次の問9から問13までの5問については，この中から1問を選択し，答案用紙の選択欄の(選)をマークして解答してください。

なお，2問以上選択した場合には，はじめの1問について採点します。

問9　次のCプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

C

〔プログラムの説明〕

与えられたパスを絶対パスに変換する関数convertである。

階層構造をもつファイルシステムにおいて，ファイルやディレクトリを特定する文字列をパスという。ルートディレクトリを基準としたパスを絶対パスと呼び，“/”から始まり，各階層を“/”で区切っていく。与えられたパスがディレクトリのとき，最後の“/”はあってもなくてもよい。例えば，図のディレクトリeの絶対パスは“/a/d/e”又は“/a/d/e/”で示す。

一方，カレントディレクトリを基準としたパスを相対パスと呼び，相対パスを指定するときに階層を一つ上にたどる場合は“..”を用いる。例えば，図においてディレクトリcをカレントディレクトリにした場合，ファイルfile1.txtの相対パスは“../file1.txt”，ディレクトリeの相対パスは“../../d/e”又は“../../d/e/”となる。また，カレントディレクトリ自身は“.”又は“./”で示す。

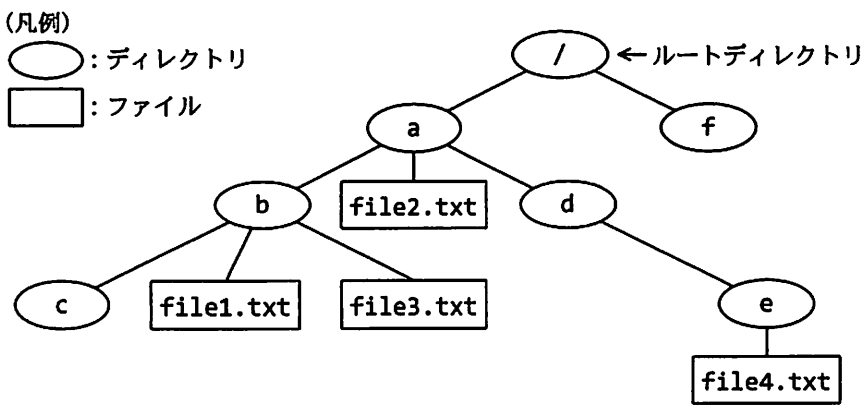

図　階層構造をもつファイルシステムの例

(1) 関数の仕様は，次のとおりである。

```
void convert(const char *path,
             const char *base,
             char *result);
```

引数：path　変換前のパス

base　カレントディレクトリの絶対パス

result　変換後の絶対パス

機能：path が相対パス表記であれば，base を基準にした絶対パス表記に変換し，result に格納する。path が絶対パス表記であれば，result には base に関係なく path をそのまま格納する。

返却値：なし。

ただし，result が参照する領域は，変換後の文字列を格納するのに十分であるとする。また，冗長なパス又はパスとして認識できない文字列が引数として与えられることはないものとする。

(2) ファイルシステム上に，指定されたディレクトリやファイルが実際に存在するかどうかのチェックは行わない。

(3) 変換例を表１に示す。

表１　変換例

path	base	result
../../b/c/	/a/d/e/	/a/b/c/
b/file1.txt	/a/	/a/b/file1.txt
c/	/a/b/	/a/b/c/
file1.txt	/a/b/	/a/b/file1.txt
./	/a/b/c/	/a/b/c/
/a/d/e/file4.txt	/a/b/c/	/a/d/e/file4.txt

(4) 次のライブラリ関数を用いる。

```
unsigned int strlen(const char *s);
```

機能：文字列 s の長さを計算する。

返却値：終端を示すナル文字に先行する文字の個数を返す。

```
int strcmp(const char *s1, const char *s2);
```

機能：文字列 s1 と文字列 s2 を比較する。

返却値：s1 と s2 が同一文字列の場合は 0，それ以外の場合は 0 以外を返す。

```
int strncmp(const char *s1, const char *s2, int n);
```

機能：文字列 s1 と文字列 s2 を先頭から n 文字，又はナル文字までを比較する。

返却値：比較した n 文字が同一の場合は 0 を，それ以外の場合（比較が途中で終了した場合も含む）は 0 以外を返す。

```
char *strcpy(char *s1, const char *s2);
```

機能：文字列 s1 に文字列 s2 をナル文字まで複写する。

返却値：s1

```
char *strncpy(char *s1, const char *s2, int n);
```

機能：文字列 s1 に文字列 s2 を n 文字複写する。s2 の長さが n 以上の場合は n 文字目までを複写し，n 未満の場合は残りをナル文字で埋める。

返却値：s1

〔プログラム〕

```
#include <string.h>

void convert(const char*, const char*, char*);

void convert(const char *path, const char *base, char *result){

   const char *pp, *bp;
   char *rp;
   int length;

   /* path が絶対パス表記の場合 */
   if(*path == '/'){
      [ a ];
      return;
   }
```

```
    /* path がカレントディレクトリの場合 */
    if(!strcmp(path, ".") || !strcmp(path, "./")){
        [ b ];
        return;
    }

    length = strlen(base);
    bp = base + length;    /* bp は文字列 base の終端を指す。 */
    if(*(bp - 1) == '/')
        --bp;

    /* path の先頭部にある".."又は"../"を解析することで,
       base のパス表記のうち，どこまで result と共通になるかを調べる。 */
    for(pp = path; *pp != '\0' && *pp == '.';){
        if(!strncmp(pp, "../", 3)){
            pp += 3;
            while(bp > base && *--bp != '/');
        }else if(!strncmp(pp, "./", 2)){
            pp += 2;
        }else if(!strncmp(pp, "..\0", 3)){
            pp += 2;
            while(bp > base && *--bp != '/');
        }else{
            break;
        }
    }
    /* base のパス表記と共通な部分を result に複写する。 */
    length = [ c ];
    strncpy(result, base, length);

    rp = [ d ];
    *rp++ = '/';

    /* path の文字列のうち，先頭部分の"./"や"../"を除いた残りの
       部分(pp が指す文字列)を，result の文字列に追加する。 */
    strcpy(rp, pp);
    return;
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア	strcpy(base, path)	イ	strcpy(base, result)
ウ	strcpy(path, base)	エ	strcpy(path, result)
オ	strcpy(result, base)	カ	strcpy(result, path)

cに関する解答群

ア bp - base　　イ bp - path

ウ pp - base　　エ pp - path

dに関する解答群

ア base + length　　イ bp + length

ウ path + length　　エ pp + length

オ result + length

C

設問2　表2の引数列で関数convertを呼んだときのプログラムの動作について，表2中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，プログラム中の［ a ］～［ d ］には正しい答えが入っているものとする。

表2　引数列

path	base	result
../../../../d/	/a/b/c/	e
d/	/a/b/c	f
d	/a/b/c/	g

eに関する解答群

ア ../d/　　イ ../../../../d/

ウ /　　エ /../d/

オ /d/　　カ d/

f，gに関する解答群

ア /a/b/c/d　　イ /a/b/c/d/

ウ /a/b/cd　　エ /a/b/cd/

オ d　　カ d/