1. Explain Helm's role in Kubernetes.
    - ○ Why is Helm preferred over managing plain Kubernetes YAML files?
        - Reusability - Charts can be shared and reused across projects.
        - Versioning - Each chart version is tracked for easy upgrades and rollbacks
        - Customization - Values files allow for environment-specific configurations (development, staging, production)

    - ○ List and describe the key components of a Helm chart.
        - Helm CLI Tool - User interface to all Helm functionalities.
        - Chart Format - Charts are collections of YAML configuration files and Go templates.
        - Repositories - HTTP-based storage for packaged charts (index.yaml and .tar.gz files)

2. Environment-specific Configurations:

    How does Helm handle environment-specific configurations? Provide an example.

    - Helm handles environment-specific configurations by using **values files** (`values.yaml`) and allowing you to override them at install/upgrade time for example having a general values.ymal with all the default values then have prod-values.ymal file with values specific to prod that you can override during install / upgrade.

3. Helm Chart Repositories:

    What is a Helm chart repository, and how can it be hosted? List at least three hosting options.

    - Helm repository host a predefined helm cheats made by 3rd party it can be community official vendor such as Bitnami or prometheus-compunity
    - They can be hosted in Github, S3 bucket, ArtifactHub

4. CI/CD Integration:

    How can Helm be integrated into a CI/CD pipeline? Explain the typical steps involved.

    1. Build app & image.
    2. Lint the chart.
    3. Chart tests.
    4. Package publish the chart.
    5. Dry-run / diff against the cluster.
    6. Deploy to environment.
    7. Post-deploy checks.
    8. Rollback on failure.

9. Secrets & config.