第60回

C言語プログラミング能力認定試験

２　級

解答時における注意事項

1.次の表に従って解答してください。

問題番号	問１～問８
選択方法	８問必須
試験時間	９０分

2.HB の黒鉛筆を使用してください。訂正する場合は，あとが残らないように消しゴムできれいに消し，消しくずを残さないでください。なお，ボールペンや万年筆等で記入した場合は，採点されません。

3.マークシート（解答用紙）の所定の欄に，級種，会場コード，受験番号を記入しマークしてください。また，会場名，氏名及びフリガナを所定の位置に記入してください。

4.解答は，次の例題にならって，「解答マーク欄」にマークしてください。

〔例題〕　日本の首都はどこか。

ア　東京　　イ　京都　　ウ　大阪　　エ　福岡

正しい答えは“ア　東京”ですから，次のようにマークしてください。

例題　■　∩イ∪　∩ウ∪　∩エ∪

指示があるまで開いてはいけません。 試験終了後，問題冊子を回収します。

受験会場	
受験番号	
氏　　名	

サーティファイ
情報処理能力認定委員会

問１～問８は，すべて必須問題です。全問について解答してください。
各設問の答えは，解答群の中から一つだけ選び，括弧中の設問番号に対応したマークシートの解答番号の「解答マーク欄」にマークしてください。なお，二つ以上マークした場合には不正解になります。

問１　Ｃ言語の特徴に関する次の記述の正誤を，解答群の中から選べ。

(1)　標準ライブラリ関数 rand は，ヘッダ stdlib.h で宣言されている。

(2)　記憶域クラス指定子 auto を指定して，外部変数を宣言することはできない。

(3)　Ｃ言語では，宣言指定子 union により構造体を定義することができる。

(4)　標準ライブラリ関数 fgets は，ストリームがファイルの終端に達している場合，NULL を返す。

(5)　論理 AND 演算子（&&）を使用して複数の条件式を記述した場合，真偽の判定は左から右へ順に行われ，一つでも偽と判定されると，それ以降の条件式は判定されない。

(6)　列挙型指定子 enum により定義される列挙定数において，3.14 のような実数を定義することができる。

(7)　int 型変数 i の値をキャスト演算子により double 型変数 d に代入する場合，「d = double(i);」と記述する。

(8)　標準関数 floor(double x)は，引数で与えた x 以下の最大の整数値を求めて，double 型で返却する。

解答群

ア　正しい　　イ　誤り

問２　ビット演算に関する次の記述中の［　　］に入れる適切な字句を，解答群の中から選べ。なお，unsigned short 型は 16 ビット長であるものとする。

& 演算子は［　(9)　］を，| 演算子は［　(10)　］を，^ 演算子は［　(11)　］を，~ 演算子は［　(12)　］のビット演算を行うために用いられる。

ビット演算子を使用した次のプログラムを実行すると，変数 a の値は［　(13)　］，変数 b の値は［　(14)　］となる。

```
unsigned short x = 0x29AE;
unsigned short y = 0x705C;
unsigned short a = x | y;
unsigned short b = x ^ y;
```

(9) ～ (12) の解答群

ア　ビットごとの論理和
イ　ビットごとの論理積
ウ　ビットごとの排他的論理和
エ　ビットの回転
オ　ビットの反転

(13)，(14) の解答群

ア　0x200C
イ　0x59F2
ウ　0x79FE
エ　0x8FA3
オ　0xD651

問３　次のプログラムを実行したとき，(15)～(19)で出力される値を，解答群の中から選べ。

＜プログラム＞

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

int main(void)
{
    char str[][10] = { "CIRCLE", "SQUARE", "TRIANGLE", "PENTAGON" };
    char buf[100];
    size_t len;

    len = strlen(str[3]);
    printf("%d¥n", len);                    … (15)

    strcpy(buf, str[2]);
    strcat(buf, str[1]);
    printf("%s¥n", buf);                    … (16)

    strcpy(buf, str[0]);
    strncpy(buf + 2, str[3], 5);
    buf[7] = '¥0';
    printf("%s¥n", buf);                    … (17)

    if (strcmp(str[1], "SQUAR") == 0) {
        strncpy(buf, str[1], 3);
        buf[3] = '¥0';
    } else {
        strncpy(buf, str[1], 2);
        buf[2] = '¥0';
    }
    printf("%s¥n", buf);                    … (18)

    strcpy(buf, str[3]);
    strcpy(buf + 5, strchr(str[2], 'G'));
    printf("%s¥n", buf);                    … (19)

    return 0;
}
```

(15) の解答群

ア 3　　イ 4　　ウ 8　　エ 9　　オ 10

(16) の解答群

ア SQUARE
イ SQUARELE
ウ SQUARETRIANGLE
エ TRIANGLE
オ TRIANGLESQUARE

(17) の解答群

ア CIPENTA
イ CIRCLEP
ウ CIRPENT
エ PECIRCL
オ PENTAGO

(18) の解答群

ア AR　　イ SQ　　ウ SQU　　エ UA　　オ UAR

(19) の解答群

ア PENTAGGLE
イ PENTAGLE
ウ PENTAGONGLE
エ PENTAGONLE
オ PENTALE

問４　main関数に関する次の記述中の［　　］に入れる適切な字句を，解答群の中から選べ。

main 関数は，二つの仮引数をもつ場合，次のように定義できる。

```
int main( [ (20) ] argc, [ (21) ] *argv[])
{
     ⋮
    return 0;
}
```

仮引数 argc には，実行時にコマンド行で指定された［ (22) ］が格納されて渡される。また，配列 argv の要素には，プログラム名と各引数の文字列のポインタが格納されて渡される。ここで，argv[argc]には［ (23) ］が格納されている。

コマンド行で

example△-size△1024△input.csv　　（△は空白文字を表す）

と指定して，次のプログラム（プログラム名 example）を実行すると，

```
a = [ (24) ]
b = [ (25) ]
```

と標準出力に出力される。

＜プログラム＞

```
#include <stdio.h>

int main( [ (20) ] argc, [ (21) ] *argv[])
{
    printf("a = %s¥n", argv[1]);
    printf("b = %c¥n", argv[2][3]);

    return 0;
}
```

(20), (21)の解答群

ア char　　イ double　　ウ float　　エ int　　オ void

(22)の解答群

ア　引数の個数
イ　引数の最大文字数
ウ　プログラム名と引数の個数の合計値
エ　プログラム名の文字数
オ　プログラム名の文字数と，すべての引数の文字数の合計値

(23)の解答群

ア "¥n"　　イ EOF　　ウ FILE　　エ NULL　　オ 不定値

(24)の解答群

ア -　　イ -size　　ウ 1024　　エ example　　オ size

(25)の解答群

ア -　　イ 2　　ウ 4　　エ n　　オ Z

問５　ファイルアクセスに関する次の記述中の［　　］に入れる適切な字句を解答群の中から選べ。

プログラムでファイルを使用する際には，関数fopenを呼び出し，引数で指定したファイル名のファイルをストリームに結びつける。関数fopenの返却値の型は［（26）］で，関数の呼出しに失敗した場合の戻り値は［（27）］となる。

ファイルに対する処理が終了した後，ストリームとファイルの結び付けを解除するには，［（28）］を引数に指定して関数［（29）］を呼び出す。

（26）の解答群

ア　char　　イ　FILE　　ウ　FILE *　　エ　int　　オ　int *

（27）の解答群

ア　'0'　　イ　""　　ウ　EOF　　エ　NULL　　オ　stderr

（28）の解答群

ア　-1
イ　NULL
ウ　関数 fopen の返却値
エ　ファイル名
オ　ファイル名の文字数

（29）の解答群

ア　fclose　　イ　fgets　　ウ　fputs　　エ　fscanf　　オ　ungetc

問６　次のプログラムを実行したとき，(30)～(34)で出力される値を，解答群の中から選べ。

＜プログラム＞

```
#include <stdio.h>

int main(void)
{
    char *istment[] = {
        "violin", "cello", "flute", "horn", "trumpet",
        "clarinet", "timpani"
    };
    char *p, **pp;
    int i, cnt;

    p = istment[2];
    printf("%s¥n", p);                          ･･･ (30)
    printf("%s¥n", p + 2);                      ･･･ (31)

    pp = istment;
    printf("%s¥n", *(pp + 4));                  ･･･ (32)
    printf("%s¥n", *pp + 4);                    ･･･ (33)

    cnt = 0;
    for (i = 0; i < 5; i++) {
        p = *(pp + i);
        while (*p != '¥0') {
            if (*p == 'l') {
                cnt++;
            }
            p++;
        }
    }
    printf("%d¥n", cnt);                        ･･･ (34)

    return 0;
}
```

(30) の解答群

ア　cello
イ　flute
ウ　horn
エ　trumpet
オ　violin

(31) の解答群

ア　flute
イ　horn
ウ　llo
エ　trumpet
オ　ute

(32) の解答群

ア　clarinet
イ　in
ウ　mpet
エ　timpani
オ　trumpet

(33) の解答群

ア　et　　イ　horn　　ウ　in　　エ　lo　　オ　te

(34) の解答群

ア　1　　イ　2　　ウ　3　　エ　4　　オ　5

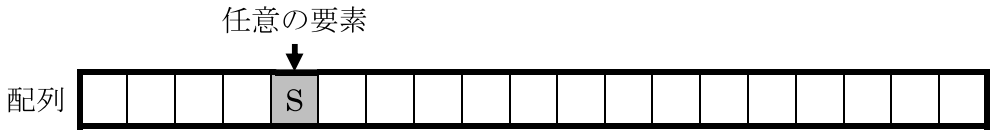
問７　次のプログラムの説明を読んで，プログラム中の［　　］に入れる適切な字句を，解答群の中から選べ。

＜プログラムの説明＞

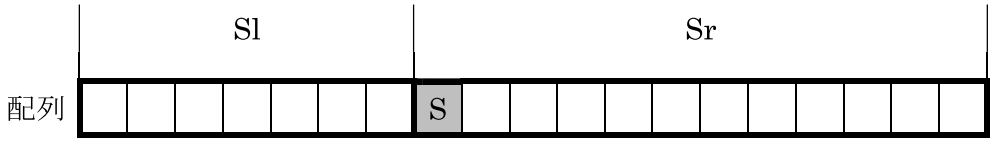
このプログラムは，クイックソートのアルゴリズムを使い，int型配列で渡されたN個のデータを昇順に並べ替える。

＜クイックソートによる並べ替えの手順＞

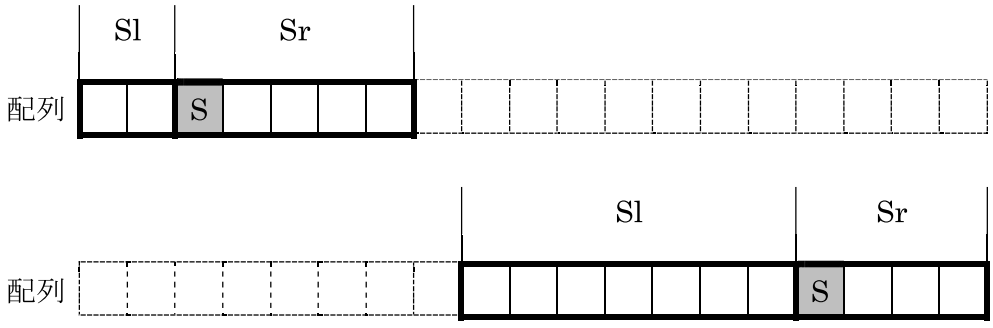
(1)　配列内の任意の要素（以下，Sという）を一つ選ぶ。



(2)　配列をS未満のグループ（以下，Slという）とS以上のグループ（以下，Srという）に分ける。



(3)　それぞれのグループ（Sの要素を除く）に対して，要素数が2以上の間(1)，(2)の処理を繰り返す。



(4)　データの並べ替え処理は関数QSortにより行われる。関数QSortでは，配列内の任意の要素（以下，基準値という）として，先頭要素を使用する。関数QSortの引数と返却値の仕様を，表１に示す。

表１　関数 QSort の引数と返却値の仕様

void QSort(int array[], int head, int tail)	
引　数	array　：並べ替え対象の配列 head　：並べ替え対象の先頭要素位置 tail　：並べ替え対象の末尾要素位置
返却値	なし

＜関数QSortの処理＞

(1)　並べ替え対象配列の先頭要素位置（以下，先頭位置という）が，並べ替え対象配列の末尾要素位置（以下，末尾位置という）より小さい場合，(2)以降の処理を行う。

(2)　並べ替え対象配列の先頭位置の要素の値を基準値とする。

(3)　Sl配列の格納位置（以下，lpという）と，Sr配列の格納位置（以下，rpという）を共に0に初期化する。

(4)　並べ替え対象配列の先頭位置＋1の要素位置から末尾位置までの値を参照し，(4-1)，(4-2)の処理を繰り返す（以下，参照する値を比較値という）。

(4-1)　比較値が基準値未満の場合，Sl配列の要素lpに比較値を格納し，lpをインクリメントする。

(4-2)　比較値が基準値以上の場合，Sr配列の要素rpに比較値を格納し，rpをインクリメントする。

(5)　Sl配列に格納されている全要素を，並べ替え対象配列の先頭から順に複写する。

(6)　(5)で複写した“並べ替え対象配列の末尾”の次要素位置（以下，確定位置という）に，基準値を格納する。

(7)　Sr配列に格納されている全要素を，並べ替え対象配列の確定位置の次の要素位置から順に複写する。

(8)　基準値未満の配列要素（Slグループ）について，関数QSortを呼び出して並べ替えを行う。

(9)　基準値要素を除く基準値以上の配列要素（Srグループ）について，関数QSortを呼び出して並べ替えを行う。

なお，Sl配列，Sr配列は外部定義され，十分な要素数をもつものとする。

＜プログラム＞

```
extern int sl[];
extern int sr[];
void QSort(int array[], int head, int tail)
{
    int s, i, j, t, lp, rp;

    if (head < tail) {
        s = (35);
        lp = 0;
        rp = 0;
        for ( (36) ) {
            if ( (37) ) {
                sl[lp] = array[i];
                lp++;
            } else {
                sr[rp] = array[i];
                rp++;
            }
        }
        for (i = 0; i < lp; i++) {
            array[head + i] = sl[i];
        }
        (38);
        array[t] = s;
        i++;
        for (j = 0; j < rp; j++) {
            (39);
        }

        QSort(array, head, t - 1);
        QSort(array, t + 1, tail);
    }
}
```

(35) の解答群

ア　array[head]
イ　array[tail]
ウ　head
エ　tail

(36) の解答群

ア　i = head; i < tail; i++
イ　i = head; i <= tail; i++
ウ　i = head + 1; i < tail; i++
エ　i = head + 1; i <= tail; i++

(37) の解答群

ア　array[i] == s
イ　array[i] != s
ウ　array[i] < s
エ　array[i] > s

(38) の解答群

ア　t = head + i
イ　t = i
ウ　t = i + 1
エ　t = tail - i

(39) の解答群

ア　array[head + i] = sr[j]
イ　array[head + i + j] = sr[j]
ウ　array[head + j] = sr[j]
エ　array[i + j] = sr[j]

問８　次のプログラムの説明を読んで，プログラム中の［　　］に入れる適切な字句を，解答群の中から選べ。

＜プログラムの説明＞

文字列（英数字，空白，および '-' 文字のみ）を15文字区切りで複数行に分割する。例えば，文字列が，

```
"AaaaaaaaaaaaaaaBbbbbbbbb"
```

のとき，分割後は，15文字で区切られて，

```
"Aaaaaaaaaaaaaaa"
"Bbbbbbbbb"
```

となる。ただし，文字列中に含まれている連続する数字と '-' 文字（以下，禁則文字という）の途中で行を区切らない。例えば，文字列が

```
"Aaaaaaaaaaaaaa1-2-3bbbbb"
```

のとき，分割後は，"1-2-3" は禁則文字であり途中で区切らないため，1行目は14文字で区切られて

```
"Aaaaaaaaaaaaaa"
"1-2-3bbbbb"
```

となる。ここで，数字と '-' 文字は，16文字以上連続することはないものとする。

文字列の分割は，関数Splitと，関数Splitから呼び出される関数IsProhibitedで処理する。以下に，関数Splitと関数IsProhibitedの仕様を示す。なお，1行に格納できる最大文字数（以下，最大文字数という）は，以下のように，マクロLINESIZEとして定義されている。

```
#define LINESIZE 15
```

＜関数Splitの仕様＞

表１　関数 Split の引数と返却値

int Split(char in[], char out[][LINESIZE + 1])	
引　数	in　:分割対象文字列 out　:分割後文字列配列
返却値	分割後文字列の行数

（機能）

引数inで渡された文字列を分割して，配列outに格納する。

（処理内容）

(1)　分割後文字列配列の行位置（以下，行位置という）と分割後文字列の行内文字位置（以下，文字位置という）を 0 に設定する。
(2)　分割対象文字列の参照位置（以下，走査位置という）を 0 に設定する。
(3)　分割対象文字列中の走査位置の文字（以下，対象文字という）が '¥0' でない間，(4)～(8)の処理を繰り返す。
(4)　文字位置が最大文字数と等しければ，分割後文字列配列の行位置，文字位置に '¥0' を格納した後，行位置をインクリメントし，文字位置を0に設定する。
(5)　対象文字が禁則文字であれば，(6)，(7)の処理を行う。禁則文字でなければ，(8)の処理を行う。
(6)　走査位置の文字から連続する禁則文字の文字数を求める。求めた文字数分の文字を，出力行の末尾に格納すると最大文字数を超えてしまう場合，分割後文字列配列の行位置，文字位置に'¥0'を格納した後，行位置をインクリメントし，文字位置を0に設定する。
(7)　走査位置の文字から連続する禁則文字を，出力行の文字位置へ複写する。複写しながら，文字位置と走査位置も，連続する禁則文字の文字分加算する。
(8)　走査位置の文字を出力行の文字位置へ複写し，文字位置と走査位置をインクリメントする。
(9)　分割後文字列配列の行位置，文字位置に '¥0' を格納し，分割後文字列の行数を返却する。

＜関数IsProhibitedの仕様＞

表２　関数 IsProhibited の引数と返却値

int IsProhibited(char ch)	
引　数	ch　：判定対象文字
返却値	禁則文字判定値（1：禁則文字である，0：禁則文字ではない）

（機能）

引数chで渡された文字が禁則文字かどうかを判定する。

（処理内容）

chが数字，または，'-' 文字であれば1を，そうでなければ0を返却する。

＜プログラム＞

```
#include <ctype.h>
#define LINESIZE 15

int IsProhibited(char ch)
{
    if ( (40) ) {
        return 1;
    }

    return 0;
}

int Split(char in[], char out[][LINESIZE + 1])
{
    int i, j, n, m, state;
    char ch;

    n = 0;
    m = 0;
    i = 0;
    while (in[i] != '¥0') {
        if ( (41) ) {
            out[n][m] = '¥0';
            n++;
            m = 0;
        }
        ch = in[i];
        state = IsProhibited(ch);
        if (state == 1) {
            j = i + 1;
            while (in[j] != '¥0') {
                if ( (42) ) {
                    break;
                }
                j++;
            }
            if ( (43) ) {
                out[n][m] = '¥0';
                n++;
                m = 0;
            }
            while (i < j) {
                out[n][m] = in[i];
                m++;
                i++;
```

```
            }

        } else {
            out[n][m] = ch;
            m++;
            i++;
        }
    }
    out[n][m] = '¥0';

    return [   (44)   ];
}
```

(40) の解答群

ア　isdigit(ch) || (ch == '-')
イ　isdigit(ch) || (ch != '-')
ウ　!isdigit(ch) && (ch == '-')
エ　!isdigit(ch) && (ch != '-')

(41) の解答群

ア　m == LINESIZE
イ　m != LINESIZE
ウ　m < LINESIZE
エ　m > LINESIZE

(42) の解答群

ア　IsProhibited(in[i + j]) == 0
イ　IsProhibited(in[i + j]) == 1
ウ　IsProhibited(in[j]) == 0
エ　IsProhibited(in[j]) == 1

(43) の解答群

ア　(LINESIZE - m) < j
イ　(LINESIZE - m) < (j - i)
ウ　m < j
エ　m < (j - i)

(44) の解答群

ア　m
イ　m + 1
ウ　n
エ　n + 1