1. We’ve cited four common causes of *project failure* arising from management shortcomings, oversights, or ill-advised compromise. Describe any *two* of them and explain their impact on a large project. (8 points each)

**Insufficient detail in the project plan.** The tasks are too long or too vague. As the project proceeds we have no solid basis against which to compare actual progress, and if a task is late we won’t know for a while.

**Inadequate functional specifications or external design.** The sponsoring users don’t know exactly what they’re going to get and may not even be motivated to read the entire specification carefully. The developers may make their own (invalid) interpretations, which may not be noticed until the late stages of system testing.

The above two were the *main* ones we cited. However, you also got full credit for either of the following (or even near-full credit for others that we hadn’t cited, but were still plausible):

**Inadequate time for thorough system testing.** A faulty product may be released, causing havoc and ill-will among users. This is often the result of compressing remaining parts of a schedule after the earlier parts have slipped.

**Undisciplined change management.** The team gets swamped by late requests from sponsoring end users, but fails to assess or communicate their impact.

2. In specifying a task for a project plan, we must state:
   - A general *description* of the work to be done
   - Estimates of *duration*
   - Estimates of *resources* required (people, equipment, etc.)
   - *Prerequisite* (or predecessor) *tasks* that must be finished before this task can begin.
   - *Contingencies* (what might go wrong, and what will we do if it does)
   - One other thing, perhaps the most important of all. What is that thing and why is it essential to specify it? (12 points)

Obviously specification of the *task deliverables*, the tangible results of performing the task, presumably needed by successor tasks. They constitute the evidence that the task is finished.

If a task produces no tangible, usable, or visible result, what’s the point of doing it?

3. Explain what’s meant by the “critical path” in a project plan. (10 points)

The path through the task network for which the sum of the estimated durations is the longest. Therefore, no additional resources (personnel, machines, overtime, etc.) can compress the schedule to finish earlier than the critical path’s estimated completion date.
4. Choose any two of the following:
   - Theory X and Theory Y
     See https://en.wikipedia.org/wiki/Theory_X_and_Theory_Y
   - Data dictionary
     The collection of rigorous definitions of every data item mentioned anywhere in the specifications (external design). For each data item we specify:
     - Its name
     - Its source.
     - Its external representation (range, units, format, etc.)
     - Its precise and unambiguous meaning in the application domain
     Those definitions will help to clarify programming tasks and avoid misunderstandings discovered late in a project.
   - Victorian Novel Specifications
     Extremely detailed (and often tedious) narrative descriptions of exactly what the application will do in response to each kind of input (transaction). That was a common form of specification before Structured Analysis, and was rarely understood by user representatives who were asked to approve it. In a simpler form it has been resurrected recently in some aspects of UML.

5. Here are two quotes about status reports from well-known writers:

   "While short, daily stand-up meetings are a far more effective way of communicating status than written [status] reports, many traditional managers still insist on documentation. These reports take valuable time—time that could be better spent developing working software."

   "...developers hate status reports. It makes them spend hours each week writing down what seems to them to be obvious, redundant information."—97 Things Every Project Manager Should Know, p. 184

State how you interpret their advice as it applies to individual team members reporting the status of their assigned tasks. State whether you agree with the above advice. Explain your reasons, and describe the likely impact on a project and on the project team of following or of not following their advice. (14 points)

Those writers have a strange idea of written status reporting. They’re thinking of the long-winded narrative reports that substitute for simple affirmation of previous estimates or brief explanations of changes.

Usually more time will be spent in walking to and from the meeting room than in filling out a simple binary status report. Furthermore, daily is usually more frequent than necessary for tight project control, except perhaps in the late stages of final installation and start-up.
“Stand up” meetings often encourage vague reports slanted toward positive content. We can always find *something* good to say about our activity, even after serious setbacks. Experience shows that such reporting leads to complacency on the part of sponsoring users and managers until very late in the project.

However, *occasional* group meetings may help to promote morale among the team members and keep everyone informed.

6 A programming task has an estimated duration of twelve weeks. After nine weeks the programmer to whom it is assigned reports that it is 75 percent done. How much longer should we expect it to take? Why? (6 points)

We have no basis for knowing. If the programmer is highly experienced and has extremely sound judgment, the remaining work may be finished in three more weeks for a resulting task duration of twelve weeks. Or “loose ends” may force the final completion beyond the original estimate.

The original estimated task duration was too long; i.e. the task was too big.

7 a. You’ve been assigned as manager for a potentially major project to develop a *Membership Records* system for a professional association. After conferring with key people in the association you prepared a statement of *scope*, which association management has approved. After careful study (phases 1 & 2 in our sample life cycle) you inform the association that the proposed system will cost between $125,000 and $380,000 to develop and can be ready between June, 2017, and November, 2017.

The association responds that your range is much too broad. They need more information before deciding whether to commit to an undertaking that might cost $380,000. They’ve made, with your help, a return on investment (ROI) analysis, which showed that the project is worth doing if it will cost less than $250,000 and can be usable before their worldwide convention in October, 2016. Before authorizing you to undertake the project, they insist on knowing whether the project will be kept within those limitations.

What are the association’s reasonable options at this point? As project manager, what would you propose that they do and why? What arguments would you use to persuade them? How can you assure the association that they won’t lose control of a project that might then greatly exceed their cost limit? (16 points)

The most important thing is to resist the temptation to state “acceptable” (to the sponsors) estimates in the hope of “somehow” being able to deliver.

A better strategy is to try to persuade the sponsoring users to fund the *specification* effort (phase 3 in our sample life cycle). If we can finish that work in, say, 15 weeks for a cost of another $30,000 we should be in a position then to give a much narrower estimate of the remaining cost and time. We may even then discover a packaged software product solution that will further shorten
the schedule and lower the price. Or we may not, but it will have cost only a few thousand dollars to find out.

We may have to persuade the sponsoring organization that there is no magic solution, and that spending a little time and money on the front end is the normal approach to major development projects. We can cite examples of projects where initial estimates were greatly exceeded and the organization lost control.

Note that the “agile” approach of developing fragments of the application in “sprints” would require an unpredictable (and possibly excessive) expenditure of funds and time before we know whether a useful application will be possible within the users’ time and budget constraints.

7 b. What reaction would you hope to get from a reasonable user executive? What would you do during the rest of the project to back-up the assurances you’ve stated to them? (8 points)

We hope that they:

- understand that there is no magic process by which we can know everything before even starting a project.
- trust our competence and integrity.

If they don’t then we may have to let them shop for an alternative outside our organization. (i.e. try to do it themselves or hire a contractor.)

Then we must keep the sponsoring users informed with up-to-the-minute status information.

8 In the sample seven-phase life cycle we used in this course, we cited the boundary between phase 3 and phase 4 as “the critical point”. What is so special about that point? Why is it critical? (8 points)

It’s the last point in a project at which we haven’t actually built anything yet. If users reviewing the specifications want something changed, the cost is just in changing documentation, not program code and no re-testing.

9 In some organizations the very best programmers get promoted to become systems analysts. Is that a good or bad strategy? Why? What results, either good or bad, are likely to occur? (10 points)

There may be some programmers who are also sincerely interested in systems analysis and are potentially good at it. They may be eager to dig into the users’ business.

Most programmers, however, prefer to continue doing creative design and problem solving work. If we force them to become analysts, they may be tempted to distort requirements into their proposed solutions. The organization loses an excellent programmer and gains an incompetent analyst.