問2　ソフトウェア開発における脆弱性対策に関する次の記述を読んで，設問 1～4 に答えよ。

V 社は，ディジタル機器の開発及びソフトウェアの受託開発を行う，従業員数 400 名の企業である。昨今，ソフトウェアの重大な脆弱性が相次いで報告されているので，V 社開発部の L 氏は，セキュリティ専門業者 S 社の情報セキュリティスペシャリストの K 氏から，ソフトウェア開発についてのアドバイスを受けることとした。

〔脆弱性及びその悪用〕

ソフトウェアの脆弱性の情報を組織間で一意に特定し共有できる仕組みが運用されている。例えば，脆弱性が報告されると，［　a　］識別子が付番され，日本国内では JPCERT/CC などが公表し注意喚起している。ただし，公表前に悪用されるものもあり，［　b　］脆弱性と呼ばれる。［　b　］脆弱性は，特定の組織を狙う攻撃，つまり［　c　］型攻撃でしばしば悪用される。

脆弱性の中でも，バッファオーバフロー脆弱性（以下，BOF 脆弱性という）は，最近開発されたソフトウェアにおいても数多く報告されている。BOF 脆弱性は，主に，スタックベース BOF 脆弱性とヒープベース BOF 脆弱性に分類される。ソフトウェアに BOF 脆弱性がある場合，当該ソフトウェアへの入力によって，①開発者が想定しないメモリ領域に書き込まれ，開発者の意図しない命令が実行されることがある。

今，Linux において実行可能ファイルが四つあるとする。図 1 は，それらが存在するディレクトリでの ls コマンドの出力である。実行権限の属性に“s”が表示されているファイルは，ファイルの所有者権限で実行されることを示している。四つのファイルのうち［　ア　］は，一般利用者 suzuki によって起動された場合でも root 権限で動作する。［　ア　］に BOF 脆弱性があると，外部の攻撃者やマルウェアによって，開発者の意図しない命令が root 権限で実行されてしまう。

```
suzuki@linux:~/temp$ ls -al
合計 40
drwxrwxr-x  2 suzuki  suzuki  4096  1月 18日 16:19 .
drwxr-xr-x 22 suzuki  suzuki  4096  1月 18日 15:54 ..
-rwxrwxr-x  1 root    root    7205  1月 18日 16:08 sample1
-rwsrwxr-x  1 root    root    7205  1月 18日 16:10 sample2
-rwxrwxr-x  1 suzuki  suzuki  7205  1月 18日 16:10 sample3
-rwsrwxr-x  1 suzuki  suzuki  7205  1月 18日 16:10 sample4
suzuki@linux:~/temp$
```

図1　ls コマンドの出力

〔ヒープベース BOF 脆弱性〕

K 氏によると，最近，ヒープベース BOF 脆弱性を悪用する攻撃の報告が増えてきているという。L 氏は，出荷前の V 社製品のソフトウェアについてヒープベース BOF 脆弱性がないか，K 氏のレビューを受けることにした。

図2のプログラム Y は，起動時に，第1引数は利用者 ID を，第2引数はパスワードを受け取る。このプログラム用にあらかじめ登録された“利用者 ID とパスワード”の組と引数で与えられた組を比較し，利用者認証する。“利用者 ID とパスワード”は，いずれも半角英数字，最小 6 文字最大 8 文字の文字列と仕様で定められている。

```
 1: #include <iostream>
 2: #include <cstring>
 3: (省略)
 4: #define UID_SIZE 8     // 利用者 ID の文字列の上限値
 5: #define PASS_SIZE 8    // パスワードの文字列の上限値
 6: (省略)
 7: using namespace std;
 8:
 9: void getPass(char *pass, char *uid)
10: {
11: (省略, uid で指定された利用者 ID を基に登録済パスワードを取得し pass に格納, 利用者
    ID が存在しない場合は長さ 0 の文字列を pass に格納)
12: }
13: (省略)
14:
15: int main(int argc, char **argv)
16: {
17:     static char *uid;
18:     static char *pass;
19:     (省略, 引数の個数をチェック)
20:     uid = new char[UID_SIZE+1];
21:     pass = new char[PASS_SIZE+1];
22:     getPass(pass, argv[1]);
23:     strcpy(uid, argv[1]);
24:
25:     if (strlen(pass) == 0 || strcmp(argv[2], pass) != 0) {
26:       cout << "認証失敗" << endl;
27:       (省略, uid を出力, 認証失敗時の処理)
28:     } else {
29:       cout << "認証成功" << endl;
30:       (省略, uid を出力)
31:     }
32: }
```

図 2　ヒープベース BOF 脆弱性のあるプログラム Y

プログラム Y にはヒープベース BOF 脆弱性があり，②引数によっては，利用者認証を回避される可能性があると，L 氏は K 氏に指摘された。ただし，K 氏によると，③このプログラム Y は引数が同じでも，実行環境によっては利用者認証を回避されないとのことだった。L 氏は，V 社の開発した他のプログラムについても確認することとした。

〔V 社の脆弱性対策〕

BOF 脆弱性は，メモリを直接操作することが可能なプログラム言語 C や C++などで発生する。その対策として，例えば，Windows ではハードウェア DEP（Data Execution Prevention）のようなデータ実行防止機能を適用することで，一部の BOF

脆弱性を悪用する攻撃を抑制できる。しかし，④プログラム Y での利用者認証を回避する攻撃に対しては有効に機能しない。そこで，V 社では，C や C++の利用が避けられない場合を除き，BOF 脆弱性が起きにくい他のプログラム言語を利用することとした。一般的には，開発時に，静的解析ツールを利用したり，通常の利用では想定しないデータを入力し，その応答から脆弱性を探す［ d ］検査を行うツールを利用したりして，脆弱性を洗い出すことも効果的である。また，V 社では，ソフトウェアのリリース後，脆弱性が発見された場合に備え，V 社製品の利用者に更新プログラムを提供するサイトを準備することとした。

設問 1　本文中の［ a ］～［ d ］に入れる最も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　CVE　　イ　ゼロデイ　　ウ　標的　　エ　ファジング

設問 2　脆弱性を悪用する攻撃について，(1)，(2)に答えよ。

(1)　本文中の下線①について，スタックベース BOF 脆弱性を悪用する攻撃の場合，関数呼出し時にスタックに必ず積まれるはずの，何の値を書き換えることによって攻撃が開始されるか。20 字以内で答えよ。

(2)　本文中の［ ア ］に入れるファイル名を全て答えよ。

設問 3　図 2 のプログラムの脆弱性について，(1)～(3)に答えよ。

(1)　本文中の下線②はどのような引数で利用者認証を回避されるか。引数の組を解答群の中から選び，記号で答えよ。

解答群

記号	第 1 引数	第 2 引数
ア	001（繰返し）0111111111	11111110
イ	001（繰返し）1111111111	11111110
ウ	011（繰返し）1111111101	11111101
エ	111（繰返し）1111111111	67891231
オ	123（繰返し）1231231231	67891231

(2) 利用者認証を回避される原因となるヒープベース BOF 脆弱性の存在箇所を，実際にバッファがオーバフローするコードの行番号で答えよ。

(3) (2)で示した行番号の行を差し替えて行う改修案として適切なものを解答群の中から全て選び，記号で答えよ。

解答群

ア `memcpy(uid, argv[1], strlen(argv[1])+1);`

イ `memcpy(uid, argv[1], UID_SIZE+1);`

ウ `pass = new char[PASS_SIZE+8];`

エ `strncpy(uid, argv[1], strlen(argv[1])+1);`

オ `strncpy(uid, argv[1], UID_SIZE+1);`

設問4 利用者認証の回避について，(1)，(2)に答えよ。

(1) 本文中の下線③について，利用者認証が回避されない理由を，攻撃のタイミングではなく実行環境の観点で，40 字以内で述べよ。

(2) 本文中の下線④について，有効に機能しない理由を，30 字以内で述べよ。