問1

プル リクエストをイシューにリンクするために使用されるキーワードは次のどれですか? （3つお選びください。）

【選択肢】

A. fix

B. resolves

C. merge

D. join

E. closed

F. connects

問1（解答）

A. fix

B. resolves

E. closed

【解説】

◆正解の理由：

A. 修正 (fix)：プルリクエストが特定のGitHub Issueを解決し、マージ時にそのIssueを自動的にクローズすることを示すキーワードです。

B. 解決する (resolve)：プルリクエストが特定のGitHub Issueを解決し、マージ時にそのIssueを自動的にクローズすることを示すキーワードです。

E. 閉じる (close)：プルリクエストが特定のGitHub Issueを解決し、マージ時にそのIssueを自動的にクローズすることを示すキーワードです。これらのキーワードは、プルリクエストの説明文やコミットメッセージに「キーワード #Issue番号」の形式で記述します。

◆不正解の選択肢の補足説明：

C. マージ (merge)：プルリクエストを対象ブランチに統合する操作そのものであり、Issueをリンクしたりクローズしたりするためのキーワードではありません。

D. 参加する (join)：GitHub上でIssueとプルリクエストをリンクするために使用される公式なキーワードではありません。

F. 接続します (connect)：GitHub上でIssueとプルリクエストをリンクするために使用される公式なキーワードではありません。

問2

保存された返信の作成にはどの構文が使用されますか?

【選択肢】

A. HTML

B. YAML

C. マークアップ

D. マークダウン

問2（解答）

D. マークダウン

【解説】

◆正解の理由：

GitHubの保存された返信（Saved replies）は、GitHubのコメントやプルリクエストの記述などと同様に、Markdown構文を使用して内容を記述します。Markdownは、シンプルで読みやすいプレーンテキスト形式で記述でき、GitHub上で自動的にリッチテキストとして表示されるため、広く利用されています。

◆不正解の選択肢の補足説明：

A. HTML：ウェブページの構造を記述するためのマークアップ言語ですが、GitHubの保存された返信やコメントを直接記述する際の主要な構文ではありません。

B. YAML：データシリアライゼーション言語であり、設定ファイルやデータ交換、例えばGitHub Actionsのワークフロー定義などに使用されますが、返信のテキスト構文ではありません。

C. マークアップ：テキストに構造や表現を追加するための記記号やタグのシステム全般を指す広範な用語です。Markdownはその一種ですが、より具体的な正解は「Markdown」です。

問3

ユーザーとして、提案された変更を GitHub 上のリポジトリにマージするためにどの機能を使用できますか?

【選択肢】

A. ディスカッション

B. プロジェクト

C. プルリクエスト

D. イシュー

問3（解答）

C. プルリクエスト

【解説】

◆正解の理由：

プルリクエストは、あるブランチで行われた変更を、別のブランチ（通常はメインブランチ）にマージすることを提案し、その変更内容をレビューし議論するための機能です。

◆不正解の選択肢の補足説明：

A. ディスカッション： GitHubリポジトリに関連する一般的な議論やアイデア共有のための機能であり、コードの変更をマージする目的ではありません。

B. プロジェクト： タスク管理や進捗追跡など、プロジェクトの計画と進行を管理するための機能であり、コードの変更をマージする目的ではありません。

D. イシュー： バグ報告、機能リクエスト、タスクの追跡など、特定の課題を管理するための機能であり、コードの変更をマージする目的ではありません。

問4

複数の README ファイルがある場合、次のどこが最初に表示されますか?

【選択肢】

A. docs

B. root

C. src

D. .gitHub

問4（解答）

D. .gitHub

【解説】

◆正解の理由：

GitHubでは、複数のREADMEファイルが存在する場合、リポジトリのルートディレクトリにあるREADMEよりも、`.github`ディレクトリ内のREADMEファイルが優先的に表示されます。これは、プロジェクトのコミュニティ関連情報や、特定のビューでの表示をカスタマイズするためによく利用されます。

◆不正解の選択肢の補足説明：

A. docs： READMEファイルの表示優先順位では、`.github`ディレクトリやルートディレクトリよりも後になります。

B. root： リポジトリのルートディレクトリも一般的なREADMEファイルの置き場所ですが、複数のREADMEファイルが存在する場合、`.github`ディレクトリ内のREADMEファイルよりも優先順位は低いです。

C. src： これは一般的なディレクトリ名であり、GitHubがREADMEファイルの表示優先順位を決定する際の特定のディレクトリ名ではありません。通常、ソースコードが配置される場所を指します。

問5

ブランチが Git の中核概念であるのはなぜですか?

【選択肢】

A. ブランチは、異なるブランチからの変更をメイン ブランチに自動的にマージするのに役立ちます。

B. ブランチにより、ディスク上にプロジェクトの物理コピーが作成され、データの冗長性とバックアップが確保されます。

C. ブランチ化により、他のブランチに影響を与えることなく、新しいアイデアを試したり変更を加えたりするための隔離された環境が作成されます。

D. 分岐は、Git リポジトリ内のファイルとフォルダーを整理するために必要です。

問5（解答）

C. ブランチ化により、他のブランチに影響を与えることなく、新しいアイデアを試したり変更を加えたりするための隔離された環境が作成されます。

【解説】

◆正解の理由：

C. Gitにおけるブランチの最も重要な役割は、メインの開発ラインに影響を与えることなく、新機能の追加やバグ修正、実験的な変更などを独立した環境で進められる点にあります。これにより、開発者は安心して新しいアイデアを試すことができます。

◆不正解の選択肢の補足説明：

A. Gitのブランチは、異なるブランチからの変更を自動的にマージする機能は持っていません。マージは通常、開発者が手動で指示するか、特定のツールを介して競合を解決しながら行われます。

B. ブランチは、プロジェクトのコード履歴内の特定の時点からの並行開発ラインを示すものであり、ディスク上にプロジェクトの物理コピーを新たに作成するわけではありません。Gitリポジトリ全体が、すべてのブランチとコミットの履歴を効率的に管理します。

D. ブランチは、Gitリポジトリ内のファイルやフォルダーを整理するためのものではありません。ファイルやフォルダーの整理は、リポジトリ内のディレクトリ構造や命名規則によって行われるものです。ブランチは開発の並行作業を管理するための概念です。

問6

Git と GitHub の違いは何ですか?

【選択肢】

A. Git はファイルの変更を追跡するためのコマンドライン ツールですが、GitHub は Git リポジトリで共同作業するためのプラットフォームです。

B. Git と GitHub は、バージョン管理とコラボレーションに使用される同じツールの別の名前です。

C. Git はクラウドベースのホスティング サービスであるのに対し、GitHub は分散バージョン管理システムです。

D. Git は集中バージョン管理システムですが、GitHub はクラウドベースのコラボレーション プラットフォームです

問6（解答）

A. Git はファイルの変更を追跡するためのコマンドライン ツールですが、GitHub は Git リポジトリで共同作業するためのプラットフォームです。

【解説】

◆正解の理由：

Git は、ローカルでファイルの変更履歴を追跡し、バージョンを管理するための分散型バージョン管理システム（DVCS）であり、主にコマンドラインで操作するツールです。一方、GitHub は、Git リポジトリをオンラインでホスティングし、プルリクエスト、課題追跡、コードレビューなどの機能を通じて、チームがコードで共同作業を行うためのウェブベースのプラットフォーム（サービス）です。

◆不正解の選択肢の補足説明：

B. Git と GitHub は同じツールではありません。Git は基盤となるバージョン管理システムであり、GitHub はそのGitを利用したオンラインサービスです。

C. Git はクラウドベースのホスティングサービスではなく、ローカルにインストールして使用するソフトウェアです。GitHub がクラウドベースのホスティングサービスにあたります。

D. Git は「分散」バージョン管理システムであり、「集中」型ではありません。集中型バージョン管理システムの代表例はSVNなどです。

問7

Git はどのような種類の操作に最適化されていますか?

【選択肢】

A. ローカルファイル操作とオフライン作業

B. リモートコラボレーションとリアルタイム編集

C. クラウドベースの操作と同期

D. Web 開発とブラウザベースのコーディング

問7（解答）

A. ローカルファイル操作とオフライン作業

【解説】

◆正解の理由：

Gitは分散型バージョン管理システムであり、各開発者はコードの完全なリポジトリをローカルに持つことができます。これにより、インターネット接続がない状態でも、ファイルの変更履歴のコミット、ブランチの作成、マージなどのほとんどのバージョン管理操作をローカルで完結させることができます。

◆不正解の選択肢の補足説明：

B. リモートコラボレーションとリアルタイム編集： Gitはリモートでのコラボレーションを強力にサポートしますが、これは「非同期」なコード共有（プッシュ/プル）によるものです。Googleドキュメントのような「リアルタイム」での共同編集には最適化されていません。

C. クラウドベースの操作と同期： Git自体はローカルで動作するツールであり、GitHubのようなサービスがGitリポジトリをクラウドでホストすることでクラウド連携を実現します。Gitそのものがクラウドベースの操作に特化しているわけではありません。

D. Web 開発とブラウザベースのコーディング： GitはWeb開発で広く利用されますが、特定の開発分野に限定されるものではなく、あらゆるコードのバージョン管理に利用できます。また、ブラウザベースのコーディング環境（Web IDEなど）はGitとは別のレイヤーで提供される機能です。

問8

GitHub はどのバージョン管理システムの上に構築されていますか?

【選択肢】

A. Subversion

B. Mercurial

C. Git

D. Perforce

問8（解答）

C. Git

【解説】

◆正解の理由：

GitHubは、分散型バージョン管理システムであるGitを基盤として、プロジェクトのコードホスティング、共同開発、バージョン管理機能を提供するWebサービスです。

◆不正解の選択肢の補足説明：

A. 転覆 (Subversion, SVN)： 集中型バージョン管理システムの一つで、Gitが普及する以前に広く使われていましたが、GitHubの基盤ではありません。

B. マーキュリアル (Mercurial)： Gitと同様に分散型バージョン管理システムの一つですが、GitHubはMercurialではなくGitの上に構築されています。

D. パーフォース (Perforce)： 集中型バージョン管理システムの一つで、特に大規模な開発や特定の業界で利用されますが、GitHubの基盤ではありません。

問9

新しいブランチを作成して切り替えるには、どのようなコマンドを入力する必要がありますか?

【選択肢】

A. git checkout -n newBranchName

B. git checkout -b newBranchName

C. git checkout newBranchName

D. git branch newBranchName

問9（解答）

B. git checkout -b newBranchName

【解説】

◆正解の理由：

git checkout -b newBranchName コマンドは、指定した名前で新しいブランチを作成し、同時にその新しいブランチへ作業ディレクトリを切り替えるためのショートカットです。-b オプションは、新しいブランチの作成 (branch) とそのブランチへの切り替え (checkout) の両方を一度に行います。

◆不正解の選択肢の補足説明：

A. git checkout -n newBranchName： Gitコマンドにおいて、git checkout に -n オプションは存在しません。これは誤ったコマンド構文です。

C. git checkout newBranchName： このコマンドは、既に存在する newBranchName というブランチへ切り替えるために使用されます。新しいブランチを作成する機能はありません。もし指定したブランチが存在しない場合、エラーが発生します。

D. git branch newBranchName： このコマンドは、指定した名前の新しいブランチを作成するだけです。作成後にそのブランチへ切り替えるには、別途 git checkout newBranchName コマンドを実行する必要があります。問題は「作成して切り替える」両方を求めているため、このコマンドだけでは不十分です。

問10

GitHub Enterprise Server (GHES) と GitHub Enterprise Cloud の違いは、GHES が次のとおりであることです。

【選択肢】

A. SAML シングル サインオンによる認証と、SAML または SCIM によるアクセス プロビジョニングが含まれます。

B. Windows と Linux の両方に展開できます。

C. レート制限を有効にできません。

D. は、組織がインフラストラクチャを完全に制御できるセルフホスト型ソリューションです。

問10（解答）

D. は、組織がインフラストラクチャを完全に制御できるセルフホスト型ソリューションです。

【解説】

◆正解の理由：

GitHub Enterprise Server (GHES) は、お客様が自身のデータセンターやプライベートクラウドにデプロイし、インフラストラクチャの管理と制御を完全に自社で行うセルフホスト型のソリューションです。これにより、セキュリティ、コンプライアンス、ネットワーク構成などを組織の要件に合わせて柔軟に設定できます。一方、GitHub Enterprise CloudはGitHubがホスト・管理するサービスです。

◆不正解の選択肢の補足説明：

A. SAML シングル サインオンによる認証と、SAML または SCIM によるアクセス プロビジョニングが含まれます。

- GitHub Enterprise ServerもGitHub Enterprise Cloudも、どちらもSAMLによるシングルサインオン認証とSCIMによるユーザープロビジョニングをサポートしています。これはGHES固有の違いではありません。

B. Windows と Linux の両方に展開できます。

- GitHub Enterprise Serverは仮想アプライアンスとして提供され、Hyper-V、VMware、OpenStack KVM、AWS、Azure、GCPなどの仮想化プラットフォーム上にデプロイされます。このアプライアンスはLinuxベースであり、Windowsオペレーティングシステムに直接展開することはできません。

C. レート制限を有効にできません。

- GitHub Enterprise Serverでは、管理者がAPIアクセスなどのレート制限を設定できます。GitHub Enterprise Cloudにもレート制限は適用されます。これはGHESとGitHub Enterprise Cloudのどちらでも利用可能な機能であり、GHESが有効にできないという記述は誤りです。