問1　Webアプリケーションに関する次の記述を読んで，設問1～3に答えよ。

B社は，従業員数50名のインターネットサービス企業であり，インターネット上で各種の情報サービスを提供している。これらのサービスの実現に当たり，必要なアプリケーションを自社開発してきた。このたびB社では，画像ファイルを対象としたオンライン共有ストレージサービス（以下，Sサービスという）を立ち上げることにし，R部長を責任者としてアプリケーション開発に着手した。Sサービスの画面には広告主のサイトへのリンクを含んだバナー広告を掲載し，利用者からの利用料収入に加えて広告料収入の獲得も目指す。

〔Sサービスの要件定義〕

B社では，P主任を中心としてSサービスの要件を検討した。検討の結果，P主任はSサービスの要件案を図1に示すとおりまとめた。

(1) クライアント側はWebブラウザだけとし，専用アプリケーションは使用しない。プロトコルは，HTTPとHTTP over TLS（以下，HTTPSという）のどちらも使用できる。
(2) 利用者は，Sサービスの利用に先立ち，英字4文字に数字4文字を付加した全8文字の利用者IDの割当てを受ける。利用者種別は，契約者と閲覧者の2種類とする。
(3) 契約者になるためには利用料を支払う必要がある。閲覧者になるための料金は不要である。
(4) 利用者がSサービスにアクセスする際は，最初に利用者IDとパスワードによる認証が行われる。
(5) 契約者は，共有フォルダを作成でき，書込・削除・読取権限をもつ。さらに，任意の閲覧者に対して，その共有フォルダへの読取権限を付与することができる。
(6) 契約者が保存できる画像ファイルのサイズの合計は，契約条件で定める最大値を超えないように制限される。
(7) 画像ファイルのアップロード時には，画像ファイルに，注釈文を添付することができる。
(8) 利用者は，読取権限をもつ共有フォルダ内の画像ファイルを，小さな画像として一覧表示（以下，サムネイル表示という）でき，必要に応じてオリジナルの画像ファイルをダウンロードできる。
（以下，省略）

図1　Sサービスの要件案

この要件案は，レビューされた後，R部長によって承認された。

〔システム・ソフトウェア要件定義〕

その後，Q主任を中心としてシステム・ソフトウェア要件定義を実施した。その結果，SサービスはJavaサーブレットを利用したWebアプリケーションで構成するこ

とにした。画像処理プログラムについては C++で記述し，プログラム内では C ベースの画像処理ライブラリである V-lib を利用することにした。Java から C++プログラムを呼び出す際には，JNI（Java Native Interface）を利用することにした。

〔ソフトウェアの基本設計〕

システム・ソフトウェア要件定義が完了し，ソフトウェアの基本設計を開始した。その中で，V 君は Web アプリケーションのセッション管理の方式について検討した。次は，検討に際しての Q 主任と V 君との会話である。

Q 主任：HTTP セッション管理には，どのような方式を採用するつもりかな。

V 君　：セッション管理には，ログオン時にランダム文字列をセッション ID として割り当てた上で，クエリパラメタの中にセッション ID を格納する方式を採用するつもりです。

Q 主任：当社でも実績のある方式だね。ただ，今回は HTTPS だけでなく HTTP も使用できるようにするから，GET メソッドではなく POST メソッドを使用しておかないと，①HTTP ヘッダ内のフィールドが原因で，バナー広告に設定されたリンク先の Web サーバへセッション ID が漏えいすることが考えられるね。

更に基本設計が進む中で，実行形式ファイルがアップロードされるとウイルスの温床になりかねないという指摘があった。議論した結果，この指摘への対策の一つとして，アップロードするファイルの拡張子を画像ファイルのものに制限することとした。制限する方式として，V 君は，Web ブラウザにおいてスクリプトによって判定することを考えた。この方式を実装したスクリプトを含む HTML ファイル例を図 2 に示す。この方式案について相談を受けた Q 主任は，②契約者の操作によっては，拡張子が画像ファイルのものでないファイルがアップロードできてしまうという問題点を指摘し，制限する方式には③スクリプトによらない方式を検討するよう指示した。V 君はその後，異なる方式を提案し，採用に至った。

```
<form action="/sservice/UploadFile" method="POST" id="form_upload"
 enctype="multipart/form-data">
    <input type="file" name="upload" size="128">
    <input type="submit" name="button_submit" value="送信">
</form>
<script type="text/javascript">
    <!--
    var input = document.getElementById("form_upload");
    input.onsubmit = function () {
    （省略）/* アップロードするファイルの拡張子を調べて，画像ファイルかどうか判定し，画像でない
            ファイルの送信を停止する。*/
    }
//   -->
</script>
```

図２　HTML ファイル例（抜粋）

〔画像処理プログラムでの不具合〕

基本設計及び詳細設計が無事完了し，U 主任を中心としたグループでプログラムの作成を開始した。U 主任は，サムネイル表示の際に使用するために，様々なファイル形式の画像ファイルを縦横とも 128 ピクセルに変換する画像処理プログラム（以下，ピクセル数変換プログラムという）の作成を T 君に指示した。T 君は，最初に，複数のファイル形式の中からファイル形式 Z を選択して，ピクセル数変換プログラムの作成に取り掛かった。

このプログラムで取り扱う，ファイル形式 Z を図 3 に示す。ピクセル数変換そのものには，V-lib 内の vlibResize 関数を使用することにした。V-lib 内で定義され，vlibResize 関数の引数として使用する vlibMat 構造体の説明を図 4 に示す。また，vlibResize 関数の外部仕様を図 5 に示す。

ヘッダ部	
	ファイル形式識別子　2 バイト（固定文字列）
	ファイル形式バージョン番号　2 バイト（整数）
	列数　2 バイト（整数）
	行数　2 バイト（整数）
	各ピクセルのバイト数　2 バイト（整数）
ピクセルデータ (1 行目のピクセルデータが RGB の順に並び，続いて 2 行目以降のピクセルデータが並ぶ。行と行の間には区切り記号はない。)	

注記　整数はいずれも符号なしとする。

図３　ファイル形式 Z

```
typedef struct {
    char* data;    /* ピクセルデータへのポインタ，各ピクセルのサイズは 3 バイト(RGB)とする */
    int  cols;     /* 画像の列数 */
    int  rows;     /* 画像の行数 */
} vlibMat;
```

data が指す領域は，1 行目のピクセルデータが RGB の順に並び，続いて 2 行目以降のピクセルデータが並ぶ。
なお，各行の先頭を 256 バイト境界に配置するために，必要に応じてパディングをする。

data 領域のパディング例： 1 行当たり 128 ピクセル，384 バイトの場合

図 4　vlibMat 構造体の説明

```
int vlibResize(const vlibMat* src, vlibMat* dst)
  機能：src を基に，縦横ピクセル数を変換して dst に書き込む。
  引数：src：入力画像
        dst：出力画像
                変換後の横ピクセル数を dst->cols に，
                変換後の縦ピクセル数を dst->rows にセットして，
                dst->data が指す領域は，書込み可能領域として確保してから呼び出す。
  返却値：正常に完了すれば 0，正常に完了しなかった場合は-1 とする。
```

図 5　vlibResize 関数の外部仕様

T 君は，図 6 のピクセル数変換プログラムを作成し，U 主任によるソースコードレビューを受けた。そのソースコードレビューで，U 主任は，入力する画像ファイルの内容によっては 38 行目において [a] が発生し，その結果として 50 行目において [b] が発生することを指摘した。

なお，この処理系において，int 型は 4 バイト，short int 型は 2 バイト，char 型は 1 バイトである。また，char 型は特に記述がなければ符号なしである。

```
 1: #include <cstdlib>
 2: #include <iostream>
 3: #include <exception>
 4: #include "vlib.h"    /* 画像処理ライブラリ V-lib 用ヘッダファイル */
 5: （省略）/* その他，必要なヘッダファイルを読み込む。 */
 6: using namespace std;
 7:
 8: const int kIndexCols = 128;    /* サムネイル表示用列数 */
 9: const int kIndexRows = 128;    /* サムネイル表示用行数 */
10: const int kMaxMatData = 104857600;    /* 100M バイト */
11: const int kMaxInteger = 0x7fffffff;
12: const int kColors = 3;
13:
14: typedef struct {
15:     char ident0, ident1;
16:     unsigned short int f_version, cols, rows, colors;
17: } zHeader;  /* ファイル形式 Z のヘッダ部 */
18:
19: int resize_fileZ(FILE *fp, vlibMat* matBuf) throw(std::out_of_range, std::bad_alloc)
20: {
21:     /* fp はファイル形式 Z の画像ファイルへのファイルポインタ */
22:     /* matBuf はピクセル数変換後の画像を保存する構造体へのポインタ */
23:     /* matBuf->data が指す領域はこの関数内で確保し，呼出し側で解放する。 */
24:     int       bytesOfRow, bytesOfBuff;
25:     zHeader   fhBuf;
26:     char *    pixBuf;
27:     vlibMat   inBuf;
28:
29:     fseek(fp, 0L, SEEK_SET);    /* ファイル読込位置をファイルの先頭に移動 */
30:     /* ファイル fp から，zHeader のバイト数だけデータを読み込み，fhBuf に書き込む。 */
31:     /* 読込の結果，ファイル読込位置は読込バイト数だけ移動する。 */
32:     fread(&fhBuf, sizeof(zHeader), 1, fp);
33:     （省略）/* ファイル形式識別子とファイル形式バージョン番号をチェックする。 */
34:     if (fhBuf.colors != kColors) throw out_of_range("Illegal header");
35:     if (fhBuf.cols == 0 || fhBuf.rows == 0) throw out_of_range("Illegal header");
36:     bytesOfRow = fhBuf.cols * fhBuf.colors;
37:     bytesOfRow += 255 - ((bytesOfRow + 255) % 256); /* 256 バイト境界に調整 */
38:     bytesOfBuff = bytesOfRow * fhBuf.rows;
39:     if (bytesOfBuff > kMaxMatData) throw out_of_range("Image too large");
40:
41:     pixBuf = (char *)malloc(bytesOfBuff);
42:     if (pixBuf == NULL) throw bad_alloc();
43:     inBuf.data = pixBuf;
44:     inBuf.cols = fhBuf.cols;
45:     inBuf.rows = fhBuf.rows;
46:
```

図６　ピクセル数変換プログラム

```
47:     for (int i = 0; i < fhBuf.rows; i ++) {
48:         /* ファイル fp から，(fhBuf.cols * kColors) バイトだけデータを読み込み，*/
49:         /* pixBuf に書き込む。ファイル読込位置は読込バイト数だけ移動する。 */
50:         fread(pixBuf, kColors, fhBuf.cols, fp);
51:         pixBuf += fhBuf.cols * kColors;
52:         pixBuf += 255 - ((fhBuf.cols * kColors + 255) % 256); /* 256 バイト境界に調整 */
53:     }
54:     matBuf->data = (char *)malloc(((kIndexCols * kColors + 255) / 256) * 256 *
   kIndexRows);
55:     if (matBuf->data == NULL) {
56:         free(inBuf.data);
57:         throw bad_alloc();
58:     }
59:     matBuf->cols = kIndexCols;
60:     matBuf->rows = kIndexRows;
61:     if (vlibResize(&inBuf, matBuf) == -1) {
62:         free(inBuf.data);
63:         throw bad_alloc();
64:     }
65:     free(inBuf.data);
66:     return 0;
67: }
```

図 6　ピクセル数変換プログラム（続き）

T 君はこの指摘を受けて，図 6 の 37 行と 38 行の間に図 7 に示す行を追加し，再度 U 主任のソースコードレビューを受けた。ソースコードレビューの結果，指摘された問題点が修正されていること，他の部分に同種の問題点が存在しないことが確認された。

```
if ( [ c ] < [ d ] ) throw out_of_range("Image too large");
```

図 7　ピクセル数変換プログラムへの追加行

その後，全てのプログラムの作成とテストが完了して，B 社は S サービスの運用を開始することができた。

設問 1　〔ソフトウェアの基本設計〕について，(1)～(3)に答えよ。

(1)　本文中の下線①の HTTP ヘッダ内のフィールド名は何か。英字 10 字以内で答えよ。

(2) 本文中の下線②の契約者の操作とは何か。30 字以内で述べよ。

(3) 本文中の下線③の方式とはどのようなものか。30 字以内で述べよ。

設問 2 〔画像処理プログラムでの不具合〕について，(1)，(2) に答えよ。

(1) 本文中の [a] ，[b] に入れる適切な字句を，それぞれ 15 字以内で答えよ。

(2) U 主任が指摘した [a] が発生する入力ファイルに関する条件を，kMaxInteger という字句を用いて 60 字以内で述べよ。

設問 3 図 7 中の [c] ，[d] に入れる適切な式又は変数を答えよ。
なお，[c] ，[d] のどちらか一方は，kMaxInteger を含むこと。