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，選択した問題については，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問以上マークした場合には，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。

入力ファイルを読み込んで，文字コードごとの出現回数を印字するプログラムである。

〔プログラムの説明〕

(1) 入力ファイルは，バイナリファイルとして読み込む。入力ファイル中の各バイトの内容（ビット構成）に制約はない。入力ファイル名は，#define で指定する。

(2) 入力ファイル中の各バイトについて，文字コード（16進数 00～FF で表示する）ごとの出現回数を求めて印字する。印字例を，図1に示す。

(3) 印字様式を次に示す（記号❶，❷，❸は，図1中の記号を指している）。

- 1行目に，処理したバイト数を❶の形式で印字する。
- 3行目以降に，出現回数とその文字コードを❷の形式で印字する。ただし，文字コードが20～7Eの場合は，文字コードの後にそれが表す文字（文字は，この冊子の末尾にあるアセンブラ言語の仕様の 1.3 で規定するもの）を❸の形式で印字する。文字コードは，64行×4列の範囲に，上から下，左から右に文字コードの昇順となるように並べる。

(4) プログラム中で使用している関数 fgetc(*s*) は，ストリーム *s* から1文字を読み込んで返す。ストリームが入力ファイルの終わりに達しているときは EOF を返す。

(5) 入力ファイルのサイズは，long 型（32ビットとする）で表現できる数値の範囲を超えないものとする。

C

```
❶
10000 bytes processed

         0 00              0 40 '@'        0 80          0 C0
         0 01 ❷          15 41 'A' ❸      0 81          0 C1
         0 02              0 42 'B'        0 82          0 C2
         0 03             20 43 'C'        0 83          0 C3
          :                  :              :             :      (中略)
      3620 20 ' '          0 60 '`'        0 A0          0 E0
         8 21 '!'        103 61 'a'        0 A1          0 E1
        76 22 '"'         10 62 'b'        0 A2          0 E2
          :                  :              :             :      (中略)
        18 3E '>'          0 7E '~'        0 BE          0 FE
         0 3F '?'          0 7F            0 BF          0 FF
```

図 1　印字例（文字コード順）

〔プログラム〕

```
#include <stdio.h>

#define  InName  "sample.c"    /* 入力ファイル名 */

int main() {

   FILE  *infile;
①→ int   chr, i;
   long  cnt;
   long  freq[256];  /* freq[i]: 文字コード i の出現回数 */

   for (chr = 0; chr <= 255; chr++)
      freq[chr] = 0;

   infile = fopen(InName, "rb");
   cnt = 0;
   while ((chr = fgetc(infile)) != EOF) {
      cnt++;
      freq[chr]++;
   }
   fclose(infile);
```

```
        printf(" %10ld bytes processed\n\n", [ a ]);
        for (i = 0; i < 64; i++) {
            for (chr = i; chr <= [ b ]; chr += [ c ]) {
                if ((0x20 <= chr) && (chr <= 0x7E))
                    printf(" %10ld %02X '%c'", freq[chr], chr, chr);
                else
                    printf(" %10ld %02X    ", freq[chr], chr);
            }
            printf("\n");
        }
②→
    }
```

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　cnt - 1　　イ　cnt　　ウ　cnt + 1

b，cに関する解答群

ア　4　　イ　64　　ウ　256

エ　i + 4　　オ　i + 64　　カ　i + 192

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。ここで，記述中の［ b ］と［ c ］には，設問1の正しい答えが入っているものとする。

文字コードを出現回数の降順に並べて印字する処理を追加する。追加した処理による印字例を，図2に示す。印字の様式は，文字コードの並び順を除いて，図1の3行目以降の様式と同じである。同じ出現回数の文字コードは，それらを文字コードの昇順に並べる。

```
3620 20 ' '        12 4B 'K'        0 80        0 C0
 404 69 'i'        12 4C 'L'        0 81        0 C1
 329 0A            12 57 'W'        0 82        0 C2
 329 0D            11 4D 'M'        0 83        0 C3
 299 65 'e'        10 58 'X'        0 84        0 C4
 295 72 'r'        10 62 'b'        0 85        0 C5
   ⋮                 ⋮                ⋮           ⋮      (中略)
  14 36 '6'         0 7E '~'        0 BE        0 FE
  12 34 '4'         0 7F            0 BF        0 FF
```

図2　追加した処理による印字例（出現回数順）

この処理のために，プログラムの行①の直後に，次の宣言を追加する。

```
int   ih, ix, code[256];
```

さらに，プログラムの行②の位置に，次の整列処理部を追加する。

〔整列処理部〕

```
        for (i = 0; i <= 255; i++)
           code[i] = i;
③→      ih = 255;
④→      while (ih > 0) {
⑤→         for (i = 0; i < ih; i++) {
⑥→            if (freq[i] < freq[i+1]) {
                 Swap(code[i], code[i+1]);
                 Swap(freq[i], freq[i+1]);
⑦→
               }
            }
⑧→          ih--;
        }
        printf("\n");
        for (i = 0; i < 64; i++) {
           for (chr = i; chr <= [ b ]; chr += [ c ]) {
              if ((0x20 <= code[chr]) && (code[chr] <= 0x7E))
                 printf(" %10ld %02X '%c'", freq[chr], code[chr], code[chr]);
              else
                 printf(" %10ld %02X    ", freq[chr], code[chr]);
           }
           printf("\n");
        }
```

ここで，整列処理部で使用する Swap(x, y) は，x と y の内容を入れ替えるために用意したマクロである。

整列処理部では，整列対象のデータ数が比較的少なく，また同じ出現回数の文字コードは元の並び順が維持されるので，バブルソートを使用している。

行④の while 文のブロックを 1 回実行すると，配列の走査範囲（要素番号 0 ～ ih）中の出現回数の最小値とその文字コードが，要素番号 ih の位置に置かれ，配列の走査範囲が 1 だけ狭められる。これを繰り返して整列を行う。この処理では while 文のブロックの実行回数は常に［ d ］回となる。

ここで，while 文のブロックの 1 回の繰返しにおいて，行⑥の if 文のブロック内の処理が最後に実行されたときの i の値を ix とすると，行⑤の for 文のブロックの実行が終了した時点で，配列 freq の要素番号［ e ］以降の要素の値は整列済みとなっている。これを利用して，整列処理部を表 1 に示すように変更すれば，while 文のブロックの実行回数を減らせる可能性がある。

表 1　整列処理部の変更内容

処置	変更内容
［ f ］の直後に追加	ix = 0;
行⑦の位置に追加	ix = i;
行⑧を置換え	［ g ］;

d に関する解答群

ア　253　　イ　254　　ウ　255　　エ　256

e に関する解答群

ア　256 − ix　　イ　ix − 1　　ウ　ix　　エ　ix + 1

f に関する解答群

ア　行③　　イ　行④　　ウ　行⑤

g に関する解答群

ア　ih = ix - 1　　イ　ih = ix　　ウ　ih = ix + 1　　エ　ix = ih