1. Suppose a systems analyst has just prepared a *detailed requirements* (external system design or functional specifications) document for a proposed new application and you’ve delivered review copies to the user-sponsors and to the programming team.

Then the user-sponsors complain that they can’t understand the I.T. jargon in your documents. One team member suggests that you rewrite the material, preparing:

   a. a special version of the specifications for the users, sticking to business terminology and avoiding technical jargon, and
   b. a separate version of the specifications for the programming team, avoiding business jargon they may not understand.

Explain why that is or is not a good idea, and what the consequences are likely to be if you do or don’t follow this advice. 

If the systems analyst prepares two separate versions of the requirements for the two audiences, we’ll have no way of verifying that they’re consistent with each other until the late stages of testing when user representatives see actual output. Until then the project will be entirely dependent upon the analyst’s consistency in every detail.

Also, any future changes would have to be made to BOTH documents with proper review and approval.

It is the analyst’s responsibility to prepare a single document that both audiences can understand in exactly the same way.

2. Suppose task T20 was estimated to take 12 weeks. At the end of 9 weeks the project team member to whom that task is assigned reports that it is 75% done. If you’re the project manager, how much longer would you expect T to take? Explain your reasoning.

A 12-week task duration is much too long. We have no certain way of verifying progress during that long period, except for the team member’s estimate, which may be influenced by wishful thinking or a desire to avoid confronting bad news.

The task should have been divided into multiple tasks with verifiable deliverables.
3. Consider this mini-project plan:

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Prerequisites</th>
<th>Est. Duration (working days)</th>
<th>Successors</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 (or START)</td>
<td></td>
<td>3</td>
<td>T2,T3,T6, (T8)</td>
</tr>
<tr>
<td>T2</td>
<td>T1</td>
<td>2</td>
<td>T5</td>
</tr>
<tr>
<td>T3</td>
<td>T1</td>
<td>2</td>
<td>T4</td>
</tr>
<tr>
<td>T4</td>
<td>T3</td>
<td>5</td>
<td>T5</td>
</tr>
<tr>
<td>T5</td>
<td>T2, T4</td>
<td>1</td>
<td>T7</td>
</tr>
<tr>
<td>T6</td>
<td>T1</td>
<td>3</td>
<td>T7,T8</td>
</tr>
<tr>
<td>T7</td>
<td>T5,T6</td>
<td>7</td>
<td>FINISH</td>
</tr>
<tr>
<td>T8</td>
<td><strong>T1</strong>, T6</td>
<td>2</td>
<td>T9</td>
</tr>
<tr>
<td>T9</td>
<td>T8</td>
<td>2</td>
<td>T10</td>
</tr>
<tr>
<td>T10</td>
<td>T9</td>
<td>9</td>
<td>FINISH</td>
</tr>
<tr>
<td>FINISH</td>
<td>T7,T10</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

We shall do parts (b), (c), and (d) together in class as an exercise, in order to show not only the result but also the process.

(a) If there are any redundant (non immediate) predecessors, cross them out. (2 points)

(b) Sketch the task network on the back of a page of this exam or on a blank sheet. (13 points)

(c) List the IDs of the tasks along the critical path (or circle them in the above table): ___________ (9 points)

(d) If the mini-project begins on Monday, July 9, and the project team doesn’t work weekends or overtime, and the project has adequate qualified staff, and there are no holidays, when should it end? __________________________________________________________________________ (7 points)

(e) Fill in the successors column with the IDs of each task’s immediate successors. (5 points)
4. You’re the project leader of a team developing an accounting system with 9 programmers and various other staff. It’s early October and the project is in the middle of heavy software development (phase 5 in our sample life cycle).

This week’s status report shows that the remaining work will take your staff until mid-February. But you have already promised the sponsoring user organization that it will be ready by mid-January so they can use it to generate required year-end reports.

In a desperate discussion with your team and your boss, three suggestions have been put forth:

(a). Hire two or three more highly-qualified programmers (either permanent employees or contractors) and put them to work immediately on components of the application. Assume there is enough money in the budget for such emergency funding.

(b). Compress the schedule for final acceptance testing from the five weeks shown on the project plan to just two weeks.

(c). Authorize unlimited overtime for the present team and make them understand that the project absolutely must be finished by January 10. Hint that there might be generous bonuses for meeting the deadline.

What would you recommend and why? (18 points)

It’s possible that there is NO strategy that will meet the sponsor’s expected deadline and we may have to face disappointing the sponsoring organization. We have to work on a best efforts basis and hope we don’t encounter any surprising problems.

We reject alternative b as extremely risky. We must have had a reason in planning the project for allocating five weeks to the final testing. Taking short-cuts there may impair the reliability of the application and cause serious damage.

Brooks’s Law warns us that alternative a may be counterproductive. Taking on more skilled programmers will help only if the program specifications are extremely rigorous and our technology (programming language, coding standards, naming conventions, etc.) is clearly defined and easy enough to be absorbed quickly.

It looks as if alternative c offers the best chance, perhaps with a partial reliance on the other two (for example hire one new programmer or reduce the testing to four weeks).

5. What could/should have been done to avoid the situation in problem 4? (7 points)

October is awfully late to discover that a January completion date is in jeopardy. Our project plan should have included more frequent review points or shorter task durations, so we should have known much sooner that we might be in trouble.