過去問題にトライ

最後に、既出問題にトライしてみましょう。今まで学んできた要素は、単体ではそれほど難しいものではありませんが、組み合わさると難しく感じます。パーツに分けて考えてみたり、場合分けをして考えてみたり、具体例をはめて考えてみたり、そのときどきでアプローチの仕方は違いますが、より多くの問題に接することで、分解の仕方がわかってくると思います。その上で、各要素の詳細を理解していたり、把握していることが求められますので、学習したことを定着させるよい機会になると思います。

問題を小さく分割する力や、各要素に対する知識はプログラミングスキルの向上に通じるところがありますので、ただ資格試験の勉強をしているということではなく、現場での能力向上に寄与すると考えています。最初は時間がかかっても構いませんので、じっくりと問題に対峙していきましょう。

1 過去問題①

問題（2015 年度（平成 27 年度）　春　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。

〔プログラムの説明〕

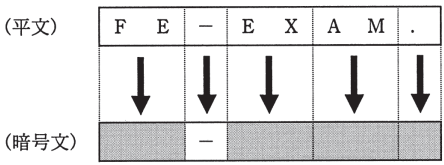
平文（ひらぶん）の文字列を暗号化する手法として，平文の文字を換字表（かえじ）に従って別の文字に置き換える方法がある。関数 **enc_str** は，与えられた平文を，換字表を用いて暗号文に変換するプログラムである。

(1) 平文は，JIS X 0201（7 ビット及び 8 ビットの情報交換用符号化文字集合）の文字で構成されている。

(2) 置換の対象となる文字（以下，置換対象文字という）は，英字（A ～ Z，a ～ z），数字（0 ～ 9），空白文字，“.” 及び “,” の 65 種類の文字であり，換字表に格納されている。

(3) 換字表は 5 行 13 列の 2 次元文字型配列であり，全ての要素に異なる文字が格納されている。

(4) 換字表による置換の方法について，図 1 に示す平文と暗号文の対応の例を用いて説明する。平文の文字列の先頭から置換対象文字を 2 文字ずつ選び出して行い，置換されたそれぞれの文字を暗号文の文字として，暗号文中で平文と同じ位置に入れる。ここで，置換対象文字の組合せによって，置換後の文字の組合せが決まる。置換対象文字数が奇数の場合，最後の置換対象文字については 1 文字で置換を行う。置換対象文字以外の文字については，平文中の文字をそのまま暗号文中で平文と同じ位置に入れる。



注記　網掛けの部分には置換された文字が入る。

図 1　平文と暗号文の対応の例

(5) 関数 enc_str の仕様は次のとおりである。ここで，引数の値に誤りはないものとする。

機能： 文字列 str を，換字表 xchg_t を用いて平文から暗号文に変換する。

引数： str 文字列

xchg_t 換字表

〔プログラム〕

```
#define RSIZ  5  /* 換字表の行数 */
#define CSIZ 13  /* 換字表の列数 */

void enc_str(char[], char[RSIZ][CSIZ]);

void enc_str(char str[], char xchg_t[RSIZ][CSIZ]) {
    int cp[2],  /* 置換対象文字の換字表中の列位置 */
        rp[2],  /* 置換対象文字の換字表中の行位置 */
        pos[2], /* 置換対象文字の文字列中の位置 */
        flg, i = 0, col, row, p = 0;

    while (str[p] != '\0') {
        flg = 0;
        for (row = 0; row < RSIZ; row++) {
            for (col = 0; col < CSIZ; col++) {
                if (str[p] == xchg_t[row][col]) {
                    flg = 1;
                    break;
                }
            }
            if (flg != 0) break;
        }
        if (flg != 0) {                                    ← α
            cp[i]    = col;
            rp[i]    = row;
            pos[i++] = p;
            if (i == 2) {
                if (str[pos[0]] == str[pos[1]]) {
                    str[pos[0]] = str[pos[1]] =
                                        xchg_t[(rp[0] + 1) % RSIZ]
                                              [(cp[0] + 1) % CSIZ];
                } else {
                    if (rp[0] == rp[1]) {
                        str[pos[0]] = xchg_t[rp[1]][cp[1]];
                        str[pos[1]] = xchg_t[rp[0]][cp[0]];
                    } else if (cp[0] == cp[1]) {
                        str[pos[0]] = xchg_t[rp[0]]
                                            [(cp[0] + 1) % CSIZ];
                        str[pos[1]] = xchg_t[rp[1]]
                                            [(cp[1] + 1) % CSIZ];
                    } else {
                        str[pos[0]] = xchg_t[rp[1]][cp[0]];
                        str[pos[1]] = xchg_t[rp[0]][cp[1]];
                    }
                }
                i = 0;                                     ← β
```

```
            }
        }
        p++;
    }
    if (i != 0) {
        str[pos[0]] = xchg_t[RSIZ - 1 - rp[0]][CSIZ - 1 - cp[0]];
    }
}
```

設問 1　図 2 の平文と換字表を引数として関数 enc_str を実行したとき，プログラムの動作に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

文字列［位置］

```
          0         10        20   25
          +---------+---------+----+
(平文)    FunctionΔΔf(x,y)ΔisΔgiven.
```

換字表[行位置][列位置]

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	a	b	c	d	e	f	g	h	i	j	k	l	m
1	n	o	p	q	r	s	t	u	v	w	x	y	z
2	0	1	2	3	4	5	6	7	8	9	,	.	Δ
3	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
4	M	L	K	J	I	H	G	F	E	D	C	B	A

注記　“Δ”は空白文字を表す。

図 2　平文と換字表の例

(1) プログラムの **α** の行が最初に実行されるとき，変数 flg の値は 1，i の値は 0，col の値は［　a　］，row の値は［　b　］になっている。

(2) プログラムの **β** の行が最初に実行されるとき，引数 str，配列 cp，rp，pos に格納されている値は，図 3 に示すとおりである。

(添字)	0	1	2	…
str		c	'n'	…

(添字)	0	1
cp	a	7
rp	b	1
pos	0	1

注記　網掛けの部分は表示していない。

図3　引数 str，配列 cp，rp，pos に格納されている値

(3) プログラムの**β**の行が5回目に実行されるとき，pos[1] の値は9，引数 str[9] の値は［ d ］になっている。

(4) プログラムの**β**の行が6回目に実行されるとき，pos[1] の値は［ e ］になっている。

a，b に関する解答群

ア　1　　イ　2　　ウ　3　　エ　4　　オ　5

カ　6　　キ　7　　ク　8　　ケ　9　　コ　10

c に関する解答群

ア　'F'　　イ　'S'　　ウ　't'

エ　'v'　　オ　'7'

d に関する解答群

ア　'n'　　イ　'0'　　ウ　'y'

エ　'Z'　　オ　空白文字

e に関する解答群

ア　11　　イ　12　　ウ　13　　エ　14　　オ　15

設問2　図2に示す換字表を使って平文“IPA”を暗号文に変換した結果として正しい答えを，解答群の中から選べ。

解答群

ア　CVa　　イ　iAP　　ウ　iCN

エ　iNC　　オ　PIa　　カ　VCa

解答

設問 1　(a) キ　(b) エ　(c) エ　(d) エ　(e) イ
設問 2　カ

解説

文字列を暗号化する問題です。プログラムの穴埋めが 1 つもなく、プログラムを正確に解釈できるかどうかが問われます。そのため、プログラムの説明文には、肝心なことはあまり記載されていませんが、いくつかヒントがあります。重要なヒントとしては、

・2 文字ずつ暗号化する
・換字表にない文字はそのまま出力される
・最後に 1 文字余った (文字数が奇数だった) ときは、1 文字で変換する

ということです。それでは、プログラムをじっくり読んでいきましょう。

図1　全体を読み解く

```
#define RSIZ  5  /* 換字表の行数 */
#define CSIZ 13  /* 換字表の列数 */

void enc_str(char[], char[RSIZ][CSIZ]);

void enc_str(char str[], char xchg_t[RSIZ][CSIZ]) {
    int cp[2],   /* 置換対象文字の換字表中の列位置 */
        rp[2],   /* 置換対象文字の換字表中の行位置 */
        pos[2], /* 置換対象文字の文字列中の位置 */
        flg, i = 0, col, row, p = 0;

    while (str[p] != '\0') {
        flg = 0;
        for (row = 0; row < RSIZ; row++) {
            for (col = 0; col < CSIZ; col++) {
                if (str[p] == xchg_t[row][col]) {
                    flg = 1;
                    break;
                }
            }
            if (flg != 0) break;
        }
        if (flg != 0) {
            cp[i]    = col;
            rp[i]    = row;
            pos[i++] = p;
            if (i == 2) {
                if (str[pos[0]] == str[pos[1]]) {
                    str[pos[0]] = str[pos[1]] =
                                        xchg_t[(rp[0] + 1) % RSIZ]
                                              [(cp[0] + 1) % CSIZ];
                } else {
                    if (rp[0] == rp[1]) {
                        str[pos[0]] = xchg_t[rp[1]][cp[1]];
                        str[pos[1]] = xchg_t[rp[0]][cp[0]];
                    } else if (cp[0] == cp[1]) {
                        str[pos[0]] = xchg_t[rp[0]]
                                            [(cp[0] + 1) % CSIZ];
                        str[pos[1]] = xchg_t[rp[1]]
                                            [(cp[1] + 1) % CSIZ];
                    } else {
                        str[pos[0]] = xchg_t[rp[1]][cp[0]];
                        str[pos[1]] = xchg_t[rp[0]][cp[1]];
                    }
                }
                i = 0;
            }
        }
        p++;
    }
    if (i != 0) {
        str[pos[0]] = xchg_t[RSIZ - 1 - rp[0]][CSIZ - 1 - cp[0]];
    }
}
```

①文字列への添字は p であることがわかる

⑤換字表から平文の文字を探している。見つかると row と col に添字が入り、flg = 1 となる。

α

この i++ を見落とさない様に！ pos[i] = p; のあとで i++ されている。

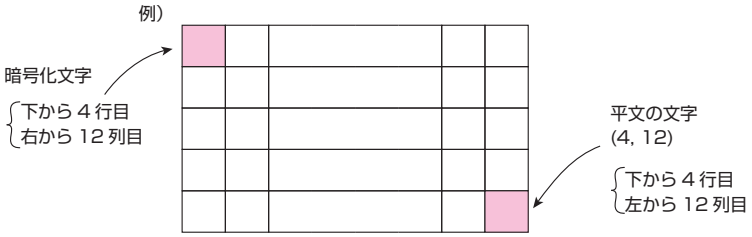
②1 文字ずつループ

③2 文字そろったときの処理

β

④ここで while 文は終了しているので、以下は最後に 1 文字残ったときの処理

下から数える　　右から数える

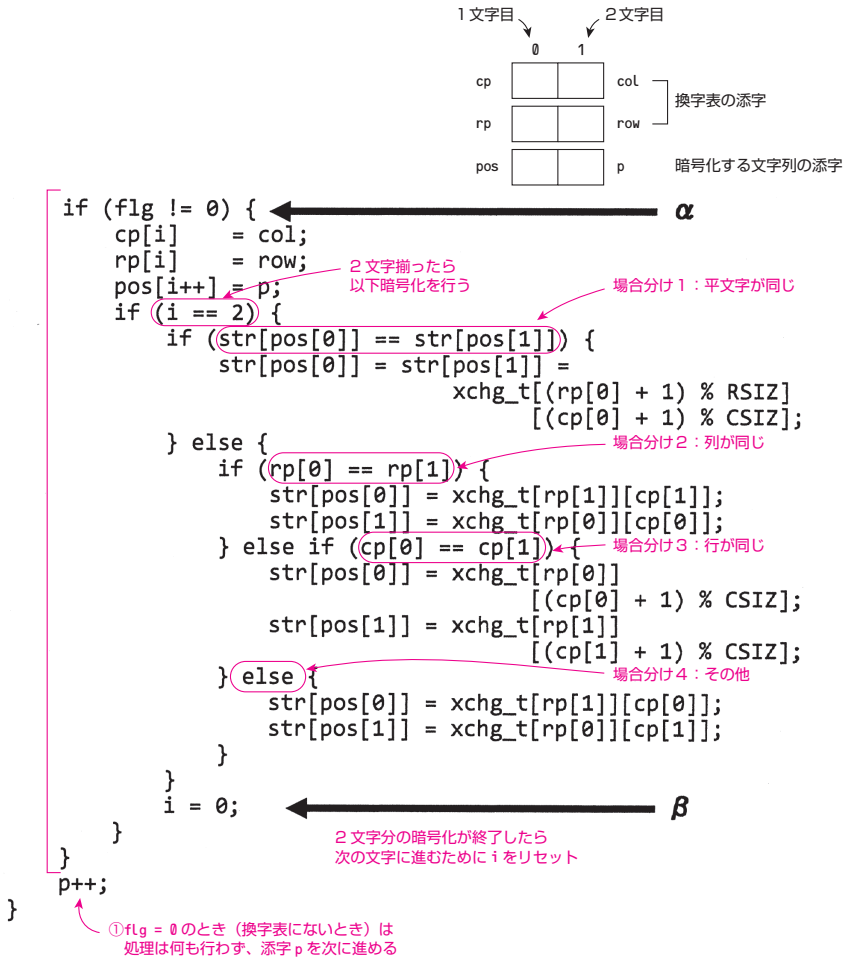


pを添字として、1文字ずつ処理が行われることがわかります（図1①②）。2文字そろっ

たところで、何等かの暗号化を行い（図１③）、while ループを終了した後で、最後に１文字残ったときの処理を行っていることがわかります（図１④）。while 文の上方では、平文字が換字表のどの位置にあるのかを調べ、row と col に求めています（図１⑤）。

続いて、核の部分を読んでいきます。

図2　暗号化部分

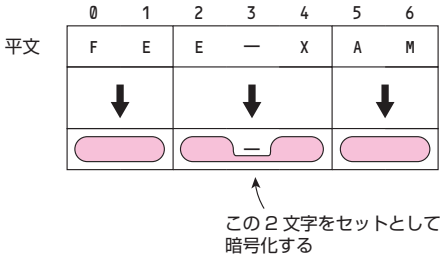



```
if (flg != 0) {
    cp[i]    = col;
    rp[i]    = row;
    pos[i++] = p;
    if (i == 2) {
        if (str[pos[0]] == str[pos[1]]) {
            str[pos[0]] = str[pos[1]] =
                                xchg_t[(rp[0] + 1) % RSIZ]
                                      [(cp[0] + 1) % CSIZ];
        } else {
            if (rp[0] == rp[1]) {
                str[pos[0]] = xchg_t[rp[1]][cp[1]];
                str[pos[1]] = xchg_t[rp[0]][cp[0]];
            } else if (cp[0] == cp[1]) {
                str[pos[0]] = xchg_t[rp[0]]
                                      [(cp[0] + 1) % CSIZ];
                str[pos[1]] = xchg_t[rp[1]]
                                      [(cp[1] + 1) % CSIZ];
            } else {
                str[pos[0]] = xchg_t[rp[1]][cp[0]];
                str[pos[1]] = xchg_t[rp[0]][cp[1]];
            }
        }
        i = 0;
    }
}
p++;
}
```

flg=0 のとき、つまり、平文字が換字表にみつからなかったときは、何もせず、次に進みます（図 5.2 ①）。問題文図１の例では、３文字目の「-」にあたります。この例では２文字と２文字の間に「-」がありましたが、２文字の途中であっても、同様に無視

されます。

図3　換字表にない文字が 2 文字の途中にある例



そしていよいよ暗号化処理に入ります。暗号化は、それまでの状態によって、4 つに場合分けされています。

[設問 1]

プログラムの α が最初に実行されるときは、対象となる文字は先頭の 2 文字ですから「F」と「u」です。換字表で調べると、図 4 が得られます。cp が同じですから、場合分け 3 に該当し、'F''u' ともに、変換表の 1 つ右の文字が採用されます。'F' は 'E' に、'u' は 'v' に暗号化されます (a)(b)(c)。ここで、CSIZ により剰余を求めていますが、これは、表の右端の文字の場合、添字に 1 を加算すると、表をはみ出してしまうので、左の端とするための処理です。場合分け 1 において、列に対しても同様です (図 5)。

図4　α が最初に実行されるとき

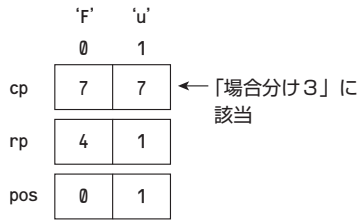
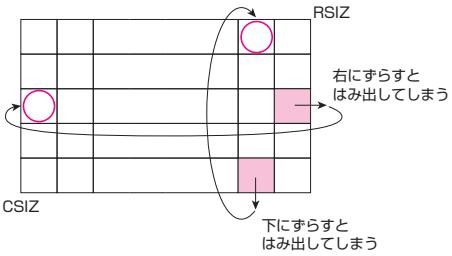
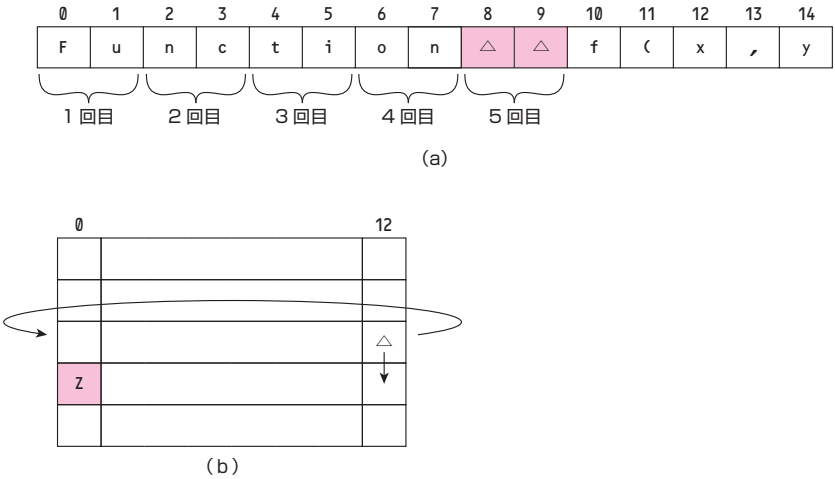


図5　CSIZ・RSIZ による剰余



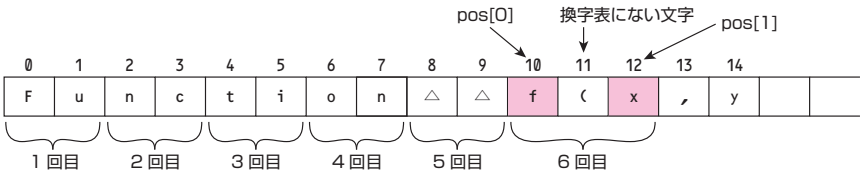
β は、2 文字ごとの変換の 5 回目であり、図 6(a) の 2 文字が該当します。該当文字は同じ文字なので、場合分け 1 の処理が行われます。このときは、2 文字とも換字表の右斜め下の文字に暗号化されます (図 6(b))。

図6　設問1の文字列5回目



βが6回目に実行されるときは、図7の文字が対象になります。

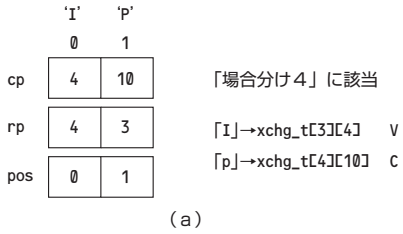
図7　設問1の文字列6回目



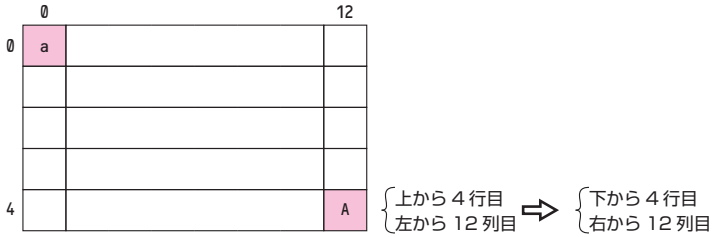
[設問2]

「I」「P」については、cp,rpの値は図8(a)のようになり、場合分け4のケースになります。最後に残った「A」は、図8(b)のとおりに暗号化されます。

図8　「I」「p」に対する換字表の対応



（a）



（b）

2 過去問題②

問題（2014 年度（平成 26 年度）秋　問 9）

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。

B 社では，セキュリティ管理のために，システムファイルから定期的に，システムの運用・保守用の利用者 ID 一覧を作成し，ファイルで保管している。

これらの利用者 ID に付加できる特権には，システムファイルの更新などができるシステム特権（以下，特権 S という）及びバックアップ作業などのために全てのファイルを参照できるオペレーション特権（以下，特権 O という）の 2 種類がある。

B 社では，セキュリティ管理強化の一環で，利用者 ID の追加・削除や特権の付加・解除が，申請に基づいて正しくシステムに反映されているかどうかを検証することになった。そのために，最新及び 1 世代前の利用者 ID 一覧を比較して，この間の利用者 ID 及び特権の登録内容の差異を印字するプログラムを作成する。

〔プログラムの説明〕

（1）最新の利用者 ID 一覧をファイル `NewFile` から，1 世代前の利用者 ID 一覧をファイル `OldFile` から，それぞれ読み込む。レコードは利用者 ID の昇順に整列されている。

（2）1 レコードは，利用者 ID（1 ～ 8 桁），利用者名（1 ～ 10 桁），属性（1 桁）及び最終使用日（8 桁）の 4 項目から成る。各項目は，空白文字で区切られている。

① 利用者 ID 及び利用者名は，英数字から成る文字列である。

② 属性は，次に示す内容の 8 ビット長のビット列 [0][1][0][0][s][o][g][r] である。

上位 4 ビット：	固定値 0100
ビット s：	特権 S が付加されていれば 1，付加されていなければ 0
ビット o：	特権 O が付加されていれば 1，付加されていなければ 0
ビット g 及び r：	その他の属性

③ 最終使用日は，数字から成る文字列で，その利用者 ID で最後にログインした年月日を表す。登録後，一度もログインしていない場合は，全桁が “0” である。

（3）利用者 ID 及び特権の登録内容の差異は，次のように印字する。

① NewFile 中にあって OldFile 中にない利用者 ID の場合
利用者 ID，利用者名の後に“利用者 ID 追加”と印字する。この利用者 ID に特権 S が付加されていれば“特権 S 付加”，特権 O が付加されていれば“特権 O 付加”を追加印字する。

② OldFile 中にあって NewFile 中にない利用者 ID の場合
利用者 ID，利用者名の後に“利用者 ID 削除”と印字する。この利用者 ID に特権 S が付加されていれば“特権 S 解除”，特権 O が付加されていれば“特権 O 解除”を追加印字する。

③ NewFile 及び OldFile の両方にあり，特権 S と特権 O の少なくとも一方の付加状況が変わった利用者 ID の場合
利用者 ID，利用者名の後に“特権 S 付加”，“特権 S 解除”，“特権 O 付加”，“特権 O 解除”の該当する全てを印字する。

(4) 入力ファイル NewFile 及び OldFile のデータ例を図 1 に，図 1 のデータ例を用いた実行結果を図 2 に，それぞれ示す。

ファイル NewFile のデータ例

利用者 ID	利用者名	属性	最終使用日
AE001	UserE1	41	20140419
AE002	UserE2	48	20141014
AE003	UserE3	48	20140716
AP005	UserP5	46	20141015
AP006	UserP6	44	00000000

ファイル OldFile のデータ例

利用者 ID	利用者名	属性	最終使用日
AE001	UserE1	40	20140419
AE002	UserE2	44	20140712
AE003	UserE3	4C	20140716
AP004	UserP4	45	20140715
AP005	UserP5	44	20140713

注記 1　属性の値（網掛け部分）は，16 進数で表示している。
注記 2　見出し行は，各ファイルには含まれないものとする。

図 1　入力ファイル NewFile 及び OldFile のデータ例

利用者 ID	利用者名	登録内容の差異
AE002	UserE2	特権 S 付加　特権 O 解除
AE003	UserE3	特権 O 解除
AP004	UserP4	利用者 ID 削除　特権 O 解除
AP006	UserP6	利用者 ID 追加　特権 O 付加

注記　見出し行は，事前に印字されているものとする。

図 2　図 1 のデータ例を用いた実行結果

(5) ライブラリ関数 strcmp(s1, s2) は，文字列 s1 と s2 を比較し，s1 < s2 のとき負の値を，s1 = s2 のとき 0 を，s1 > s2 のとき正の値を，それぞれ返す。

〔プログラム〕

```
#include <stdio.h>
#include <string.h>

#define  BitS  0x08
#define  BitO  0x04
#define  BitR  0x01

FILE         *NewFile,   *OldFile;
int           NewEof,     OldEof;
char          NewID[9],   OldID[9];
char          NewName[11],OldName[11];
unsigned char NewAttr,    OldAttr;
char          NewDate[9], OldDate[9];

void ReadNewRecord() {

   fscanf(NewFile, "%s %s %c %s", NewID, NewName, &NewAttr, NewDate);
   if (feof(NewFile) != 0) {
      NewEof = EOF;                 /* EOF:  負の整数定数 */
      strcpy(NewID, "\xFF");        /* \xFF: 8 ビット符号の最大値 */
   }
}

void ReadOldRecord() {

   fscanf(OldFile, "%s %s %c %s", OldID, OldName, &OldAttr, OldDate);
   if (feof(OldFile) != 0) {
      OldEof = EOF;
      strcpy(OldID, "\xFF");
   }
}

void main() {

   NewFile = fopen("NewFile", "rb");
   OldFile = fopen("OldFile", "rb");
   NewEof = 0;
   OldEof = 0;
   ReadNewRecord();
   ReadOldRecord();
```

```
   while (      a      ) {
      if (strcmp(NewID, OldID) == 0) {
         if (      b      ) {
            printf("\n%-8s  %-10s", NewID, NewName);
            if ((NewAttr & BitS) > (OldAttr & BitS))
               printf("  特権 S 付加");
            if ((NewAttr & BitS) < (OldAttr & BitS))
               printf("  特権 S 解除");
            if ((NewAttr & BitO) > (OldAttr & BitO))
               printf("  特権 O 付加");
            if ((NewAttr & BitO) < (OldAttr & BitO))
               printf("  特権 O 解除");
         }
         c
α     }
      else {
         if (      d      ) {
            printf("\n%-8s  %-10s  利用者 ID 追加", NewID, NewName);
            if ((NewAttr & BitS) == BitS)  printf("  特権 S 付加");
            if ((NewAttr & BitO) == BitO)  printf("  特権 O 付加");
            ReadNewRecord();
         }
         else {
            printf("\n%-8s  %-10s  利用者 ID 削除", OldID, OldName);
            if ((OldAttr & BitS) == BitS)  printf("  特権 S 解除");
            if ((OldAttr & BitO) == BitO)  printf("  特権 O 解除");
            ReadOldRecord();
         }
      }
   }
   fclose(OldFile);
   fclose(NewFile);
}
```

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 に関する解答群

ア　(NewEof != EOF) && (OldEof != EOF)

イ　(NewEof != EOF) || (OldEof != EOF)

ウ　NewEof != OldEof

エ　NewEof == OldEof

b に関する解答群

ア　`((NewAttr & OldAttr) & (BitS + Bit0)) != 0x00`

イ　`((NewAttr | OldAttr) & (BitS + Bit0)) != 0x00`

ウ　`(NewAttr & (BitS + Bit0)) != (OldAttr & (BitS + Bit0))`

エ　`(NewAttr | (BitS + Bit0)) != (OldAttr | (BitS + Bit0))`

c に関する解答群

ア
```
ReadNewRecord();
```

イ
```
ReadNewRecord();
ReadOldRecord();
```

ウ
```
ReadOldRecord();
```

d に関する解答群

ア　`(NewAttr & (BitS + Bit0)) != 0x00`　　イ　`NewAttr > OldAttr`

ウ　`strcmp(NewID, OldID) < 0`　　エ　`strcmp(NewID, OldID) > 0`

設問 2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

利用者 ID が有効に使用されているかどうかを検証するために，一定期間未使用，又は現在使用不可の利用者 ID 一覧を印字するプログラムを作成する。

(1) 最新の利用者 ID 一覧をファイル **NewFile** から，一定期間前の世代の利用者 ID 一覧をファイル **OldFile** から，それぞれ読み込む。

(2) 次の①，②の少なくとも一方に該当する利用者 ID について，利用者 ID，利用者名の後に，①に該当する場合は“現在使用不可”を，②に該当する場合は“期間中未使用”を印字する。

① **NewFile** 中にあって，属性のビット r が 1 である利用者 ID
ここで，属性のビット r は，利用者 ID が使用可能なら 0，使用不可なら 1 である。

② **NewFile** 及び **OldFile** の両方にあって，最終使用日の値が等しい利用者 ID（全桁が“0”同士で等しい場合を含む）

(3) 図 3 に，図 1 のデータ例を用いた実行結果を示す。

```
利用者 ID   利用者名      使用状況
AE001       UserE1       現在使用不可   期間中未使用
AE003       UserE3       期間中未使用
```

注記　見出し行は，事前に印字されているものとする。

図３　図１のデータ例を用いた使用状況の印字結果

この処理を実装するためには，プログラム中の **while** 文のブロック内（*α* で示した部分）を次のように変更すればよい。ここで，プログラム中の［　a　］には，正しい答えが入っているものとする。

```
if (strcmp(NewID, OldID) <= 0) {
   if ((      e      ) || (      f      )) {
      printf("\n%-8s  %-10s", NewID, NewName);
      if (      e      )
         printf("  現在使用不可");
      if (      f      )
         printf("  期間中未使用");
   }
   ReadNewRecord();
}
else
   ReadOldRecord();
```

e，f に関する解答群

ア　(NewAttr & BitR) != (OldAttr & BitR)
イ　(NewAttr & BitR) == BitR
ウ　strcmp(NewDate, OldDate) == 0
エ　strcmp(NewID, OldID) == 0
オ　strcmp(NewID, OldID) == 0 && (NewAttr & BitR) == BitR
カ　strcmp(NewID, OldID) == 0 && strcmp(NewDate, OldDate) == 0

解答

設問１　(a)イ　(b)ウ　(c)イ　(d)ウ
設問２　(e)イ　(f)カ

解説

セキュリティを話題にはしていますが､セキュリティの知識はまったく不要です。ビット処理と文字列処理とファイル処理が組み合わされていますが、その分アルゴリズムは比較的平易です。問題文とプログラムとを読み合わせていくことで理解を深めましょう。

1）関数 ReadNewRecord() の仕様

最新ファイルから 1 レコードを読み込む。

変数名	型	内容
NewID	char[]	最新ファイルの利用者 ID
NewName	char[]	最新ファイルの利用者名
NewAttr	unsigned char	最新ファイルの属性
NewDate	char[]	最新ファイルの最終使用日

ファイルの終端では、newID を 0xFF にする。

2）関数 ReadOldRecord() の仕様

1 世代前のファイルから 1 レコードを読み込む。

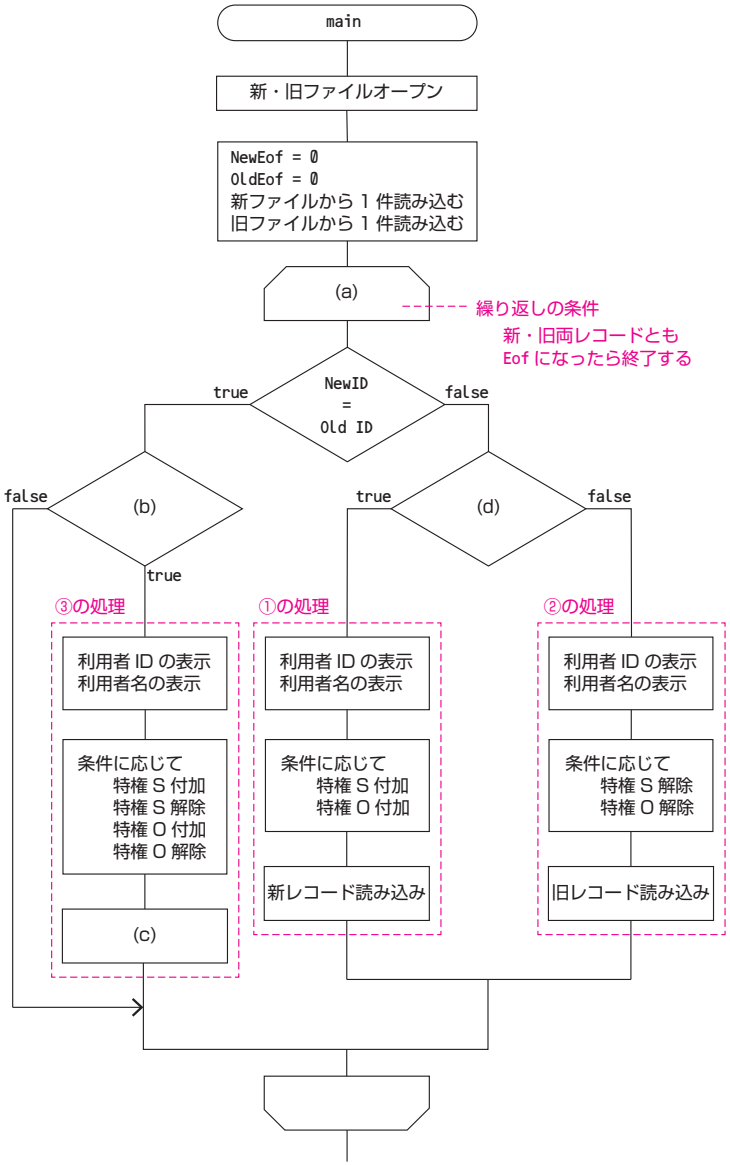
変数名	型	内容
OldID	char[]	1 世代前の利用者 ID
OldName	char[]	1 世代前の利用者名
OldAttr	unsigned char	1 世代前の属性
OldDate	char[]	1 世代前の最終使用日

ファイルの終端では、oldID を 0xFF にする。

3）main() の処理

流れ図を図 1 に示します。

図1　処理の流れ図



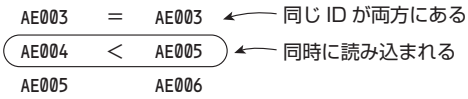
[設問 1]

(a) は繰り返し条件です。新旧両ファイルが EOF になったら終了です。ここには、継続条件を書きますので、新ファイルが EOF でない、または旧ファイルが EOF でない間繰り返します。

(b) は問題文中のプログラムの説明 (3) ③に該当します。新・旧の属性のビット s またはビット o が異なるときに変化を表示します。属性そのものを比較するのではなく、該当する 2 ビットでマスクをしてから比較します。ビットのマスクをするにはビットごとの AND (演算子は &) をとります。その結果が不一致のとき、処理が発生します。このケースでは、新・旧両レコードの処理が終了しますので、最後に新・旧両方のレコードを読み込みます。

(d) により、問題文中のプログラムの説明 (3) の①と②が分かれます。(d) が true のとき①の処理、つまり、新レコードにあって旧レコードにない場合となります。新レコードにあって旧レコードにないとは、図 2 のような状態です。新・旧ファイルは昇順に並んでいますから、新<旧のとき、新ファイルにだけレコードがある状態といえます。

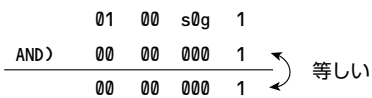
図2　新ファイルにあって旧ファイルにない



[設問 2]

条件 (e) のとき、「現在使用不可」と表示しますから、(e) には、設問 2 問題文中 (2) ①の内容が入ります。新レコードの属性ビット r が 1 のとき使用不可ですから、NewAttr にビット r でマスクをとります。その結果、ビット r の位置だけが 1 になれば、使用不可となります (図 3)。

図3　NewAttr にマスクをとる



条件 (f) のとき、「期間中未使用」と表示しますから、(f) には、設問 2 問題文中 (2) ②の内容が入ります。新・旧レコードの最終使用日が等しいとき、とありますので、新・旧 2 つの利用者 ID と最終使用日を文字列として比較し、両方等しいとき、期間中未使用となります。

3 過去問題③

問題（2018 年度（平成 30 年度）　春　問 9）

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。

簡易集計プログラムである。このプログラムを用いると，例えば，図 1 の入力ファイルから，品番ごとのレコード件数と金額の合計を求めて，図 2 の集計ファイルを得ることができる。

プログラムは，キー項目及び数値項目の，開始桁位置及び桁数を引数で受け取り，キー項目で整列済みのレコードを入力ファイルから読み込み，キー項目の値ごとに，その件数と数値項目の値の合計を求め，集計ファイルにレコードとして書き出す。ここで，数値項目には整数の値が入る。

年月日 時分秒 品番 数量 金額

```
20180410,112610,FE-111,0001,000100
20180410,154358,FE-111,0001,000100
20180410,123820,FE-222,0002,000400
20180410,153249,FE-333,0001,000300
20180410,135044,FE-333,0001,000300
20180410,152859,FE-333,0001,000300
20180410,131923,FE-444,0002,000800
20180410,123907,FE-444,0001,000400
```

注記　見出しの行はレコードに含まれない。
改行文字は表示していない。
破線は項目の区切りを表す。

図 1　入力ファイルのレコード例

品番	件数	合計金額
FE-111	2	200
FE-222	1	400
FE-333	3	900
FE-444	2	1200

注記　見出しの行はレコードに含まれない。
改行文字は表示していない。

図 2　集計ファイルのレコード例

〔プログラム 1 の説明〕

(1) 入力ファイルは，固定長レコードの並びから成る。レコードは，1,000 文字以下の 1 バイト文字の並びであり，最後の文字の後には改行文字が付いている。ファイル名は，引数 dataFile で指定する。

(2) レコード中のキー項目の開始桁位置及び桁数は，それぞれ引数 keyPos 及び keyLen で指定する。また，数値項目の開始桁位置及び桁数は，それぞれ引数 valuePos 及び valueLen で指定する。開始桁位置は，レコードの先頭文字の桁位

置を 0 として数える。キー項目及び数値項目の桁数は，いずれも 9 桁以下とする。

(3) 入力レコードは，キー項目の昇順に整列されている。

(4) 集計ファイルにレコードとして，キー項目の値，キー項目ごとの件数及び数値項目の値の合計を各 9 桁分の領域に右詰めで出力し，各項目の直前に 1 個の空白文字を出力する。レコードの最後の文字の後には改行文字を付ける。ファイル名は，引数 **listFile** で指定する。

(5) 引数及び入力レコードの内容に誤りはないものとする。また，数値項目の値の合計は 9 桁以下であり，算術演算であふれは起きないものとする。

(6) プログラム中で使用しているライブラリ関数(一部)の概要は，次のとおりである。

- **atol(*s*)**：　文字列 *s* が表す数値を **long** 型の表現に変換した値を返す。
- **fgets(*s*, *m*, *f*)**：　ストリーム *f* から文字の列(改行文字まで,最大 *m* − 1 文字)を読み取り，配列 *s* に格納し，*s* を返す。ストリームの終わりに達した場合は **NULL** を返す。
- **strcmp(*s1*, *s2*)**：　文字列 *s1* と *s2* を比較し，*s1* < *s2* のとき負の値を，*s1* = *s2* のとき 0 を，*s1* > *s2* のとき正の値を，それぞれ返す。
- **strcpy(*s1*, *s2*)**：　文字列 *s2* を文字列 *s1* に複写する。
- **strncpy(*s1*, *s2*, *n*)**：文字列 *s2* の先頭から *n* 個の文字を文字列 *s1* に複写し，文字列 *s1* の値を返す。

[プログラム 1]

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define  recSize  1002   /* 入力レコードの最大文字数 + 2 ('\n', '\0') */

void outList(char *dataFile, char *listFile,
             int keyPos, int keyLen, int valuePos, int valueLen) {
    [ a ] *inFile, *outFile;
    char inBuf[recSize], inKey[10], key[10], temp[10];
    long count, inValue, value;
    char format[] = " %9s %9ld %9ld\n";
```

```
    key[keyLen] = '\0';
    inKey[keyLen] = '\0';
    temp[valueLen] = '\0';
    outFile = fopen(listFile, "w");
    inFile = fopen(dataFile, "r");


    if (fgets(inBuf, recSize, inFile) != NULL) {
       strncpy(key, inBuf + keyPos, keyLen);
       count = 1;
       value = atol(strncpy(temp, inBuf + valuePos, valueLen));
       while (fgets(inBuf, recSize, inFile) != NULL) {
          strncpy(inKey, inBuf + keyPos, keyLen);
          inValue = atol(strncpy(temp, inBuf + valuePos, valueLen));
          if (strcmp(key, inKey) != 0) {
             fprintf(outFile, format, key, count, value);
             count = 1;
             strcpy(key, inKey);
             value = inValue;
          }
          else {
             [ b ]
          }
       }
       [ c ]
    }
    fclose(inFile);
    fclose(outFile);
}
```

設問 1　プログラム 1 中の［　　］に入れる適切な答えを，解答群の中から選べ。

a に関する解答群

ア　char　　イ　FILE　　ウ　file　　エ　int

b，c に関する解答群

ア　fprintf(outFile, format, inKey, count, inValue);

イ　fprintf(outFile, format, key, count, value);

ウ　count++;
　　value += inValue;

エ　count++;

```
    value += inValue;
    fprintf(outFile, format, key, count, value);
オ  count--;
    fprintf(outFile, format, inKey, count, inValue);
```

設問 2　次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。ここで，プログラム 2 中の［ a ］には，設問 1 の正しい答えが入っているものとする。

図 3 に示す，時分秒のうちの時をキー項目として昇順に整列済みのレコードを入力ファイルから読み込み，時間帯（0 時台，1 時台，…，23 時台）ごとの件数と金額の合計（合計金額）を求め，時，件数，合計金額と合計金額を表す棒グラフを印字する。この処理を，次の手順で行う。

(1) プログラム 1 を利用して，図 3 の入力ファイルから，図 4 の集計ファイルを得る。
(2) プログラム 2 を利用して，図 4 の集計ファイルから，図 5 の印字結果を得る。

```
年月日   時 分秒   品番    数量   金額
20180410,11,2610,FE-111,0001,000100
20180410,12,3820,FE-222,0002,000400
20180410,12,3907,FE-444,0001,000400
20180410,13,1923,FE-444,0002,000800
20180410,13,5044,FE-333,0001,000300
20180410,15,2859,FE-333,0001,000300
20180410,15,3249,FE-333,0001,000300
20180410,15,4358,FE-111,0001,000100
```

注記　見出しの行はレコードに含まれない。
　　改行文字は表示していない。
　　破線は項目の区切りを表す。

図 3　入力ファイルのレコード例

```
時      件数   合計金額
11         1        100
12         2        800
13         2       1100
15         3        700
```

注記　見出しの行はレコードに含まれない。
　　改行文字は表示していない。

図 4　集計ファイルのレコード例

```
時      件数   合計金額
11         1        100  |**
12         2        800  |******************
13         2       1100  |*************************
15         3        700  |***************
```

注記　見出しの行は印字結果に含まれない。

図 5　印字結果の例

〔プログラム 2 の説明〕

(1) プログラム 1 で書き出した集計ファイルからレコードを読み込む。ファイル名は，

引数 listFile で指定する。

(2)読み込んだ各レコードに，そのレコードの合計金額の値 value に応じた長さのグラフを追加して印字する。集計ファイル中の合計金額の値の最大値 valueMax を 25 個の“*”で表し，他の値は，25 × value ÷ valueMax 個（小数点以下切捨て）の“*”で表す。ここで，集計ファイル中の合計金額の値の最大値は正であり，最小値は 0 以上であるものとする。

〔プログラム 2〕

```
#include <stdio.h>
#include <stdlib.h>

#define  listLen  32     /* 入力レコードの最大文字数 + 2 ('\n', '\0') */

void outListG(char *listFile) {
    [    a    ] *inFile;
    char inBuf[listLen];
    long value, valueMax;
    char graph[] = "*************************";

    inFile = fopen(listFile, "r");
    valueMax = 0;
    while (fgets(inBuf, listLen, inFile) != NULL) {
        value = atol(inBuf + 21);
        if (value > valueMax) {
            valueMax = value;
        }
    }
    fclose(inFile);
    inFile = fopen(listFile, "r");
    while (fgets(inBuf, listLen, inFile) != NULL) {
        value = atol(inBuf + 21);
        printf("%.30s  |%s\n",
                inBuf, &graph[25 - 25 * value / valueMax]);  /* α */
        /* %.30s はレコードの先頭からの 30 桁（\n の直前まで）を出力する */
    }
    fclose(inFile);
}
```

プログラム 2 は，作成途中である。図 4 の集計ファイルを用いると図 5 の印字結果が得られるが，集計ファイル中の合計金額の値によっては，コメント /* α */ を付した印字処理の実行時に問題が発生する場合がある。

〔プログラム 2 の説明〕の (2) にある前提 “ 集計ファイル中の合計金額の値の最大値は正であり，最小値は 0 以上である ” が満たされない場合も考慮に含め，コメント **/* α */** を付した印字処理の実行時に発生し得る事象として，①算術演算であふれが発生，②算術演算でゼロ除算が発生，③配列の定義外の要素位置を参照，がある。これらの事象が発生し得る **value** と **valueMax** の値の例を，表 1 にまとめた。ここで，**long** 型の数値の範囲は，$-2^{31} \sim 2^{31} - 1 (2^{31} = 2{,}147{,}483{,}648)$ とする。

表 1　事象 ① ～ ③ が発生し得る value と valueMax の値の例

発生し得る事象	value と valueMax の値の例
① 算術演算であふれ	d
② 算術演算でゼロ除算	e
③ 配列の定義外の要素位置を参照	f

d ～ f に関する解答群

	value の値	valueMax の値
ア	-1,000,000	0
イ	-10,000,000	10,000,000
ウ	0	10,000,000
エ	100	10,000,000
オ	10,000,000	10,000,000
カ	100,000,000	100,000,000

解答

設問 1　(a) イ　(b) ウ　(c) イ
設問 2　(d) カ　(e) ア　(f) イ

解説

ファイルから読み込んだ情報を集計する問題です。レコードの開始桁数の指定によって、何をキーにするか、品番なのか年月日なのかを切り替え、また、数量を集計したり金額を集計したりできるようになっています。

[設問 1]

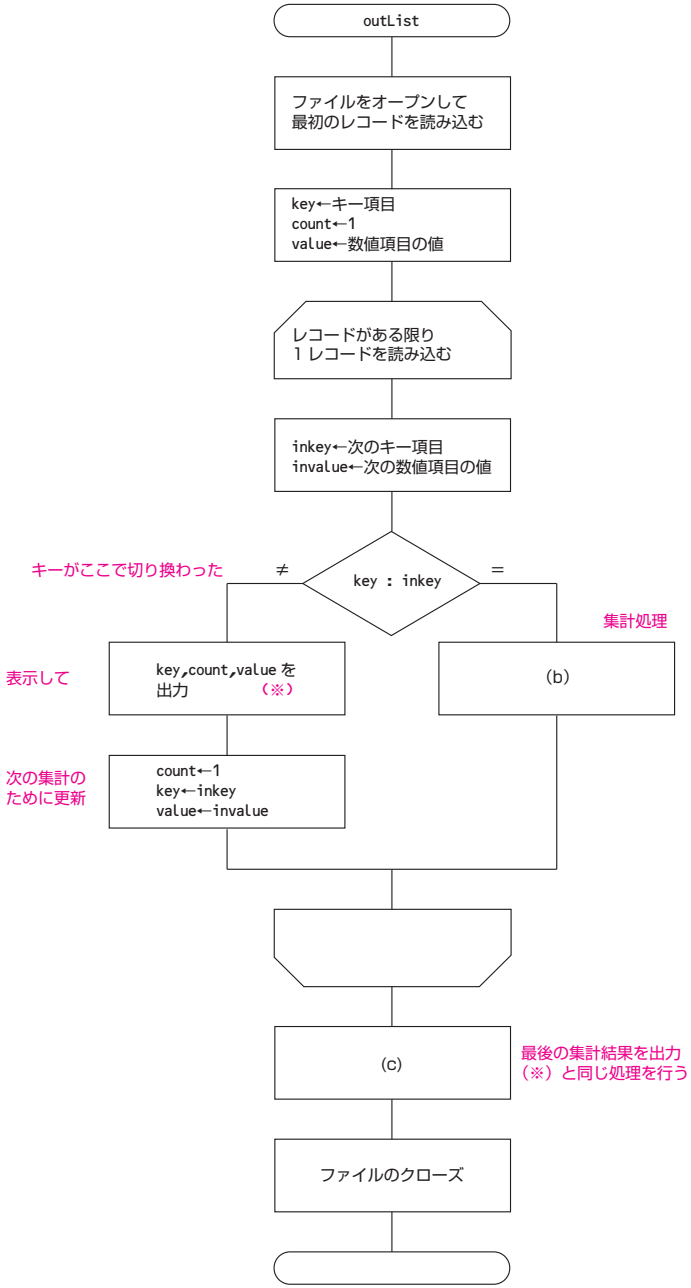
outFile・inFile は、プログラム中に

```
outFile = fopen(listFile , “w”);
inFile = fopen(datafile , “r”);
```

とありますから、FILE * 型であることがわかります。これが (a) の解答になります。

(b)(c) については、流れ図を描いてみましょう。(b) は前のレコードとキーコードが一致している場合、つまり、まだ集計の途中である場合です。カウンタ count を 1 進めて、inValue の値を value に累積していきます。(c) は、while 文後の処理です。最後に集計していた分の出力を行います。同様の処理が流れ図中の※にありますから、同じものを選択肢から選べばよいでしょう。

図1　処理の流れ図


outList
ファイルをオープンして
最初のレコードを読み込む
key←キー項目
count←1
value←数値項目の値
レコードがある限り
1 レコードを読み込む
inkey←次のキー項目
invalue←次の数値項目の値
キーがここで切り換わった
≠
key : inkey
=
集計処理
表示して
key,count,value を
出力　（※）
(b)
次の集計の
ために更新
count←1
key←inkey
value←invalue
(c)
最後の集計結果を出力
（※）と同じ処理を行う
ファイルのクローズ

[設問 2]

プログラムの前半で、数量の最大値を valueMax に求めています。それから、改めてファイルを開き、次式で「*」の数を求めています。

$$\frac{(25 \times value)}{valueMax}$$

value と valueMax の組み合わせによって、式の結果に不具合が出る例を問う問題です。

算術演算のあふれとは、演算の結果が格納できる範囲を超えてしまうことです。ここで問題なのは、先に 25 × value を計算することです。その段階で、long 型の数値の範囲を超えてしまうと発生します。valueMax の値に関わらず、25 × value の値が 2,147,483,648 を超える選択肢を選びます。

算術演算でゼロ除算とは、ゼロで除算を行うことです。value の値に関わらず、valueMax が 0 である選択肢を選びます。

配列の定義外の要素位置を参照とは、添字が配列要素を超えている、またはマイナスになっていることです。ここで、「*」の出力について少し触れておきましょう。配列 graph は「*」が 25 個並んだ文字列です。文字列なので、最後に「'\0'」が付加されており、26 個の要素を持ちます（図 2）。「*」の数に応じて、図 2 のように出力開始位置を決めています。ここでは、value の最大値は valueMax ですから、式の値が 25 を超えることはなく、value の値が正であれば、添字は 25 ～ 0 の範囲になります。しかし、value が負のときは、式の値が負になって、（25- 式の値）は 25 を超えてしまいます。以上により、maxValue が正で value が負のときに、この問題は発生します。

図 2　配列 graph の使い方