1) Considering the program:

```c
int countZeros(int array L, int length) {
    postcondition: returns number of zero elements in L
    int count = 0;
    int index = 0;
    while (index < length) {
        if (L[index] == 0) {
            count = count + 1;
        }
        index = index + 1;
    }
    return count;
}
```

If the line:
```
while (index < length) {
```
were miswritten as:
```
while (index <= length) {
```
What test case would catch this error (assume the program behaves like a C program)?

2) To speed up mutation testing, one idea that was tried was to only test a method with test data that covered that method being mutated rather than using the whole test suite. Why did this produce different results than when the whole test suite was used (give an example of the sort of situation that would raise this issue).
3) Trying to speed up the program in question 1, we write

```c
int countZeros(int array L, int length) {
    postcondition: returns number of zero elements in L
    int count = 0;
    int index = 0;
    while (index < length - 1) {
        LI:
        if (L[index++] == 0) {
            count = count + 1;
        }
        if (L[index++] == 0) {
            count = count + 1;
        }
    }
    AF:
    if (index < length) {
        if (L[index] == 0) {
            count = count + 1;
        }
    }
    PO:
    return count;
}
```

What do we want to use for A) LI, the loop invariant, B) AF the situation just after the loop, and C) PO, the post condition before the return. Express your answer in the SPEC# style logic we have been using so far.

4) How does testing this program differ from testing the original program?