1a) a race condition occurs when more than one thread accesses the same memory location (same data) with no ordering constraints, such that at least one of these accesses is a write. Both threads in question 1 perform reads and writes to both x and y, and these operations can be interleaved such that the order of the operations is not deterministic. Therefore this program DOES have race conditions. Note: the fact that the variables are all declared as volatile doesn't change anything with regards to race conditions.

1b) the following control flow would lead the program to terminate:

```
T2: while (x == y)          // true
T1: while (stop == 0)       // true
T2: y = x;                  // y = 1
T1: y = 2*y;                // y = 2
T2: x = y;                  // x = 2
T1: x = 2*x;                // x = 4
T2: stop = (x - y);         // stop = 2
T1: while (stop == 0)       // false
T2: while (x == y)          // false
--terminated--
```

1c) Please see next page

1c) The Java code can be translated into machine code:

T1		T2	
0) - check condition	2 ops	X) - check condition	3 ops
1) - load y	4 ops	A) - load x	4 ops
2) - store 2y into y		B) - store x into y	
3) - load x		C) - load y	
4) - store 2x into x		D) - store y into x	
		E) - calculate stop	3 ops

- First lets consider the total number of interleavings including the while condition checks and the last instruction of T2 (calculating stop = x-y).
  T1 has 6 operations, and T2 has 10 operations ∴ total # interleavings = $\frac{(6+10)!}{6! \times 10!} = 8008$

- Now lets consider the particular interleavings such that parts of T2 cause T1 to terminate
  Order of execution:
  1, 2, A, B, C, D, 3, 4
  1, 2, A, B, C, 3, D, 4
  1, 2, A, B, 3, C, D, 4
  1, 2, A, 3, B, C, D, 4
  1, 2, 3, A, B, C, D, 4.
  } These 5 interleavings result in T1 termination since it must be 1, 2, *, *, *, *, *, 4.

- Now lets consider the particular interleavings such that parts of T1 cause
  Order of execution:
  A, B, 1, 2, C, D, 3, 4
  A, 1, B, 2, C, D, 3, 4
  A, 1, B, 2, C, 3, D, 4
  A, B, 1, 2, C, 3, D, 4
  1, A, B, 2, 3, C, D, 4
  1, A, B, 2, C, 3, D, 4
  1, A, B, 2, C, D, 3, 4
  (etc) → such that B is ahead of 2 and D is ahead of 4
  } There are 9 interleavings that result in T2 termination since there are 3 ways to write the first half and 3 ways to write the second half such that B is ahead of 2 and D is ahead of 4.

- Now we consider the "stop = x-y" in T2.
  This instruction must not interleave for the termination to occur. It must be at the end of T2.

- Now we consider the number of interleavings of the while-loop condition checks with each other (but not the body).
  In T1 the condition is 2 ops
  In T2 the condition is 3 ops
  } → total # interleavings = $\frac{(2+3)!}{2! \times 3!} = 10$

- Therefore there are 10 × 5 ways for T1 to terminate = 50.
  and 10 × 9 ways for T2 to terminate = 90.

- So this means there are 8008 − (50+90) = 7868 interleavings that result in <u>both</u> threads <u>NOT</u> terminating.

- Now there are 7868 × 140 × 10 interleavings that result in both loops executing twice and then terminating. The ×10 is for the number of ways of executing the while condition interleavings the 3rd time to determine that both fail.
  = 11015200 interleavings.

- Now, how many <u>TOTAL</u> number of interleavings are there for both threads looping twice and then failing?

  T1 will go through 2 × 6 + 2 = 14 ops, and
  T2 will go through 2 × 10 + 3 = 23 ops,
  so total # of interleavings in this case = $\frac{(14+23)!}{14! \times 23!} = 6107086800.$

Therefore the odds of this situation happening are:

$$\frac{11015200}{6107086800} \cong 0.00180 \text{ or about } \boxed{0.18\%}$$