Random Testing

Question 1. Consider the following concurrent program with two threads and shared variable x. Variables tmp1 and tmp2 are local to the respective threads. This program has a concurrency bug: it can lose an update to x.

```
    Thread 1              Thread 2
 1: tmp1=x             4: tmp2=x
 2: tmp1=tmp1+1        5: tmp2=tmp2+1
 3: x=tmp1             6: x=tmp2
```

a. Write one possible execution of the six statements that does not cause a concurrency bug.

b. Write one possible execution of the six statements that does trigger a concurrency bug.

c. What is the depth of the concurrency bug?

d. Specify the ordering constraints needed to trigger the bug.

Question 2. Consider the following pseudo-Java function, in which `HashMap<char,int>` is used. A `HashMap<K,V>` is a data structure that associates a value of type `V` to a key of type `K`. The value `v` associated with a key `k` can be set with the API call `put(k, v)`, and the value associated with the key `k` is returned by the API call `get(k)`. For this problem, if no value has been associated with `k`, then assume `get(k)` returns `0`.

```
double charRatio(String s, char a, char b) {
  int N = s.length();
  HashMap<char,int> counts = new HashMap<char,int>();
  for (int i = 0; i < N; i++) {
    char c = s.charAt(i);
    int v = counts.get(c);
    counts.put(c, v+1);
  }
  return counts.get(a) / counts.get(b);
}
```

Describe how you could use a fuzzer to test this function. What bugs would you expect a fuzzer to identify in this function? What bugs would be more challenging for a fuzzer to identify? Explain your reasoning fully, including any assumptions you are making.