1. **How are software changes classified by their purpose? What is the most common purpose of the change?**

   Software Changes are characterized by their purpose and can be classified into:
   1. Perfective changes: These are the most common changes. they introduce new functionality into the existing software and increases its value.

   2. Adaptive changes: These are the changes that are to be done to the existing software to adapt to the existing software. The purpose of adaptive changes is to protect the existing value of the program. If nothing was done in the changed circumstances, the value would greatly decrease or completely disappear.

   3. Corrective changes: these changes correct software bugs and malfunctions. These bugs and malfunctions are deviations from the intended functionality and impact the users in often unexpected ways.

   4. Protective Changes: these changes are invisible to the user, and they shield the software and its value in a proactive way.

   Perfective changes are the most common but do not mean other changes are not common. The purposes of a change are not mutually exclusive, and there are no sharp boundaries.

2. **When is it permissible to do quick-fix changes?**

   The only acceptable circumstance for a quick fix during the evolution stage is in the situation of an emergency, where human life or a substantial value is at stake; thus, the fix has to be done quickly, and the speed outweighs every other consideration. The quick fix is a common strategy of change during the servicing stage; the software value at that point is low; there are no plans for future evolution, and the fixes just keep it afloat. In this situation, there is no reason for any gold plating, and the change is done as cheaply as possible. The patchwork left in the wake of quick fixes does not matter because the value of the software is low anyway, and any additional lowering of the software value is inconsequential.

3. **What is a product backlog?**

   The product backlog is also called a requirements database, and in other contexts, it is called a project wish list because it lists desired future product properties and functions. The product backlog describes a shared vision of the project stakeholders for the future of the product.
   The product backlog is created and incremented in a process of requirements elicitation. The product backlog is updated as the requirements increase or change and from time to time changes are pushed into the existing code.

4. **Describe a situation when a grep search fails. What would you do if this happened to you?**

   grep originally stands for global expression regular print. it is now used as a tool used for concept location. Using grep, programmers formulate queries using the naming convention. this gives a list of the code lines and the files where the specified pattern appears. Thus, making concept location in a large software system easier.

   Let's say there is a software program for a manipulator. This has many components like a camera module, trajectory planning, and tracking, among many others. Now, I want to implement a change in the camera module, say the change is in the detection of aruco transform. As there is a large number of code files, I use grep with the existing naming convention. Eg: grep "Aruco-Transform" aruco.cpp. I expect it to show all the code lines where it detects the pattern. Now, if the naming convention is not followed, I get 0 results. Now, I try to check where the code lines are using other naming conventions and get some results and I try to find where it is located. This has to be done manually, and there is a chance of human error and also missing some components. This results in code bugs after the development with minimum information. This makes the software lifecycle difficult and can lead software into servicing stage if programmers doesn't have sufficient knowledge of the code base to rectify the errors.