1. Readers-Writers Problem에서 Reader 프로세스가 다음과 같은 자료 구조를 가지고 있다.

```
semaphore rw_mutex = 1;
semaphore mutex = 1;
int read_count = 0;
```

다음 사례들 중에서 rw_mutex 세마포어를 사용할 필요가 없는 경우로 가장 적절한 것은?

1) 첫번째 reader 프로세스가 임계구역에 진입할 때.
2) 마지막 reader 프로세스가 임계구역에 진입할 때.
3) writer 프로세스가 쓰기 작업을 완료할 때.
4) 두번째 reader 프로세스가 임계구역에 진입할 때.

2. Bounded Buffer Problem의 consumer process 구조를 아래와 같이 구현했다.

```
while (true) {
    (A) wait(full);
    (B) wait(mutex);
    ...
    /* remove an item from buffer to next_consumd */
    ...
    (C) signal(mutex);
    (D) signal(empty);
    ...
}
```

위 슈도 코드에 대한 설명으로 가장 적절한 것은?

1) (A)에서 wait(full)을 했을 때 signal(full)을 해 주지 않으므로 deadlock이 발생할 것이다.
2) (B)에서 mutex를 wait()해 주는 것은 counting semaphore를 사용했기 때문이다.
3) (C)에서 signal(mutex)를 호출하는 것은 다른 consumer 프로세스가 임계구역에 진입할 수 있게 해 준다.
4) (D)에서 signal(empty)를 호출했으므로 producer 프로세스가 임계구역에 진입할 수 있을 것이다.

3. 〈잠자는 TA 문제〉
   주니온 교수는 운영체제 과목을 어려워하는 학생들을 도와줄 수 있는 TA를 한 명 배정하기로 했다.
   TA가 대기하는 IT융복합관 999호는 좁아서 한 명의 TA와 한 명의 학생만 들어갈 수 있다.
   999호 바깥 복도에는 3개의 의자가 있으므로 TA가 한 명을 돕고 있을 때는 여기서 기다릴 수 있다.
   도움을 요청하는 학생들이 없으면 잠이 많은 TA는 항상 졸고 있다.
   TA의 도움이 필요한 학생들은 999호에 도착해서 TA가 졸고 있으면 깨워서 도움을 받을 수 있다.
   만약 이미 도움을 받는 학생이 있으면 빈 의자에서 대기하고, 빈 의자가 없으면 돌아갔다가 다시 와야 한다.

   위 문제에 대해서 동기화 방법을 조율하는 해결책에 대한 설명으로 옳다고 할 수 없는 것을 모두 고르시오.

   1) TA 1명과 도움이 필요한 n명의 학생들을 프로세스, 혹은 쓰레드로 구현하면 좋겠다.
   2) TA가 졸고 있을 때 도착한 학생은 조교를 깨우기 위해 뮤텍스 락을 이용해 깨우면 좋겠다.
   3) 빈 의자에 앉을 때는 의자의 수로 초기화된 카운팅 세마포어를 이용하면 좋겠다.
   4) 빈 의자가 가득찼을 때, 학번이 더 높은 학생이 도착하면 학번이 낮은 학생을 선점(premption)하도록 하면 deadlock 문제가 발생할 수 있다.
   5) TA의 도움을 받은 학생이 나가면서 바깥 의자에 대기하는 학생들에게 끝났다고 알려주지 않으면 starvation 문제가 발생할 수 있다.

4. 강의자료에 제공된 Dining Philosophers Problem에 대한 Pthread / Java 솔루션에 대한 설명으로 옳다고 할 수 없는 것을 모두 고르시오.

   1) 철학자들의 저녁식사 문제의 Pthread 솔루션은 Pthread에서 제공하는 monitor lock을 사용하고 있다.
   2) pthread_cond_wait() 함수가 호출되기 전에 condition 변수와 연계된 mutex 락이 먼저 잠겨져야 한다.
   3) 공유 데이터를 변경하는 쓰레드는 pthread_cond_signal() 함수를 호출하여 condition 변수가 참이 되기를 기다리는 쓰레드에게 신호를 보낸다.
   4) Java 솔루션에서는 DiningPhilosopherMonitor 클래스가 monitor 기능을 제공하고 있으며, Philosopher 쓰레드의 갯수만큼의 monitor를 생성하여 각 Philosopher 객체가 monitor instance 하나씩을 가지고 있다.
   5) Java 솔루션에서는 n 명의 철학자 쓰레드를 생성하기 위해서 카운팅 세마포어를 정의하고 초기값으로 n을 설정해 주었다.

5. Thread-safe Concurrent Application을 개발하기 위한 여러 가지 대안에 대한 설명으로 가장 틀린 것은?

   1) transactional memory를 통해 메모리 읽기와 쓰기 연산을 원자적 연산(atomic operation)으로 만들 수 있다.

   2) OpenMP에서 #pragma omp critical 컴파일러 디렉티브로 임계 구역을 지정해 줄 수 있다.

   3) 함수형 프로그래밍 언어들은 명령형 프로그래밍 언어와 달리 상태를 유지하지 않으므로 경쟁조건이나 교착상태가 아예 발생하지 않는다.

   4) 여러 개의 CPU가 존재하는 다중 코어 시스템에서는 각 코어에서 별도의 프로세스(혹은 쓰레드)가 독립적으로 실행되기 때문에 동기화 문제는 크게 문제가 되지 않는다.

Answers (indended by the Question Provider):

1) 4
2) 4
3) 4, 5
4) 1, 4, 5
5) 4