Races and locks

How many data races does the following two-thread program have?

```
int counter = 0;
```

thread t

```
int cnt;

1   cnt = counter;
2   counter = cnt + 1;
```

thread u

```
int cnt;
cnt = counter;              3
cnt = cnt + 1;              4
System.out.println(cnt);    5
```

1. One
2. Two
3. None
4. It depends on the run

How many data races does the following two-thread program have?

```
int counter = 0;
```

thread t		thread u	
`int cnt;`		`int cnt;`	
`cnt = counter;`	1	`cnt = counter;`	3
`counter = cnt + 1;`	2	`cnt = cnt + 1;`	4
		`System.out.println(cnt);`	5

1. One
2. Two
3. None
4. It depends on the run

What does the following two-thread program print?

```
int counter = 0;
```

thread t

```
int cnt;

1   cnt = counter;
2   counter = cnt + 1;
```

thread u

```
int cnt;
cnt = counter;              3
cnt = cnt + 1;              4
System.out.println(cnt);    5
```

1. It always prints 0
2. It always prints 1
3. It sometimes prints 0 and sometimes prints 1
4. It sometimes prints 1 and sometimes prints 2

What does the following two-thread program print?

```
int counter = 0;
```

thread t		thread u	
`int cnt;`		`int cnt;`	
		`cnt = counter;`	3
1 `cnt = counter;`		`cnt = cnt + 1;`	4
2 `counter = cnt + 1;`		`System.out.println(cnt);`	5

1. It always prints 0
2. It always prints 1
3. It sometimes prints 0 and sometimes prints 1
4. It sometimes prints 1 and sometimes prints 2

What does the following two-thread program print?

```
Lock one = new Lock(); Lock two = new Lock();
```

thread t		thread u	
1	`one.lock();`	`one.lock();`	6
2	`two.lock();`	`two.lock();`	7
3	`System.out.prinln("t");`	`System.out.prinln("u");`	8
4	`two.unlock();`	`two.unlock();`	9
5	`one.unlock();`	`one.unlock();`	10

1. It prints `"t"` followed by `"u"`.
2. It prints `"u"` followed by `"t"`.
3. Either of the answers above.
4. Either of the answers above or it does not print anything.

What does the following two-thread program print?

```
Lock one = new Lock(); Lock two = new Lock();
```

thread `t`

```
1  one.lock();
2  two.lock();
3  System.out.prinln("t");
4  two.unlock();
5  one.unlock();
```

thread `u`

```
one.lock();                          6
two.lock();                          7
System.out.prinln("u");              8
two.unlock();                        9
one.unlock();                        10
```

1. It prints `"t"` followed by `"u"`.
2. It prints `"u"` followed by `"t"`.
3. Either of the answers above.
4. Either of the answers above or it does not print anything.

What does the following two-thread program print?

```
Lock one = new Lock(); Lock two = new Lock();
```

thread `t`

```
1  one.lock();
2  two.lock();
3  System.out.prinln("t");
4  two.unlock();
5  one.unlock();
```

thread `u`

```
two.lock();                    6
one.lock();                    7
System.out.prinln("u");        8
two.unlock();                  9
one.unlock();                  10
```

1. It prints `"t"` followed by `"u"`.
2. It prints `"u"` followed by `"t"`.
3. Either of the answers above.
4. Either of the answers above or it does not print anything.

What does the following two-thread program print?

```
Lock one = new Lock(); Lock two = new Lock();
```

thread `t`

```
1  one.lock();
2  two.lock();
3  System.out.prinln("t");
4  two.unlock();
5  one.unlock();
```

thread `u`

```
two.lock();                    6
one.lock();                    7
System.out.prinln("u");        8
two.unlock();                  9
one.unlock();                  10
```

1. It prints `"t"` followed by `"u"`.
2. It prints `"u"` followed by `"t"`.
3. Either of the answers above.
4. Either of the answers above or it does not print anything.

What behavior does class `S` implement?

```
class S {
  private Semaphore s = new Semaphore(1);
  public void x() { s.down(); }
  public void y() { s.up(); }
}
```

1. Class S implements a lock using a semaphore.
2. Class S implements a thread-safe counter.
3. Class S implements a weak semaphore using a binary semaphore.
4. Class S implements a strong semaphore using a weak semaphore.

What behavior does class `S` implement?

```
class S {
  private Semaphore s = new Semaphore(1);
  public void x() { s.down(); }
  public void y() { s.up(); }
}
```

1. Class S implements a lock using a semaphore.
2. Class S implements a thread-safe counter.
3. Class S implements a weak semaphore using a binary semaphore.
4. Class S implements a strong semaphore using a weak semaphore.