perform tests	1	2022/4/12
6	Complete Lab2 report	1	2022/4/12

1.4 Design Approach

The **ISP** technique will be used to design the test cases. Specifically, the possible partitions and boundary values of input parameters shall be identified first using the **Mine Map** and **domain knowledge** (if applicable). The possible **valid** combinations of the partitions (i.e., **all combination coverage**) as well as the boundary values shall be computed for the input parameters of each selected method. Each of the partition combination can be a possible test case. *Add more test cases by considering the possible values and boundary of the outputs for the methods or by using test experiences.*

1.5 Success criteria

All test cases designed for the selected methods must pass and *the statement coverage should have achieved at least 70%.*

2 Test Design

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.

No.	Class	Method	Test Objective	Inputs	Expected Outputs
1-1	Base32	encodeBase32 (long i, int length)	Functional correctness	T1{i=-1, length=1}, T2{i=0, length=1}, T3{i=1, length=1}, T4{i=1, length=-1}, T5{i=1, length=0}	T1{"-1"}, T2{"0"}, T3{"1"}, T4{"1"}, T5{"1"}
1-2	Base32	encodeBase32 (long i)	Functional correctness	T1{i=-1, }, T2{i=0}, T3{i=1}	T1{"-000000000001"}, T2{"000000000000"}, T3{"000000000001"}
1-3	Base32	getCharIndex (char ch)	Functional correctness	T1:{ch='0' } T2:{ch='z' } T3:{ch='a' }	T1:{0} T2:{30} T3:{IllegalArgumentException}
1-4	Base32	decodeBase32(String hash)	Functional correctness	T1:{hash="rj"} T2:{hash="-rj"}	T1:{753} T2:{-753} T3:{ 753}

				T3:{hash="0000rj"} T4:{hash="a"}	T4:{ IllegalArgument Exception}
1-5	Base32	padLeftWithZeros ToLength (String s, int length)	Functional correctness	T1:{s="a", length=0} T2:{s="a", length=1} T3:{s="a", length=2}	T1:{ "a"} T2:{ "a"} T3:{ "0a"}
2-1	GeoHash	adjacentHash(String hash, Direction direction)	Functional correctness	T1:{hash="gzzzzzzzzzzz", length=Direction.TOP}, T2{hash="rzzzzzzzzzzz", length=Direction.RIGHT}, T3:{hash="", length=Direction}	T1:{"zzzzzzzzzzb"}, T2{"2pbpbpbpbpbp"}, T3:{"adjacent has no meaning for a zero length hash that covers the whole world"}
2-2	GeoHash	adjacentHash(String hash, Direction direction, int steps)	Functional correctness	T1:{hash="gzzzzzzzzzzz",length=Direction.TOP, step=1}, T2{hash="rzzzzzzzzzzz",length=Direction.RIGHT, step=1}, T3{hash="rzzzzzzzzzzz",length=Direction.RIGHT, step=0}, T4{hash="rzzzzzzzzzzz",length=Direction.RIGHT, step=-1}	T1:{"zzzzzzzzzzb"}, T2{ "2pbpbpbpbpbp"}, T3{ "rzzzzzzzzzzz"}, T4{ "rzzzzzzzzzzz"}
2-3	GeoHash	right(String hash)	Functional correctness	T1{hash="rzzzzzzzzzzz"}	T1{"2pbpbpbpbpbp"}
2-4	GeoHash	neighbours(String hash)	Functional correctness	T1{hash="7zzzzzzzzzzz"}	T1{("7zzzzzzzzzx", "kpbpbpbpbpbp", "ebpbpbpbpbpb", "7zzzzzzzzzzy", "ebpbpbpbpbp8", "7zzzzzzzzzzw", "s00000000000", "kpbpbpbpbpbn")}
2-5	GeoHash	encodeHash(double latitude, double longitude)	Functional correctness	T1{latitude=0.000000, longitude=0.000000}, T2{latitude=0.000000, longitude=180.000000 }, T3{latitude=0.000000, longitude=-180.000000}, T4{latitude=90.000000,	T1{"s00000000000"}, T2{"xbpbpbpbpbpb"}, T3{"800000000000"}, T4{"upbpbpbpbpbp"}, T5{"h00000000000"}

				longitude=0.000000}, T5{latitude=0.000000, longitude=-90.000000}	
2-6	GeoHash	encodeHashToLong(double latitude, double longitude, int length)	Functional correctness	T1{latitude=0.000000, longitude=0.000000, length=1}, T2{latitude=0.000000, longitude=180.000000, length=1}, T3{latitude=0.000000, longitude=-180.000000, length=1}, T4{latitude=90.000000, longitude=0.000000, length=1}, T5{latitude=0.000000, longitude=-90.000000, length=1}, T6{latitude=0.000000, longitude=0.000000, length=10}	T1{-461168601842738790 3}, T2{-172938225691027046 3}, T3{461168601842738 7905}, T4{-345876451382054092 7}, T5{-922337203685477580 7}, T6{-461168601842738789 4}
2-7	GeoHash	fromLongToString (long hash)	Functional correctness	T1:{hash=-4611686018427387903L}, T2:{hash=-4611686018427387894L}, T3:{hash=-9223372036854775795L }	T1:{"s"}, T2:{"s000000000"}, T3:{IllegalArgumentException}
2-8	GeoHash	coverBoundingBox(double topLeftLat, final double topLeftLon, final double bottomRightLat, final double bottomRightLon)	Functional correctness	T1:{topLeftLat=0.000000, topLeftLon=0.000000, bottomRightLat=-90.000000, bottomRightLon=180.000000}, T2:{topLeftLat=0.000000, topLeftLon=0.000000, bottomRightLat=0.000000, bottomRightLon=0.000000 }, T3:{topLeftLat=90.000000, topLeftLon=-180.000000, bottomRightLat=0.000000, bottomRightLon=0.000000},	T1:{"Coverage [hashes=[s000000000 00], ratio=Infinity]"}, T2:{"Coverage [hashes=[s000000000 00], ratio=Infinity]"}, T3:{"Coverage [hashes=[8, 9, b, c, d, e, f, g, s, u], ratio=1.25]"}, T4:{"Coverage [hashes=[8, 9, b, c, d,

				T4:{topLeftLat=100.000000,topLeftLon=-190.000000, bottomRightLat=0.000000, bottomRightLon=0.000000}	e, f, g], ratio=0.9529411764705882]"}
2-9	GeoHash	calculateHeightDegrees(int n)	Functional correctness	T1:{n=0}, T2:{n=1}, T3:{n=12}, T4:{n=13}	T1:{18.0}, T2:{45.0}, T3:{1.6763806343078613E-7}, T4:{4.190951585769653E-8}
2-10	GeoHash	widthDegrees(int n)	Functional correctness	T1:{n=0}, T2:{n=1}, T3:{n=12}, T4:{n=13}	T1:{360.0}, T2:{45.0}, T3:{3.3527612686157227E-7}, T4:{4.190951585769653E-8}

The details of the design are given below:
Lab2_ISP_testCaseDesign_110598109.xlsx(因為報表篇幅過寬，插入影響閱讀，我放置報表於相同資料夾(Lab2)下)

3 Test Implementation

The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 3 selected test cases are given below. The rest of the test script implementations can be found in the gitlab branch.

No.	Test method	Source code
1	encodeBase32 (long i, int length)	`public void encodeBase32_withParameterLength() throws Exception { assertEquals("-1", Base32.encodeBase32(-1, 1)); assertEquals("0", Base32.encodeBase32(0, 1)); assertEquals("1", Base32.encodeBase32(1, 1)); assertEquals("1", Base32.encodeBase32(1, -1));`

		assertEquals("1", Base32.encodeBase32(1, 0));
		}
		public void encodeBase32_withoutParameterLength() throws Exception {
		assertEquals("-000000000001", Base32.encodeBase32(-1));
		assertEquals("000000000000", Base32.encodeBase32(0));
		assertEquals("000000000001", Base32.encodeBase32(1));
		}

2	encodeBase32(long i)	

3	adjacentHash(String hash, Direction direction)	public void testAdjacentHash_withoutSteps() throws Exception {
		String hashResult = "";
		// Test border situation (at the poles)
		// The geoHash in (90, 0) is "gzzzzzzzzzzz"
		// The "zzzzzzzzzzzb" will be the result after go Direction.TOP
		hashResult = GeoHash.adjacentHash("gzzzzzzzzzzz", Direction.TOP);
		assertEquals("zzzzzzzzzzzb", hashResult);
		
		// Test border situation (at the -180,180 longitude boundaries)
		// The geoHash in (0, 180) is "rzzzzzzzzzzz"
		// The "2pbpbpbpbpbp" will be the result after go Direction.RIGHT
		hashResult = GeoHash.adjacentHash("rzzzzzzzzzzz", Direction.RIGHT);

		`assertEquals("2pbpbpbpbpbp", hashResult);`
		
		`// ISP`
		`try {`
		`hashResult = GeoHash.adjacentHash("", Direction.RIGHT );`
		`} catch ( IllegalArgumentException throwMessage ) {`
		`assertEquals( "adjacent has no meaning for a zero length hash that covers the whole world" , throwMessage.getMessage());`
		`}`
		`}`

4	decodeBase32(String hash)	`public void decodeBase32() throws Exception {`
		`// the String "rj" is 753 in number`
		`long i = 0;`
		`long decode = 753;`
		`// test basic decode`
		`i = Base32.decodeBase32("rj");`
		`assertEquals(i, decode);`
		`// test decode with prefix "0"`
		`i = Base32.decodeBase32("0000000000rj");`
		`assertEquals(i, decode);`
		`// test negative reverse`
		`i = Base32.decodeBase32("-rj");`
		`decode = -decode;`
		`assertEquals(i, decode);`
		`// ISP`
		`String exceptionMessage = "not a base32 character: a";`
		`try {`
		`Base32.decodeBase32("a");`
		`} catch ( IllegalArgumentException throwMessage ) {`
		`assertEquals( exceptionMessage , throwMessage.getMessage());`

		`}` `}`

5	getCharIndex(char ch)	`public void getCharIndex() throws Exception {`
		`// get the index inside characters[](inside Base32.java)`
		`char ch = '0'; // test first index`
		`assertEquals( 0, Base32.getCharIndex(ch) );`
		`ch = 'z'; // test last index`
		`assertEquals( 31, Base32.getCharIndex(ch) );`
		`ch = 'a'; // test char not in list`
		`String exceptionMessage = "not a base32 character: " + ch;`
		`try {`
		`Base32.getCharIndex(ch);`
		`} catch ( IllegalArgumentException throwMessage ) {`
		`assertEquals( exceptionMessage , throwMessage.getMessage());`
		`}`
		`}`

6	padLeftWithZeros ToLength (String s, int length)	`public void padLeftWithZerosToLength() throws Exception {`
		`// ISP`
		`assertEquals("a", Base32.padLeftWithZerosToLength("a", 0));`
		`assertEquals("a", Base32.padLeftWithZerosToLength("a", 1));`
		`assertEquals("0a", Base32.padLeftWithZerosToLength("a", 2));`
		`}`

7	adjacentHash(Stri ng hash, Direction direction, int steps)	`public void testAdjacentHash_withSteps() throws Exception {`
		`// positive steps value`
		`String hashResult = GeoHash.adjacentHash("wsqqmx41x6fs", Direction.RIGHT, 5 );`

		assertEquals("wsqqmx41x6gu", hashResult);
		// negative steps value
		hashResult = GeoHash.adjacentHash("wsqqmx41x6fs", Direction.RIGHT, -5);
		assertEquals("wsqqmx41x6ck", hashResult);
		// ISP
		assertEquals("zzzzzzzzzzb", GeoHash.adjacentHash("gzzzzzzzzzzz", Direction.TOP, 1));
		assertEquals("2pbpbpbpbpbp", GeoHash.adjacentHash("rzzzzzzzzzzz", Direction.RIGHT, 1));
		assertEquals("rzzzzzzzzzzz", GeoHash.adjacentHash("rzzzzzzzzzzz", Direction.RIGHT, 0));
		assertEquals("rzzzzzzzzzzx", GeoHash.adjacentHash("rzzzzzzzzzzz", Direction.RIGHT, -1));
		}

8	right(String hash)	public void testRight() throws Exception {
		assertEquals("2pbpbpbpbpbp", GeoHash.right("rzzzzzzzzzzz"));
		}

9	neighbours(String hash)	public void testNeighbours() throws Exception {
		// give the GeoHash in (0, 0) which is "7zzzzzzzzzzz"
		List<String> neighbourList = GeoHash.neighbours("7zzzzzzzzzzz");
		List<String> expectNeighborList = Arrays.asList("7zzzzzzzzzzx", "kpbpbpbpbpbp",
		"ebpbpbpbpbpb", "7zzzzzzzzzzy",
		"ebpbpbpbpbp8", "7zzzzzzzzzzw",

		"s00000000000", "kpbpbpbpbpbn");
		for(int index=0; index<8; index++){
		assertEquals(neighbourList.get(index), expectNeighborList.get(index));
		}
		}

10	encodeHash(double latitude, double longitude)	@Test
		public void testEncodeHash_latitudeAndLongitude_withoutLength() throws Exception {
		// the GeoHash in Taipei Tech is "wsqqmx41x6fs"
		// the (double latitude, double longitude) in Taipei Tech is (25.043608, 121.533823)
		String geoHash = GeoHash.encodeHash(25.043608, 121.533823);
		assertEquals("wsqqmx41x6fs", geoHash);
		// ISP
		assertEquals("s00000000000", GeoHash.encodeHash(0.000000, 0.000000));
		assertEquals("xbpbpbpbpbpb", GeoHash.encodeHash(0.000000, 180.000000));
		assertEquals("800000000000", GeoHash.encodeHash(0.000000, -180.000000));
		assertEquals("upbpbpbpbpbp", GeoHash.encodeHash(90.000000, 0.000000));
		assertEquals("h00000000000", GeoHash.encodeHash(-90.000000, 0.000000));
		}

11	encodeHashToLong(double latitude, double longitude, int length)	public void testEncodeHashToLong() throws Exception { assertEquals(-4611686018427387903L, GeoHash.encodeHashToLong(0.000000, 0.000000, 1));

		assertEquals(-1729382256910270463L, GeoHash.encodeHashToLong(0.000000, 180.000000, 1));
		assertEquals(4611686018427387905L, GeoHash.encodeHashToLong(0.000000, -180.000000, 1));
		assertEquals(-3458764513820540927L, GeoHash.encodeHashToLong(90.000000, 0.000000, 1));
		assertEquals(-9223372036854775807L, GeoHash.encodeHashToLong(-90.000000, 0.000000, 1));
		assertEquals(-4611686018427387894L, GeoHash.encodeHashToLong(0.000000, 0.000000, 10));
		}

12	fromLongToString (long hash)	public void testFromLongtoString() throws Exception {
		assertEquals("s", GeoHash.fromLongToString(-4611686018427387903L));
		assertEquals("s000000000", GeoHash.fromLongToString(-4611686018427387894L));
		try {
		GeoHash.fromLongToString(-9223372036854775795L);
		} catch (IllegalArgumentException throwMessage) {
		assertEquals("invalid long geohash -9223372036854775795" , throwMessage.getMessage());
		}
		}

13	coverBoundingBox(double topLeftLat, final double	public void testCoverBoundingBox_withoutMaxHashes() throws Exception {

	topLeftLon, final double bottomRightLat, final double bottomRightLon)	assertEquals("Coverage [hashes=[h, j, k, m, n, p, q, r, s, t, w, x], ratio=1.5]", GeoHash.coverBoundingBox(0.000000, 0.000000, -90.0000000, 180.000000).toString());
		assertEquals("Coverage [hashes=[s00000000000], ratio=Infinity]", GeoHash.coverBoundingBox(0.000000, 0.000000, 0.0000000, 0.000000).toString());
		assertEquals("Coverage [hashes=[8, 9, b, c, d, e, f, g, s, u], ratio=1.25]", GeoHash.coverBoundingBox(90.000000, -180.000000, 0.0000000, 0.000000).toString());
		assertEquals("Coverage [hashes=[8, 9, b, c, d, e, f, g], ratio=0.9529411764705882]", GeoHash.coverBoundingBox(100.000000, -190.000000, 0.0000000, 0.000000).toString());
		}

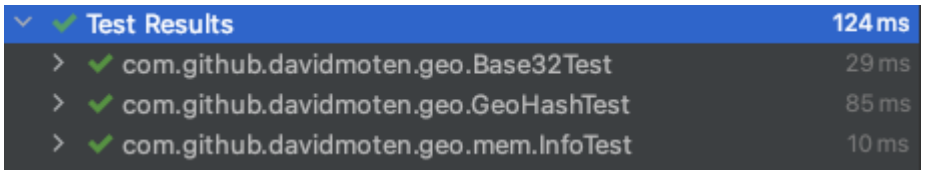
14	calculateHeightDegrees(int n)	public void heightDegrees() throws Exception {
		assertEquals(180.0, GeoHash.heightDegrees(0), 0.001);
		assertEquals(45.0, GeoHash.heightDegrees(1), 0.001);
		assertEquals(1.6763806343078613E-7, GeoHash.heightDegrees(12), 0.001);
		assertEquals(4.190951585769653E-8, GeoHash.heightDegrees(13), 0.001);
		}

15	widthDegrees(int n)	public void widthDegrees() throws Exception {
		assertEquals(360.0, GeoHash.widthDegrees(0), 0.001);
		assertEquals(45.0, GeoHash.widthDegrees(1), 0.001);

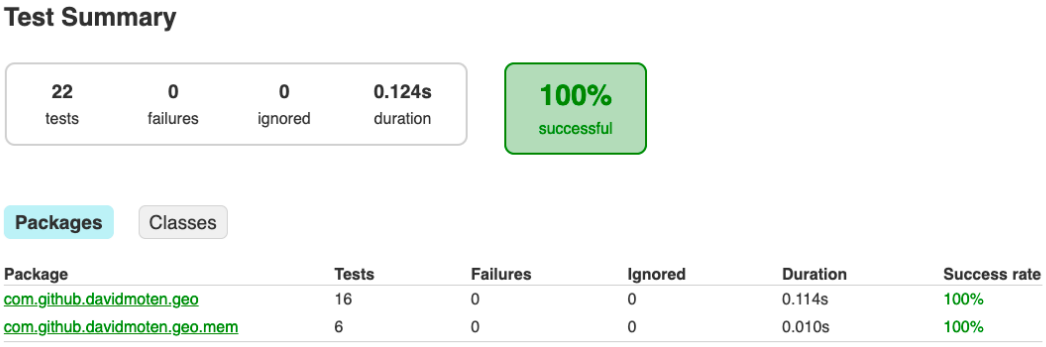
```
        assertEquals(3.3527612686157227E-7,
GeoHash.widthDegrees( 12 ), 0.001);
        assertEquals(4.190951585769653E-8,
GeoHash.widthDegrees( 13 ), 0.001);
    }
```

4 Test Results

4.1 JUnit test result snapshot



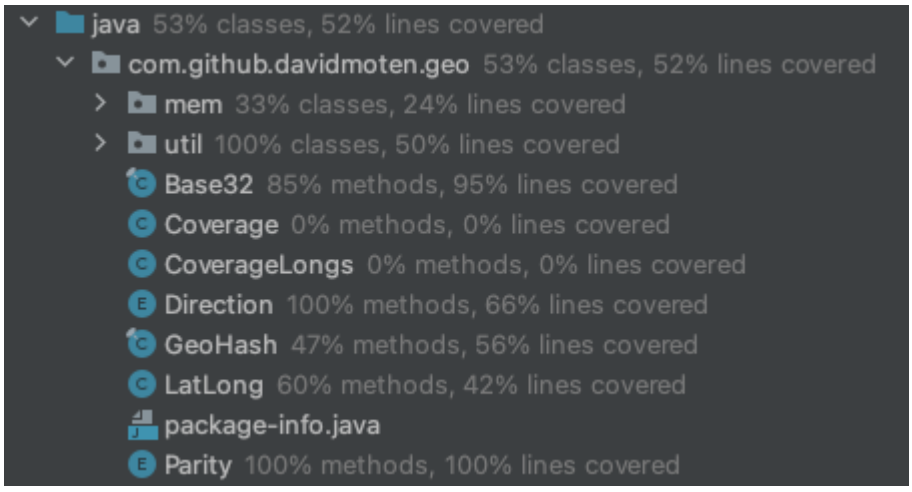
Functional correctness



Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	16	0	0	0.114s	100%
com.github.davidmoten.geo.mem	6	0	0	0.010s	100%

4.2 Code coverage snapshot

- Coverage of each selected method



- Total coverage

geo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo.mem		19%		0%	23	30	48	61	13	20	2	3
com.github.davidmoten.geo		86%		83%	37	149	48	348	17	68	0	10
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	530 of 2,326	77%	48 of 186	74%	61	183	97	415	30	90	2	14

4.3 CI result snapshot (3 iterations for CI)

- CI#1 Lab1 的實作案例(10 個)

README.md

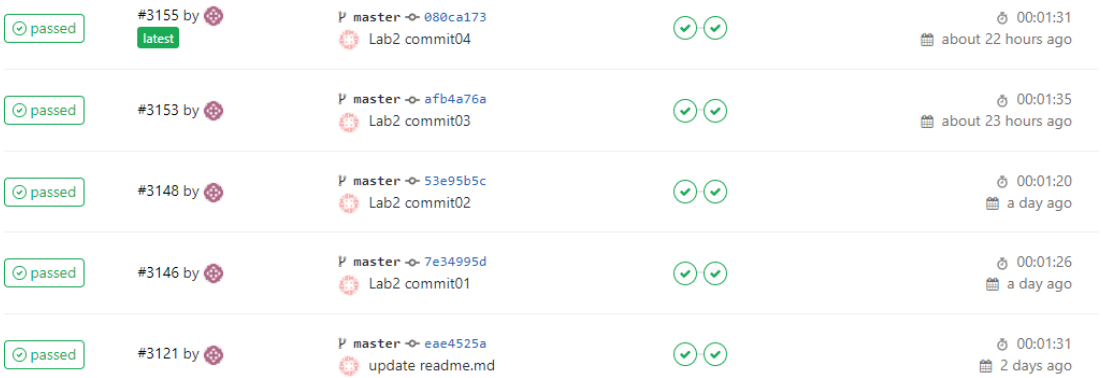
- CI#2 Lab2 加入新增的 5 個 method

README.md

- CI#3 以 ISP 重新設計測試案例(提升 1%)

README.md

- CI Pipeline



5 Summary

In Lab 2, **15 test cases have been designed and implemented using JUnit and the ISP technique**. The test is conducted in 3 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 70%.** Thus, the test requirements described in Section 1 are satisfied.

在此 Lab 中，三次 CI 測試依序是(1)原先 Lab1 的 10 個 Test case (2)新增的 5 個 Test case，且透過 ISP 設計之 (3)以 ISP 重新設計 Lab1 的 10 個 test case。在第二至第三的測試流程中，所產生之 coverage 僅上升 1%，但我認為這是因為在 Lab1 中比起 monkey testing，我已經先行參閱程式碼的各個 branch，並且盡可能的提升 state coverage，這也導致加入 ISP 效果不顯著。

透過這次實作，了解到 ISP 好之處在於只需要具備 Domain knowledge，分配好輸入邊界，就可以在不用實際參閱程式碼的情況下將 Coverage 達到一定的品質。