Structural Testing and Code Coverage

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

```java
public int play(int left, int right) {
    int ln = left;
    int rn = right;
    if (ln > 21)
        ln = 0;
    if (rn > 21)
        rn = 0;
    if (ln > rn)
        return ln;
    else
        return rn;
}
```

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

a) What is the line coverage?

b) What is the branch coverage?

c) What is the branch+condition coverage?

d) What is the path coverage?

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

a) line coverage? ... branch coverage?

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

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?

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?

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

```java
public V computeIfPresent(K key,
            BiFunction<? super K, ? super V, ? extends V> rf) {
  if (rf == null) {
    throw new NullPointerException();
  }
  Node<K,V> e;
  V oldValue;
  int hash = hash(key);
  e = getNode(hash, key);
  oldValue = e.value;
  if (e != null && oldValue != null) {
    V v = rf.apply(key, oldValue);
    if (v != null) {
      e.value = v;
      afterNodeAccess(e);
      return v;
    } else {
      removeNode(hash, key, null, false, true);
    }
  }
  return null;
}
```

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

a) line coverage?

b) branch coverage?

c) branch+condition coverage?

d) path coverage?

e) MC/DC?

Consider the following method:

```java
public String sameEnds(String string) {
  int length = string.length();
  int half = length / 2;
  String left = "";
  String right = "";
  int size = 0;
  for (int i = 0; i < half; i++) {
    left = left + string.charAt(i);
    right = string.charAt(length - 1 - i) + right;
    if (left.equals(right)) {
      size = left.length();
    }
  }
  return string.substring(0, size);
}
```

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

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

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

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

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

Consider the following remove method:

```java
public boolean remove(Object o) {
  if (o == null) {
    for (Node<E> x = first; x != null; x = x.next) {
      if (x.item == null) {
        unlink(x);
        return true;
      }
    }
  } else {
    for (Node<E> x = first; x != null; x = x.next) {
      if (o.equals(x.item)) {
        unlink(x);
        return true;
      }
    }
  }
  return false;
}
```

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

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