Structural Testing and Code Coverage

Consider the following piece of code, which plays a game of Blackjack:

```
1  public int play(int left, int right) {
2      int ln = left;
3      int rn = right;
4      if (ln > 21)
5          ln = 0;
6      if (rn > 21)
7          rn = 0;
8      if (ln > rn)
9          return ln;
10     else
11         return rn;
12 }
```

1. You have written only one test where left=22 and right=21.

a) What is the line coverage? 80%

b) What is the branch coverage? 50%

c) What is the branch+condition coverage? 50%

d) What is the path coverage? 12.5%

2. What is the minimum number of tests needed for 100%:

a) line coverage? 2 ... branch coverage? 2

b) branch+condition coverage? 2 ... path coverage? 8

Consider the expression (A & B) | C with the following truth table:

Test case	A	B	C	(A & B) | C
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	T
8	F	F	F	F

3. What test suite(s) achieve 100% MC/DC?

```
A - {2,6}
B - {2,4}
C - {3,4}, {5,6}, {7,8}
Either {2,3,4,6} or {2,4,5,6}
```

4. Draw the truth table for the expression A & (A||B). What test suite(s) achieve 100% MC/DC? What can you say about this piece of code?

```
A - {1,3}, {2,4}
B - none
Either {1,3}, {2,4}
Expression can be simplified to A
```

Consider Java's implementation of the LinkedList's computeIfPresent() method:

```
1  public V computeIfPresent(K key,
2              BiFunction<? super K, ? super V, ? extends V> rf) {
3    if (rf == null) {
4      throw new NullPointerException();
5    }
6    Node<K,V> e;
7    V oldValue;
8    int hash = hash(key);
9    e = getNode(hash, key);
10   oldValue = e.value;
11   if (e != null && oldValue != null) {
12     V v = rf.apply(key, oldValue);
13     if (v != null) {
14       e.value = v;
15       afterNodeAccess(e);
16       return v;
17     } else {
18       removeNode(hash, key, null, false, true);
19     }
20   }
21   return null;
22 }
```

5. What is the minimum number of tests needed for 100% (and why):

a) line coverage? 3 (some conditions are mutually exclusive)

b) branch coverage? 4 (most nested needs 2 + 1 x 2 for each outer)

c) branch+condition coverage? 5 (one branch has 2 conditions)

d) path coverage? 16 (2^4)

e) MC/DC? 5 (4 + 1)

Consider the following method:

```
1  public String sameEnds(String string) {
2    int length = string.length();
3    int half = length / 2;
4    String left = "";
5    String right = "";
6    int size = 0;
7    for (int i = 0; i < half; i++) {
8      left = left + string.charAt(i);
9      right = string.charAt(length - 1 - i) + right;
10     if (left.equals(right)) {
11       size = left.length();
12     }
13   }
14   return string.substring(0, size);
15 }
```

6. How many tests are needed to achieve 100% line coverage? 1

7. How many tests are needed to achieve 100% branch coverage? 3

8. How many tests are needed to achieve 100% branch+condition coverage? 3

9. How many tests are needed to achieve 100% path coverage? infinite

10. How many tests are needed to satisfy the ***loop boundary adequacy criterion***? What are these tests? Give concrete examples.

3
loop zero times
loop once
loop multiple times

Consider the following remove method:

```
1  public boolean remove(Object o) {
2    if (o == null) {
3      for (Node<E> x = first; x != null; x = x.next) {
4        if (x.item == null) {
5          unlink(x);
6          return true;
7        }
8      }
9    } else {
10     for (Node<E> x = first; x != null; x = x.next) {
11       if (o.equals(x.item)) {
12         unlink(x);
13         return true;
14       }
15     }
16   }
17   return false;
18 }
```

11. Provide a test suites that achieves 100% line coverage:

null, remove null
7 remove 7
empty remove 5

12. Does this test suite satisfy the ***loop boundary adequacy criterion***? If not, provide additional tests.

No
7, null remove null
7,5, null remove null
5, 7 remove 7
1,5,7 remove 7
7 remove 5
1,7 remove 5