Examen blanc

1) Git : Système de gestion de version de code
   GitLab : Système de gestion de pleins de fonctions (pipelines, tickets…)
2) Git = Système de contrôle de version distribué : Répertoire est le même pour tout le monde.
   Avantages par rapport à SVN : Pas de dépendance.
3) Git blame : Voir la dernière personne qui a modifié une ligne.
4) Git revert et Git reset : Revert> reset. Ne change pas l'historique.
5) Ajout sans créer de nouveau commit : git add, puis git amend
6) Branche : Pointeur à un commit.
   Pourquoi ? : Faire des essais, organiser le code ;
7) Créer une branch : git branch [nom]
   Basculer vers cette branche : git checkout [branche créer]
   Fusionner les modifications vers la branche principale : git merge [branche principal]
8) Modification de la même ligne de code et qu'on tente de les fusionner : conflits ;
   Comment résoudre : 3 méthodes
   - Fusionne mon truc ;
   - Fusionne le truc de l'autre personne ;
   - Faire la fusion manuellement à l'aides des chevrons.
9) Merge : Fusionne 2 branches
   Rebase : Change le lien de ce commit vers celui souhaiter.
10) Clone : Copie un dépôt git dans notre système.
    Fork : Nouveau système GitLab avec le même dépôt.
11) Up-stream : Envoyer le dépôt système dans le dépôt distant.
12) Workflow « Trunk-based » : Fusion dans la branche main.
13) Workflow « GitFlow » : Faire des branches par fonctionnalités.
14) Différences de travail entre avec ou sans fork :
    - Sans fork : Petit projet.
    - Avec fork : Gros projet.
15) Annuler une fusion : git revert.
16) « Merge commit » : Commit de fusion ;
17) Fetch : Récupère les données du dépôt distant.
    Pull : Récupère et fusionne.
18) Git cherry-pick : Récupérer qu'une partie du code.
19) Commit « squash » : Compression du nombre de commit.
20) Importance des messages de commit claires : Meilleur relecture.
21) Git stash : Stocker les modifications pour éviter de le commit.
22) Git bisect : Dichotomie, trouver des bugs.
23) Checkout avec une modification : checkout non possible
24) Release : Tag + texte (release log et changelog)
    Tag : Version
25) Intégration continue : Vérifier que le programme marche.
    Avantages : Moins d'échec de projet.
26) Etape d'un pipeline : build, lint (style), test, deploy.
27) CI/CD : Continuous Integration : Test en permanence ;

Continuous Deployment : Test sur le déploiement.

28) Nécessaire : Bon test ;
29) Cycle en V : Spécification → Conception → Code → Test
30) Méthode agile : Méthode itérative, le client peut demander des changements.
    Cycle en V : Le client doit savoir ce qu'il veut.
31) Exemple où Cycle en V > Méthode Agile : Banque.
32) Avantage de la méthode Agile : Moins de risque que le client ne soit pas satisfait.
33) Rôle du client en méthode Agile : Le client est toujours là pendant la phase de développement.
34) Oracle : Résultat attendu d'un test.
    Problème de l'Oracle : Difficile à obtenir le résultat attendu.
35) Cas de test : Scénario qu'on veut donner.
    Donnée de test : On test le scénarios.
36) Recette : Test réalisée par le client.
37) Dans les tests logiciels :
    - Validation : Plantage du programme ?
    - Vérification : Résultat attendu ?
38) Test système : Test fait par l'équipe de développement.
    Recette : Test fait par le client.
39) Test unitaire : Mini test
    Test d'intégration : Test de plusieurs fonctionnalités entre eux.
    Test système : Test sur la globalité.
40) Approche pair-wise : Hypothèse que le problème vient d'une paire de fonctionnalités.
41) Intérêt du framework de tests unitaires : Afficher un message d'erreur.
42) Mocking : Simuler une classe pas encore implémenter, ou qu'on ne veut pas utiliser lors des tests.
43) TDD = Test Drien Development : Ecrire le test avant de coder.
44) Mutating testing : Méthode pour vérifier qu'on a fait assez de test.
45) Couverture du code : Pourcentage de code testé
46) Contexte d'utilisation de test aléatoire : Oracle automatisé.
47) Avantages : Génération faciles de gros nombre de tests.
    Inconvénient : Difficulté à repérer les cas rares.
48) Exemples : Propriétés algébriques.
49) Property based testing : Test avec l'Oracle.
50) Test de non-régression : Test lié à des bugs pour éviter de refaire la même erreur
51) Linter : Analyse statique du style du code, des normes imposées, ...
52) Calcul de couverture : Question 45
53) Types de couverture : Instruction, ..., condition, conditions multiples, MCDC
54) Test à 100% de couverture suffisant ? : Non
55) Boite noire : On test sans savoir comment le code marche
    - Avantages : Test possible avant que le code marche
    - Inconvénient : Aveugle aux cas particulier

    Boite blanche : On test par rapport au code
    - Avantages : Couverture plus rapides
    - Inconvénient : On se base sur cette couverture.