Q1. Quelle est la complexité moyenne d'accès à un élément dans une liste chaînée ?

(a) $O(1)$
(b) $O(\log n)$
(c) $O(n)$
(d) $O(n^2)$

Q2. Dans une pile (stack), l'élément inséré en dernier est :

(a) Le premier à sortir
(b) Le dernier à sortir
(c) Jamais supprimé
(d) Trié automatiquement

Q3. La différence principale entre une pile et une file est :

(a) L'ordre d'insertion
(b) L'ordre de retrait
(c) La taille maximale
(d) L'implémentation en mémoire

Q4. Une file (queue) suit le principe :

(a) LIFO
(b) FIFO
(c) FILO
(d) Random Access

Q5. Une liste doublement chaînée permet :

(a) De parcourir les éléments dans un seul sens
(b) D'accéder directement à la fin sans parcourir
(c) De parcourir dans les deux sens
(d) De stocker uniquement des entiers

Q6. Quelle structure de données est la plus adaptée pour implémenter un système de backtracking ?

(a) Pile
(b) File
(c) Tableau
(d) Graphe

Programmation Orientée Objet (OOP)

Q7. Une classe définit :

(a) Une instance d'objet
(b) Un modèle d'objet

(c) Un espace mémoire réservé

(d) Une fonction globale

Q8. Le polymorphisme permet :

(a) D'utiliser plusieurs classes différentes avec une même interface

(b) D'éviter l'héritage

(c) De supprimer les méthodes virtuelles

(d) De créer des objets sans constructeur

Q9. L'héritage multiple est :

(a) Toujours autorisé

(b) Interdit dans tous les langages

(c) Autorisé selon le langage

(d) Équivalent au polymorphisme

Q10. Une méthode abstraite :

(a) Possède une implémentation par défaut

(b) N'a pas d'implémentation

(c) Est toujours statique

(d) Est privée

Q11. Une interface sert à :

(a) Définir un contrat sans implémentation

(b) Créer des objets

(c) Étendre une classe existante

(d) Optimiser les performances

Q12. L'encapsulation permet de :

(a) Cacher les détails internes d'une classe

(b) Supprimer les attributs publics

(c) Créer plusieurs constructeurs

(d) Empêcher l'héritage

Q13. Une méthode statique :

(a) Peut être appelée sans instance

(b) Nécessite une instance

(c) Est abstraite

(d) Ne peut pas retourner de valeur

Q14. En Python, le mot-clé `super()` sert à :

(a) Appeler la méthode parente

(b) Créer une nouvelle classe

(c) Supprimer une méthode

(d) Créer un constructeur

Clean Architecture

Q15. Le principe fondamental de la Clean Architecture est :

(a) Séparer les couches métier et infrastructure
(b) Minimiser les tests unitaires
(c) Fusionner les contrôleurs et services
(d) Éviter les dépendances inversées

Q16. La dépendance inversée signifie :

(a) Le domaine ne dépend pas des détails techniques
(b) Les modules externes contrôlent la logique métier
(c) Les classes filles dépendent des parents
(d) Les entités dépendent des frameworks

Q17. Dans une architecture propre, les entités :

(a) Ne doivent pas dépendre des frameworks
(b) Sont implémentées dans les contrôleurs
(c) Dépendent des adaptateurs
(d) Font partie de la couche interface

Q18. Le but des "use cases" est de :

(a) Exprimer les règles métier
(b) Gérer la base de données
(c) Contrôler les requêtes HTTP
(d) Stocker la configuration

FastAPI

Q19. FastAPI est principalement basé sur :

(a) Flask
(b) Django
(c) Starlette et Pydantic
(d) Tornado

Q20. Le décorateur `@app.get("/")` définit :

(a) Une route GET
(b) Une route POST
(c) Une route PUT
(d) Une fonction interne

Q21. Pour définir un modèle de données dans FastAPI, on utilise :

(a) `dataclass`

(b) `pandas.DataFrame`

(c) `BaseModel` de Pydantic

(d) `ORMModel`

Q22. FastAPI est connu pour :

(a) Sa rapidité et la validation automatique

(b) Sa lenteur mais stabilité

(c) Son intégration unique avec Flask

(d) Son absence de typage

Q23. Pour lancer une application FastAPI, on utilise :

(a) `python app.py`

(b) `uvicorn main:app -reload`

(c) `flask run`

(d) `fastapi run app`

LLM (Large Language Models)

Q24. Un LLM est principalement entraîné sur :

(a) Des images

(b) Du texte

(c) Du code binaire

(d) Des données sonores uniquement

Q25. Le but d'un LLM est de :

(a) Prédire la prochaine séquence de mots

(b) Compiler du code

(c) Convertir du texte en image

(d) Traduire automatiquement les pages web

Q26. Les modèles comme GPT utilisent :

(a) Des réseaux de neurones de type Transformer

(b) Des arbres binaires

(c) Des réseaux CNN

(d) Des machines à vecteurs de support

Q27. Le prompt engineering consiste à :

(a) Optimiser la formulation d'entrée pour obtenir de meilleures réponses

(b) Modifier le modèle interne

(c) Retrainer un modèle LLM

(d) Changer le format des données d'entraînement

Q28. Les API comme OpenAI ou Ollama permettent :

(a) D'intégrer un LLM dans une application
(b) De créer des bases de données
(c) D'héberger un site web
(d) De gérer des transactions financières

