問4　Web アプリケーションプログラムに関する次の記述を読んで，設問に答えよ。

A 社は，加工食品の製造・販売を行う従業員 500 名の会社である。問屋や直販店からの注文の受付に，商品の注文と在庫を管理するシステム（以下，業務システムという）を利用している。業務システムは，A 社内に設置したサーバ上に構築されている。

このたび，販売拡大を目指して，インターネットを使ったギフト販売を行うことになり，個人顧客から注文を受けるための Web システム（以下，Web 受注システムという）を構築することになった。

A 社は IT ベンダーの B 社との間で開発の委託契約を締結し，両社は Web 受注システムの開発に着手した。

〔Web 受注システムの要件〕

Web 受注システムの要件を表 1 に示す。

表 1　Web 受注システムの要件

No.	要件名	要件内容
1	機能	個人顧客が利用できる機能：商品検索，在庫照会，注文，決済，注文変更，注文キャンセル，注文照会，配送照会，ユーザー登録，ユーザー情報変更，パスワード変更，退会である。これら機能全てをアプリケーション（以下，AP という）サーバに実装する。 その他の機能：（省略）
2	アクセス方式	Web ブラウザからインターネット経由でアクセスして利用する。
3	想定ユーザー	個人顧客
4	想定ユーザー数	登録ユーザーの数が 100,000 までを想定
5	重要情報[1)]に該当するデータ項目	氏名，住所，電話番号，メールアドレス，パスワード，銀行口座情報，決済情報
6	商品数	1,000
7	想定トランザクション数	注文：1,000 件／日 注文変更・注文キャンセル：各 30 件／日 注文照会・配送照会：各 2,000 件／日
8	稼働時間	24 時間 365 日。ただし，メンテナンス時間は除く。
9	メンテナンス時間	毎週月曜日 0:00～5:00。ただし，緊急の脆弱(ぜい)性修正プログラム適用など，他の日時に臨時でメンテナンスを実施する場合もある。
10	稼働率目標	99.9%。ただし，メンテナンス時間は除く。
11	開発体制	A 社と B 社が協働で開発する。
12	開発言語／DBMS	Java／RDBMS

表1　Web受注システムの要件（続き）

No.	要件名	要件内容
13	システム基盤	APサーバ，バッチサーバ，ログサーバは，IaaS上に構築する。Web受注システムのデータベース（以下，データベースをDBという）は，クラウドサービスのマネージドDBを利用する。 本番環境は，本番APサーバ，本番バッチサーバ，本番ログサーバ，本番DBで構成する。 開発環境は，開発APサーバ，開発バッチサーバ，開発ログサーバ，開発DBで構成する。重要情報は保管されない。
14	サーバOS	APサーバ，バッチサーバ，ログサーバのOSはLinuxを使用する。
15	APログ[2]	APサーバのプログラム及びバッチサーバのプログラムはAPログをログサーバに転送し，ログサーバはAPログをテキストファイル形式で保存する。
16	APサーバの標準出力と標準エラー出力	APサーバの標準出力と標準エラー出力は，リダイレクトしてAPサーバの/var/log/serverlogディレクトリ配下のテキストファイルに出力する。 なお，/var/log/serverlogディレクトリのオーナーはwebappuserであり，パーミッションは774[3]とする。その配下のテキストファイルのオーナーはwebappuserであり，パーミッションは664[3]とする。
17	システム運用	システム運用はB社に委託し，システム運用担当者はB社の要員とする。 重要情報の取扱いは重要情報取扱運用者だけとし，重要情報取扱運用者はA社での役職が管理職以上の要員とする。 各サーバ及び各DBの管理はシステム管理責任者が行い，システム管理責任者はA社の情報システム部の管理職とする。
18	システムのユーザーと役割	(1)　個人顧客 Web受注システムのAPサーバで注文，決済などを行う。 (2)　システム運用担当者 本番APサーバ及び本番バッチサーバの稼働を監視する。バッチ処理が異常終了したときは，手動で再実行する。これら以外のサーバについては，統合監視システムの画面から死活監視だけを行う。重要情報にアクセスしてはならない。 (3)　重要情報取扱運用者 本番環境に保管されている重要情報を参照し，個人顧客からの問合せに対応する。 (4)　システム開発者 開発環境においてプログラムの開発・保守を行う。障害発生時は，本番ログサーバにアクセスして障害原因を調査する。重要情報にアクセスしてはならない。 (5)　システム管理責任者 各サーバのOS，ミドルウェアの脆弱性修正プログラムの適用などのメンテナンス作業を行う。各サーバのOSアカウントを管理する。各DBのアカウント管理を行う。
19	パスワードの保存	パスワードは，CRYPTREC暗号リスト（令和5年3月30日版）の電子政府推奨暗号リストに記載されているハッシュ関数でハッシュ化してDBに保存する。

注[1]　A社では，扱う情報を“重要情報”と“その他の情報”に分類している。

注[2]　システム稼働時に出力され，システム障害の際に，システム開発者が障害原因調査のために確認するファイルである。

注[3]　chmodコマンドの絶対モードでLinuxのパーミッションを設定する。

〔Web 受注システムの設計〕

A 社と B 社は Web 受注システムを設計した。

Web 受注システムのサーバで定義される OS アカウントの一覧を表 2 に，所属グループとその権限を表 3 に示す。

表 2　OS アカウントの一覧

No.	ユーザーID	所属グループ	OS アカウントが定義されるサーバ	説明
1	root	root	（省略）	システム管理責任者が利用する。
2	operator	operation	（省略）	システム運用担当者が利用する。
3	personal	personal	（省略）	重要情報取扱運用者が利用する。
4	developer	develop	（省略）	システム開発者が利用する。
5	batchappuser	operation	本番バッチサーバ	データ連携機能 [1)]の各プログラムの実行に利用される。
6	webappuser	personal	本番 AP サーバ	AP サーバのプログラムの実行に利用される。

注記　root，operation，personal，develop という所属グループは各サーバに定義されている。
注 [1)]　業務システムと Web 受注システムがデータ連携を行うための機能である。

表 3　所属グループとその権限

No.	所属グループ	権限
1	root	特権ユーザーである。全てのアクセス権がある。
2	operation	一般ユーザー権限である。本番 AP サーバと本番バッチサーバへのアクセス権がある。
3	personal	一般ユーザー権限である。本番環境へのアクセス権がある。
4	develop	一般ユーザー権限である。開発環境と本番ログサーバへのアクセス権がある。

注記　Web 受注システムでは，OS アカウントの権限を所属グループ単位で管理する。

業務システムと Web 受注システムは，CSV 形式のデータ連携用ファイル（以下，CSV ファイルという）でデータ連携を行う。1 時間ごとに業務システムのバッチサーバと Web 受注システムのバッチサーバにおいて CSV ファイルを作成し，HTTPS で他方のバッチサーバに送信し，他方のバッチサーバでは受信した CSV ファイルを保存する。保存した CSV ファイルを使用して Web 受注システム又は業務システムの DB に対して更新処理を実行する。更新処理後の CSV ファイルは，障害発生に備えて 1 週間保存する。

データ連携機能のプログラム一覧を表 4 に示す。

表4　データ連携機能のプログラム一覧

No.	プログラム名	実行するサーバ	概要
1	バッチ処理管理1	Web 受注システムのバッチサーバ	Web 受注システムの各バッチ処理のプログラムの起動，監視などを行う。
2	バッチ処理管理2	業務システムのバッチサーバ	業務システムの各バッチ処理のプログラムの起動，監視などを行う。
3	注文データ CSV 出力バッチ処理	Web 受注システムのバッチサーバ	Web 受注システムの DB 内の注文テーブルから注文データを取得し，CSV ファイルに出力する。
4	注文データ CSV 取込みバッチ処理	業務システムのバッチサーバ	保存された CSV ファイルを読み込んで，業務システムの DB を更新する。
5	在庫データ CSV 出力バッチ処理	業務システムのバッチサーバ	業務システムの DB 内の在庫テーブルから在庫データを取得し，CSV ファイルに出力する。
6	在庫データ CSV 取込みバッチ処理	Web 受注システムのバッチサーバ	保存された CSV ファイルを読み込んで，Web 受注システムの DB を更新する。
7	データ送信1バッチ処理	Web 受注システムのバッチサーバ	CSV ファイルを業務システムのバッチサーバに HTTPS で送信する。
8	データ送信2バッチ処理	業務システムのバッチサーバ	CSV ファイルを Web 受注システムのバッチサーバに HTTPS で送信する。
9	データ受信1バッチ処理	Web 受注システムのバッチサーバ	受信した CSV ファイルを Web 受注システムのバッチサーバの指定されたディレクトリに保存する。
10	データ受信2バッチ処理	業務システムのバッチサーバ	受信した CSV ファイルを業務システムのバッチサーバの指定されたディレクトリに保存する。

表4のうち，No.3 のプログラムの内容を図1に示す。

・注文テーブルの連携済フラグ[1]が 0 である注文データを，CSV ファイルとして平文で/var/data ディレクトリに出力する。なお，/var/data ディレクトリのオーナーは batchappuser で，パーミッションは 770 [2]とする。CSV ファイルのオーナーは batchappuser で，パーミッションは 660 [2]とする。
・注文テーブルの内容は，次のとおりである。
注文 ID [3]，注文番号，注文ユーザーID，注文日時，決済金額，銀行コード，銀行支店コード，預金種別，銀行口座番号，銀行口座氏名，注文ステータス，お届け先郵便番号，お届け先住所，お届け先電話番号，お届け先氏名，送り主郵便番号，送り主住所，送り主電話番号，送り主氏名，連携済フラグ

注[1]　“ 0 ” は CSV ファイル出力前であることを，“ 1 ” は CSV ファイル出力後であることを示す。
注[2]　chmod コマンドの絶対モードで Linux のパーミッションを設定する。
注[3]　主キーである。

図1　No.3 のプログラムの内容（抜粋）

Web 受注システムの開発が進み，結合テスト前に，A 社は，設計書とソースコードのセキュリティレビューを，セキュリティ専門会社の C 社に委託した。C 社の情報処理安全確保支援士（登録セキスペ）の E 氏は，セキュリティレビューを実施した。

〔データ連携機能のセキュリティレビュー〕

E 氏は，表 2～4 及び図 1 の内容では表 1 の要件を満たしておらず，［ a ］が CSV ファイルを閲覧できてしまうという問題を発見した。また，CSV ファイルには重要情報が記録されるので，本番バッチサーバにアクセスできる者が不正に閲覧するリスクを軽減するための保険的対策も併せて実施することを提案した。具体的には，次のように提案した。

(1) 問題に対しては，表 2 の batchappuser について，所属グループを［ b ］に変更する。

(2) 保険的対策としては，表 4 の No.3 のプログラムに暗号化を行う処理を追加し，表 4 の No.［ c ］のプログラムに復号を行う処理を追加する。

A 社は，E 氏の提案どおり修正することにした。

〔ユーザー登録機能のセキュリティレビュー〕

ユーザー登録機能は，UserData クラスによって実現している。UserData クラスのプログラム仕様を図 2 に，UserData クラスのソースコードを図 3 に示す。

・addUser メソッドは，データをユーザーマスターテーブルに挿入する。
・各インスタンス変数は，ユーザーマスターテーブルの各レコードに対応し，画面から入力された値を String 型で保持する。
・ユーザーマスターテーブルの列名は，次のとおりである。
ユーザーOID [1]，ユーザーID，パスワード [2]，氏名，郵便番号，住所，電話番号，メールアドレス，作成日時，更新日時

注 [1] 主キーである。オブジェクト ID であり，データを一意に識別する文字列が格納される。
注 [2] パスワードのハッシュ値が格納される。

図 2 UserData クラスのプログラム仕様（抜粋）

```
  (省略)  //package 宣言, import 宣言など
 1: public UserData(HttpServletRequest request) {
 2:   this.userId = request.getParameter("userId");
 3:   this.password = request.getParameter("password");
      (省略)  //入力値チェックなど
 4:   try {
 5:     MessageDigest mdObj = MessageDigest.getInstance("SHA-1");
 6:     byte[] hashByte = mdObj.digest(this.password.getBytes());
 7:     this.password = String.format("%x", new BigInteger(1, hashByte));
 8:   } catch (NoSuchAlgorithmException e) {
 9:     log.debug("error:" + e);
10:   }
      (省略)
11: }
    //引数 conn は DB コネクションオブジェクトを示す。
12: public void addUser(Connection conn) {
13:   PreparedStatement psObj;
14:   String sql = "INSERT INTO USER_MASTER" +
15:                "(USER_OID, USER_ID, PASSWORD, USER_NAME, ZIP_CODE" +
                   (省略);
16:   try {
17:     psObj = conn.prepareStatement(sql);
18:     psObj.setString(1, this.userOid);
19:     psObj.setString(2, this.userId);
20:     psObj.setString(3, this.password);
        (省略)
        //次の 2 行はデバッグログの出力
21:     System.out.println("SQL:" + sql);
22:     System.out.println("InsertData:" + this.toString());
        //次の 2 行はログサーバへの AP ログの出力
23:     log.debug("SQL:" + sql);
24:     log.debug("InsertData:" + this.toString());
25:     psObj.execute();
26:     conn.commit();
27:   } catch (SQLException e) {
        (省略)  //例外処理
28:   }
      (省略)
29: }
    (省略)
```

注記　log.debug()は，引数の文字列をログサーバに送信するメソッドである。

図 3　UserData クラスのソースコード（抜粋）

E氏は，図3のソースコードについて，次のように指摘した。

・パスワードからハッシュ値を得るためのハッシュ関数が，表1の要件を満たしていない。

・今後，メンテナンスなどで実行環境を変更した場合に，［ d ］行目で［ e ］が発生すると，25，26行目では，パスワードが平文でユーザーマスターテーブルに保存されてしまう。

・①システム運用担当者とシステム開発者が，要件でアクセスが禁止されている情報にアクセスできてしまう。

・利用するAPサーバの実装では，変数psObjの指すメモリ領域においてメモリリークが発生する可能性がある。

E氏の指摘を受け，システム開発者は，UserDataクラスのソースコードを修正した。修正後のUserDataクラスのソースコードを図4に示す。

```
   (省略)  //package 宣言, import 宣言など
 1: public UserData(HttpServletRequest request) {
 2:   this.userId = request.getParameter("userId");
 3:   this.password = request.getParameter("password");
      (省略)  //入力値チェックなど
 4:   try {
 5:     MessageDigest mdObj = MessageDigest.getInstance("  f  ");
 6:     byte[] hashByte = mdObj.digest(this.password.getBytes());
 7:     this.password = String.format("%x", new BigInteger(1, hashByte));
 8:   } catch (NoSuchAlgorithmException e) {
 9:     log.debug("error:" + e);
        //回復不能な例外発生
10:     [ g ];
11:   }
      (省略)
12: }
    //引数 conn は DB コネクションオブジェクトを示す。
13: public void addUser(Connection conn) {
14:   PreparedStatement psObj = null;
15:   String sql = "INSERT INTO USER_MASTER" +
16:                "(USER_OID, USER_ID, PASSWORD, USER_NAME, ZIP_CODE" +
                   (省略);
```

図4　修正後のUserDataクラスのソースコード（抜粋）

```
17:   try {
18:     psObj = conn.prepareStatement(sql);
19:     psObj.setString(1, this.userOid);
20:     psObj.setString(2, this.userId);
21:     psObj.setString(3, this.password);
        (省略)
22:     UserData userMaskDataObj = this.maskUserData(this);
        //次の2行はログサーバへのAPログの出力
23:     log.debug("SQL:" + sql);
24:     log.debug("InsertData:" + userMaskDataObj.toString());
25:     psObj.execute();
26:     conn.commit();
27:   } catch (SQLException e) {
        (省略)  //例外処理
28:   } [    h    ] {
29:     if (psObj != null) {
30:         try {
31:           psObj.close();
32:         } catch (SQLException e) {
            (省略)  //例外処理
33:         }
34:     }
35:   }
      (省略)
36: }
37: private UserData maskUserData(UserData inUserData) {
      (省略)  //UserData内の重要情報を含む変数の値を ＊ に置換する。
38:   return userMaskDataObj;
39: }
    (省略)
```

図4　修正後のUserDataクラスのソースコード（抜粋）（続き）

図4のソースコードについて，E氏は，セキュリティレビューを再度実施した。

E氏は，図4のソースコードでは，レインボーテーブル攻撃を受けたときに攻撃が成立してしまうので，図2の仕様及び②図4のソースコードの6，7行目を修正すべきであると指摘した。

A社は，E氏の指摘の対応を完了した。その後，テストを実施し，Web受注システムをリリースした。

設問1　〔データ連携機能のセキュリティレビュー〕について答えよ。

(1)　本文中の［　a　］に入れる適切な字句を，解答群の中から選び，記号で答えよ。

解答群

ア　システム運用担当者

イ　システム運用担当者とシステム開発者

ウ　システム開発者

エ　システム開発者と重要情報取扱運用者

オ　重要情報取扱運用者

(2)　本文中の［　b　］に入れる適切な所属グループを，表3中から選び答えよ。

(3)　本文中の［　c　］に入れる適切なプログラムを，表 4 中から選び，No. 列の番号で答えよ。

設問2　〔ユーザー登録機能のセキュリティレビュー〕について答えよ。

(1)　本文中の［　d　］に入れる適切な行番号を，図3中から選び，答えよ。

(2)　本文中の［　e　］に入れる適切な字句を答えよ。

(3)　本文中の下線①について，システム運用担当者とシステム開発者が，アクセスが禁止されているのにアクセスできてしまう情報は何か。図2中のユーザーマスターテーブルの列名で，それぞれ全て答えよ。また，その情報が出力される場所を，解答群の中から選び，それぞれ記号で答えよ。

解答群

ア　開発ログサーバの AP ログを保存したテキストファイル

イ　本番 AP サーバの/sbin ディレクトリ配下のバイナリファイル

ウ　本番 AP サーバの/var/data ディレクトリ配下の CSV ファイル

エ　本番 AP サーバの/var/log/serverlog ディレクトリ配下のテキストファイル

オ　本番ログサーバの AP ログを保存したテキストファイル

(4)　図4中の［　f　］に入れる適切な字句を答えよ。

(5)　図4中の［　g　］に入れる適切な処理を，ソースコード又は具体的な処理内容のいずれかで答えよ。

(6) 図4中の［ h ］に入れる適切なソースコードを答えよ。

(7) 本文中の下線②について，図4の6，7行目をどのように修正すればよいか。修正後の適切なソースコードを解答群の中から選び，記号で答えよ。ここで，変数 salt には，addUser メソッドの呼出しごとに異なる 32 バイトの固定長文字列が入っているものとし，ユーザーマスターテーブルの定義に変更はないものとする。

解答群

ア	byte[] hashByte = mdObj.digest((salt + this.password).getBytes()); this.password = salt + String.format("%x", new BigInteger(1, hashByte));
イ	byte[] hashByte = mdObj.digest((salt + this.password).getBytes()); this.password = String.format("%x", new BigInteger(1, hashByte));

ウ	byte[] hashByte = mdObj.digest(this.password.getBytes());
	byte[] saltByte = mdObj.digest(salt.getBytes());
	this.password = String.format("%x", new BigInteger(1, hashByte)) + String.format("%x", new BigInteger(1, saltByte));

エ	byte[] hashByte = mdObj.digest(this.password.getBytes()); this.password = salt + String.format("%x", new BigInteger(1, hashByte));

オ	byte[] hashByte = mdObj.digest(this.password.getBytes());
	this.password = String.format("%x", new BigInteger(1, hashByte));
	byte[] saltHashByte = mdObj.digest((salt + this.password).getBytes());
	this.password = String.format("%x", new BigInteger(1, saltHashByte));