問1　Webサイトのセキュリティに関する次の記述を読んで，設問1～3に答えよ。

M社は，従業員数200名の小売業である。コーポレートサイトであるWebサイトA（URLは，https://site-a.m-sha.co.jp/）と，自社の特定のブランドを取り扱うECサイト（以下，ブランドサイトという）を複数運営している。現在運営しているブランドサイトは，WebサイトBからWebサイトFの五つである。Webサイトの開発や運用は自社の開発部で行っている。

Webサイト Aは，ブランドサイト全体のポータルサイトでもあり，各ブランドのキャンペーン情報などを掲載している。会員専用の機能は有していない。

Webサイト B（URLは，https://site-b.m-sha.co.jp/）は，ブランド Bの商品を扱うECサイトで，会員数は10万名である。WebサイトBでは，Cookieを利用したセッション管理を行っている。

会員情報は，各ブランドサイトで個別に管理している。

〔各ブランドサイトからWebサイトAへの情報連携〕

今回，各ブランドサイトの売上数を基にした，ブランド別の売れ筋商品情報を，Webサイト A上で表示するとともに，希望があれば，各ブランドサイトの会員に電子メールでも定期的に配信することにし，そのために売れ筋商品情報及び会員情報を取得する機能（以下，情報連携機能という）を実装することにした。具体的な機能は次のとおりである。

機能1　WebサイトAが各ブランドサイトの売れ筋商品情報を取得する。

機能2　希望する会員に電子メールを配信するために，Webサイト Aは，当該会員の会員情報を取得する。

なお，配信の申込みは，Webサイト A上で行う。

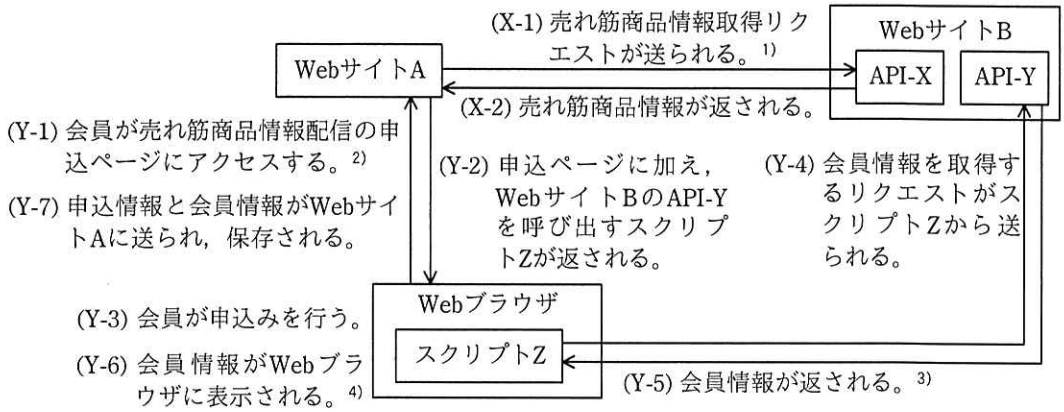
情報連携機能の実装は，開発部の C さんが中心になって進めることになった。まず初めに Webサイト B から Webサイト A への情報連携を行うために，次の二つの Web API を Webサイト B に実装することにした。

・Webサイト Bの売れ筋商品情報を取得可能とするための Web API（以下，API-X という）

・Webサイト Bの会員情報を取得可能とするための Web API（以下，API-Y という）

なお，Web API で受け渡されるデータは，JSON（JavaScript Object Notation）形式にする。C さんは，API-Y からブランドサイトの会員情報を取得する際，配信を希望する会員の同意を得たいと考えた。そこで，会員情報の取得には，会員の Web ブラウザを経由して行う方式を採用することにした。

Web サイト B から Web サイト A への情報連携機能を図 1 に示す。



注 1) API-X では，Web サイト A からだけアクセスできるように，接続元の IP アドレスを制限する。

2) 会員が Web サイト B にログインした状態のときにアクセスする。

3) 会員情報が得られない場合，エラーを返す。

4) （Y-5）でエラーが返ってきた場合，Web サイト B にログイン後再度操作を行うよう促す。

図 1　Web サイト B から Web サイト A への情報連携機能

〔情報連携機能の実装についての検討〕

スクリプト Z は，［ a ］ポリシによって，［ b ］，［ c ］，［ d ］のいずれかが異なるリソースへのアクセスが制限される。そこで，C さんは，この制限をう回するために JSONP（JavaScript Object Notation with Padding）を用いることを開発部の D 課長に提案した。次は，その時の会話である。

JavaScriptにおけるオブジェクトの表記法を応用したデータ形式であるJSONに、関数呼び出しなどのコードを付加したもの。
Webブラウザが表示しているWebページとは別のドメイン（クロスドメイン）のデータを呼び出して利用する場合などに用いられる
ただし、Same-Originポリシーが適用されません

C さん：API-Y からの会員情報の取得に JSONP を用いるつもりです。

D 課長：JSONP は，アクセス先を制限する機能をもたないので，その実装では問題がある。例えば，まず，会員情報を窃取するように攻撃者がスクリプト Z を変更して，攻撃者の Web サイトのページに置く。次に，被害者に①特定の操作をさせた上で，そのページにアクセスさせると，攻撃者が被害者の

会員情報を窃取できてしまう。

C さん：JSONP の代わりに何の技術を用いればよいでしょうか。

D 課長：CORS（Cross-Origin Resource Sharing）を用いるのがよいだろう。

〔CORS の概要〕

CORS とは，ある Web サイトから他の Web サイトへのアクセスを制御することができる仕組みである。XMLHttpRequest を使って“https://test2.example.com/test”にリクエストを送るスクリプトの例を図 2 に示す。

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://test2.example.com/test', true);
xhr.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
xhr.send(null);
```

図 2　XMLHttpRequest を使ったスクリプトの例

Web ブラウザが“https://test1.example.com/”にアクセスし，図 2 のスクリプトを含むページを読み込んだとする。図 2 のスクリプトが実行されると，最初に Web ブラウザは“https://test2.example.com/test”にプリフライトリクエストと呼ばれるリクエストを送る。そうすると，実際のリクエスト（以下，メインリクエストという）で許可されるメソッド名やヘッダフィールド名などがレスポンスとして返る。その後，メインリクエストを送り，レスポンスが返る。この一連の動作を図 3 に，また，図 3 中の（iii）～（vi）のリクエストとレスポンスの先頭部分の例を図 4～図 7 に示す。

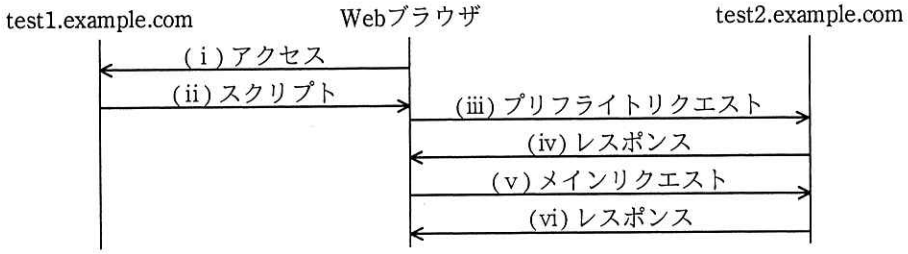


図 3　一連の動作

```
OPTIONS /test HTTP/1.1
Host: test2.example.com
Access-Control-Request-Method: GET [1)]
Access-Control-Request-Headers: x-requested-with [2)]
Origin: https://test1.example.com [3)]
```

注 1) Access-Control-Request-Method には，メインリクエストで利用したいメソッド名を指定する。
2) Access-Control-Request-Headers には，メインリクエストで利用したいヘッダフィールド名を指定する。
3) Origin は，スクリプトを含むページのオリジンであり，Web ブラウザが付与している。

図 4 （iii）のリクエストの先頭部分の例（抜粋）

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://test1.example.com [1)]
Access-Control-Allow-Methods: GET, POST, OPTIONS [2)]
Access-Control-Allow-Headers: x-requested-with [3)]
```

注 1) Access-Control-Allow-Origin には，Web サイトが許可するオリジンが返される。
2) Access-Control-Allow-Methods には，Web サイトが許可するメソッド名が返される。
3) Access-Control-Allow-Headers には，Web サイトが許可するヘッダフィールド名が返される。

図 5 （iv）のレスポンスの先頭部分の例（抜粋）

```
GET /test HTTP/1.1
Host: test2.example.com
X-Requested-With: XMLHttpRequest
Origin: https://test1.example.com
```

図 6 （v）のリクエストの先頭部分の例（抜粋）

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://test1.example.com
```

図 7 （vi）のレスポンスの先頭部分の例（抜粋）

また，CORS では通常，Web ブラウザは，スクリプトを読み込んだページのオリジンだけに Cookie や，ベーシック認証の情報を送る。図 2 では設定していないが，XMLHttpRequest のプロパティの withCredentials の値が true に設定されている場合，図 3 であれば，［ e ］の動作の際に，test2.example.com から発行された Cookie が送られる。

〔CORS を利用した実装〕

C さんは，スクリプト Z の実装に CORS を用いたときの一連の動作を検討し，表 1 にまとめた。

表 1　スクリプト Z の実装に CORS を用いたときの一連の動作

No.	内容
1	Web ブラウザは，Web サイト A の売れ筋商品情報配信の申込ページにアクセスする。
2	Web サイト A は，Web サイト B の API-Y を呼び出すスクリプト Z を含むページをレスポンスとして返す。
3	Web ブラウザは，会員が申込みを行うと，Web サイト B にプリフライトリクエストを送信する。プリフライトリクエストは，OPTIONS メソッドの呼出しであり，Origin ヘッダフィールドには“https://site-a.m-sha.co.jp”が設定されている。
4	API-Y は，送られてきたリクエストに Origin ヘッダフィールドが存在する場合，Access-Control-Allow-Origin ヘッダフィールドを付加し，レスポンスを返す。Access-Control-Allow-Origin ヘッダフィールドの値は，“ f ”である。Origin ヘッダフィールドが存在しない場合，エラーを返す。
5	Web ブラウザは， g と Access-Control-Allow-Origin ヘッダフィールドの値を照合し，アクセスが許可されていることを確認する。許可されている場合は，次の処理に進む。確認できない場合は，メインリクエストを送らずに終了する。
⋮	⋮
9	スクリプト Z は，受け取った JSON 形式の値を変数に格納し，表示する。さらに，受け取った値は Web サイト A に送られ，保存される。

C さんは，表 1 について D 課長に確認した。次は，その時の D 課長と C さんの会話である。

D 課長：今後，他のシステムでも CORS を利用することが考えられるので，コーディング規約も併せてまとめておきたい。Access-Control-Allow-Origin ヘッダフィールドに指定できるオリジンは一つだけなので，複数のオリジンからのアクセスを許可するような仕様であった場合に，No. 4 の内容では不十分である。Web API のプログラム内に，許可するオリジンのリストを用意しておく必要がある。プリフライトリクエスト又はメインリクエストが Web API に送られてきたときに，そのリクエスト中の h を， i と突合し， j した値があればその値を Access-Control-Allow-Origin ヘッダフィールドに設定するという内容もコーディング規約に含めればよ

いだろう。

C さん：分かりました。

C さんは，CORS の利用に関するコーディング規約をまとめ，表 1 をこれに合うように修正し，D 課長に再度確認した。修正後の内容で問題ないということだったので，C さんは実装を行った。

その後，セキュリティ専門業者に脆弱性診断を依頼し，脆弱性が検出されないことを確認した上で，情報連携機能をリリースした。その後，同様に残り四つのブランドサイトから Web サイト A への情報連携機能も実装した。

設問 1　〔情報連携機能の実装についての検討〕について，(1)～(3)に答えよ。

(1) 本文中の［ a ］に入れる適切な字句を答えよ。

(2) 本文中の［ b ］～［ d ］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 Cookie	イ FQDN
ウ Location ヘッダフィールド	エ Referer ヘッダフィールド
オ User-Agent ヘッダフィールド	カ 時刻
キ スキーム	ク ポート番号

(3) 本文中の下線①について，操作の具体的な内容を，20 字以内で答えよ。

設問 2　本文中の［ e ］に入れる適切な記号を，(iii)～(vi)の中から選び，答えよ。

設問 3　〔CORS を利用した実装〕について，(1)～(3)に答えよ。

(1) 表 1 中の［ f ］に入れる適切な URL を答えよ。

(2) 表 1 中の［ g ］に入れる適切な字句を，30 字以内で答えよ。

(3) 本文中の［ h ］，［ i ］に入れる適切な字句を，それぞれ 20 字以内で，本文中の［ j ］に入れる適切な字句を，5 字以内で答えよ。

〔 メ モ 用 紙 〕