CS 430 Concurrency and Error Handling

Consider the following code in a C-like language:

```
00      int next_ID = 0;
01
02      int get_next_ID()
03      {
04          int new_ID = next_ID;
05          next_ID = new_ID + 1;
06          return new_ID;
07      }
```

1. Suppose the above function is called concurrently from two different threads A & B. Assume individual lines of code execute atomically. Using the notation “A.4” to denote thread A running line 4, provide a trace that demonstrates a race condition. In your trace you only need to consider lines 4-6.

2. Using the same notation, provide a trace that will produce correct results.

Consider the following code in a Java-like language:

```
final int MIN_ALLOWED_VALUE = 1;

int minNum(int nums[])
{
    int min = 0;
    try {
        if (nums.length == 0) {
            throw new ZeroLengthException();
        }
        min = nums[0];
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] < min) {
                min = nums[i];
            }
            if (min < MIN_ALLOWED_VALUE) {
                throw new InvalidDataException();
            }
        }
        return min;
    } catch (InvalidDataException ex) {
        min = -1;
    } catch (ZeroLengthException ex) { /* do nothing */ }
    return min;
}
```

3. What is the return of `minNum([3,5,2,8])`?

4. What is the return of `minNum([])`?

5. What is the return of `minNum([84,99,0,12])`?

6. Re-write this code without exception handlers using `goto` statements and labels.