1. For each of the following situations, write "**agile**" if an Agile process is (generally) better suited, "**planned**" if plan-driven process is better suited, or "**equal**" if they are both about equally suitable or if the situation is irrelevant to choice of process type.

__Planned__ Most of the team are excellent developers; only about 10% of the team have good software process capabilities. (*According to the book, the Agile home ground is teams with 30% Cockburn level-2 or -3 developers.*)

__Planned__ There are 40 people on the development team.

___Agile____ Additional requirements will be identified as the software is written.

__Planned__ Project must provide assurance that all code is thoroughly reviewed and tested.

___Equal___ Project must use iterative development.

__Planned__ Many non-functional requirements, such as response-time and scalability.

___Agile____ Project will release software to customer for feedback and testing every week.

__Planned__ There are 4 projects developed by different teams that will be integrated into final product.

___Equal___ A mobile application using web services provided by Facebook and Google.

___Equal___ The application must have a complete written user's guide that describes all features.

2. Write a **one-sentence description** of the typical Agile approach and RUP approach (RUP as example of plan-driven process) to each of the following.

2.1 Architecture and overall software design.

RUP: Design and validate (using an architectural prototype) the architecture early. This does not mean the architecture or design are "fixed".

Agile: No up-front architectural design. In each iteration design only for the features that will be implemented in that iteration. In general, use the simplest design that works for requirements implemented so far. Wrong answer: "no design". Agile does some design work during each iteration.

You need to understand this difference to answer some other questions correctly. This point was mentioned many, many times in the chapter from Balancing Agility and Discipline. A consequence of Agile's "simple design" based on the current subset of requirements means that the design often must be changed during the project, resulting in effort spent on refactoring. Hence, when there are many requirements (known in advance), you need to integrate many components, or other

2.2 Roles and Assignment of Work

RUP: Has roles with clearly defined responsbilities. Work is generally assigned by team leader, development leader, or test leader.

Agile: No predefined roles, other than Scrum Master or the like. Teams are self-organizing and team members *take on tasks themselves*. Work is not assigned. This does not mean anyone can do any job. Agile teams will have people with particular strong skills and interests, just like any team. So you may find one person doing all the database related tasks, a few people doing all the UI work, etc.

2.3 Communication and shared knowledge about the project

RUP: Relies more on explicit, written documentation about the project, including architectural notebook, software design document, memos, and analysis of requirements. Face-to-face communication is important, but followed up by written record of significant information and decisions.

Agile: Relies more on tacit knowledge and verbal communications.

3. Why is high turn-over of developers (people leaving a project) more a problem for Agile processes than for plan-driven processes? [Anaswer in 1-2 sentences.]

Since Agile relies more on tacit knowledge, when a person leaves his knowledge about the project is lost, too. In plan-driven teams, more of the knowledge gained is preserved in writing so a new team member can get up to speed using written docs. This reasoning is from *Balancing Agility and Discipline*.

Wrong: (1) because Agile teams are smaller (plan-driven teams can be small, too). (2) Because Agile developers are more skilled and therefore harder to find a replacement. This is wrong because there's no hard evidence that Agile devs are more skilled than RUP devs, and difficulty of replacing a skilled developer is the same regardless of process type.

4. According to Martin Fowler, what are the objectives of *refactoring?* (Circle **two** correct answers.)

(a) Make the code more efficient.

(b) Make the code easier to understand.

(c) Make the code easier to maintain or modify.

(d) Apply design patterns to existing code.

(e) Make the methods shorter and/or simpler.

Correct: (b) and (c)

5. Name and describe any one refactoring pattern from Martin Fowler's book, other than "Extract Method".

Refactoring Name: *you must use the correct name to get credit*

Motivation [1-2 sentences]

Answer depends on your choice of refactoring. Correctness of answer based on the Refactoring book by Martin Fowler.

Mechanics (what is done to code): [1-2 sentences]

As above, answer depends on your choice of refactoring.

6. Why do Agile methods typically rely more on refactoring than plan-driven methods?

Becuase they do less up-front design, and only enough design to handle the requirements implemented so far. Hence, when a new requirement (from the backlog) is implemented there is more chance that a change in design will be needed to accommodate it, or a better design emerges.

As a result, existing code often needs to be reorganized, i.e. refactored.

7. What does RUP recommend regarding design of software architecture? (This is one of the *Ten Essentials of RUP*.)

Use a component-based architecture.

Note: "component-based architecture" is not the same meaning as "component architecture". But I did not deduct for this.

8. What is the RUP recommended practice for handling changes in requirements? (Another *Ten Essentials*.)

Manage and control changes.

9. In which RUP development phase do the following occur?

Construction	Any components not developed in previous phases are implemented and tested. Components are integrated and the product is tested.
Inception	Business model and business case (potential market, revenues, risks) fare created.
Elaboration	All the use cases are identified, and the majority are written in detail.
Contruction or Transition	User documentation is written and reviewed. *Correct answer is Construction phase, but Transition is acceptable.*
Inception	A project plan is created, showing phases and iterations with the work to be done.
Construction	Software is *beta tested* to validate it against user expectations.
Elaboration	The software architecture is decided and verified by an architectural prototype.
Inception	Stakeholders agree on the scope of project.
Inception	At the end of this phase, a decision is made whether the project is worth committing to additional effort to investigate in detail. (If not, the project is cancelled or substantially revised.)
Elaboration	A the end of this phase, a decision is made whether it is worth committing effort to develop the software. (If not, the project is cancelled or substantially revised.)

10. What are **three criteria** used in RUP to decide which use cases to implement first? [Circle **3** answers]

(a) Ease of implementation

(b) High risk (higher risk UC is done first)

(c) Low risk (lower risk UC is done first)

(d) Earned value of use case

(e) Business value (to customer)

(f) Significance to establishing software architecture

(g) Significance to establishing database schema

Correct: b, e, f

11. Answer these questions about Scrum's required group activities, called "ceremonies".

11.1 What is the purpose of the *Sprint Planning Meeting* and what **2 items** are the result of the meeting?

Purpose: to prioritize and agree on stories to be implemented in the sprint, and agree on a sprint goal.

Results (output): Sprint backlog and sprint goal

11.2 What **3 questions** are answered (by everyone) in the daily stand-up meeting?

What did you do yesterday? (or "today" if the meeting is held at end of the workday)

What will you do today? (or "tomorrow" if the meeting is held at end of the workday)

What problems did you encounter? *or* Are you blocked?

11.3 At the end of a sprint, what group activity occurs that involves the customer and the work products?

Product demo!

11.4 At the end of a sprint, what is the purpose of a retrospective?

The real purpose is to find ways to improve the process based on experience.

Specifically, to review what worked, what needs improvement, and what new things to try.

Wrong answer: *to discuss problems. That is only one part of the retrospective, and doesn't capture the real purpose.*

12. What are 3 questions that a Vision statement should answer about the proposed software?

There are many correct answers to this. Here are a few:

- What is the problem that this software will address, and how does the problem affect stakeholders?
- Who are the stakeholders (those affected by the software), and what are their interests/goals?
- What is the vision of a successful outcome of this project, that is, a successful product?
- What will the software do, and why is it useful?
- Why is this project worth doing? What is the value to stakeholders?
- What is the market opportunity for this product?
- What are the main features of this product?
- What are the scope and limitations of this project?
- Who are the stakeholders, how are they affected by the current way of doing things, and how would they benefit from the proposed product?

13. Here are the results for one iteration of an Agile project with 3 developers. The iteration was 5 days long, but only 4 days are available for development. Everyone worked the full 4 days, and completed this work. The remaining time was lost to unplanned tasks, interruptions, and non-productive work.

Iteration 1:

Story #	Importance	Story Points	Developer	Actual Effort (person-days)
S7	25	3	Bird	2.5
S4	22	12	Win	2.0
S5	21	4	Win	1.0
S8	21	2	Bird	1.0
S2	20	6	Pun	0.5
S11	10	9	Pun	2.0

Here is the remaining backlog:

Story #	Importance	Story Points	Developer	Actual Effort (person-days)
S3	14	9		
S6	20	10		
S10	15	12		
S12	12	3		
S13	14	12		
S14	12	4		

13.1 What was the team's ***velocity*** for iteration 1?

Answer: the available effort was 3 people x 4 days = 12 person-days. Hence,

Velocity = 36 story-points / 12 person-days = 3 s.p./person-day

Wrong: using the actual effort spent on stories, i.e. 36 s.p/9 person-days = 4 s.p./person-day

The relation between actual effort and the maximum available effort is called the *focus factor*.

In this problem: *focus factor* = 9 / 12 = 0.75 which is typical for a project.

Acceptable Answer: the book also says that an easy way to calculate velocity is just the number of Story Points per sprint. So I accepted: velocity = 36 story-points/sprint.

But, you'll still need to do the more detailed calculation to answer 13.3

Velocity is the rate at which work is done. It is described on page 26 of *Scrum and XP from the Trenches*, and mentioned many times in the book.

13.2 Based on these tables, did the team violate any Agile guidelines in the first iteration? Justify your answer.

Yes. They implemented the low priority story S11 instead of higher priority S6 (and many other stories) from the backlog.

13.3 For the second iteration Win and Pun will work 4 days and Bird will work only 2 days. Which stories should be included in this iteration's backlog? Show how you calculate this.

Many wrong answers to this question. People tried to estimate the work each developer could do and pick stories for each dev to work on (not Agile). In Agile, developers take on work themselves, its not assigned. Stories may (usually are) broken down into smaller tasks, so you don't need to "fit" the entire story into one developer's available time.

Besides, it is less accurate to estimate each person's productivity. Its more accurate to base estimates on productivity of the entire team.

Many students made statements like "Bird is less skilled because he spent a lot of time doing S7 and S8". You cannot infer this! Maybe the stories were more harder than estimated; maybe Bird spent a lot of time going over them with the customer; maybe Bird's initial solution for S7 and S8 didn't satisfy some requirement, so he redesigned them. Maybe Bird included some indirect time in his effort, such as initializing a git repository or configurating the C.I. system for everyone to use.

Answer: in the second iteration there are 10 person-days available. In the first iteration the velocity was 3, so the estimated amount of work the team can do is:

velocity x manpower = 3 s.p./person-day x 10 person-day = 30 Story Points

So, choosing the **highest priority** stories from the product backlog, the sprint backlog should be:

Story	Importance	Story Points
S6	20	10
S10	15	12
S3	14	9

You should *not* choose S12 or S14 (lower priority) and not S13 (too much effort).

This backlog is 31 story points, which is only *slightly* above the estimate and the *best fit* to the available person-days. The team can *at least* start work on S3 (probably by breaking it down into tasks) during the sprint. If you don't include S3, the team would likely have time remaining before the sprint ends.