第44回

C言語プログラミング能力認定試験

3　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６
選択方法	６問必須
試験時間	６０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，性別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６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)　Ｃ言語は，特定の計算機のアーキテクチャに依存するので，異なるハードウェアで動作させることのできる「移植可能なプログラム」をコーディングすることは困難である。

(2)　式をもつ `return` 文を実行すると，その式の値を関数呼出し式の値として呼出し元に返す。

(3)　Ｃ言語の注釈は，`/*`と`*/`の間にさらに`/*`と`*/`を記述することで，入れ子にすることができる。

(4) 標準ライブラリ関数 `isspace` は，引数に水平タブ(`'¥t'`)を指定された場合「真」を返す。

(5)　`unsigned int` は，符号無し整数型を表す。

(6) 「`int array[10] = {0, 1, 2};`」と宣言された配列 `array` の要素数は 3 となる。

(7) 「`while (式) 文`」で記述される `while` 文の「式」の部分には，反復処理の継続条件を記述する。

(8)　整数型変数 `a`，`b` に対して，「`a = b = 5;`」と「`b = 5; a = b;`」を行った場合の結果は同じである。

解答群

ア　正しい　　イ　誤り

問２　Ｃ言語の定数に関する次の記述の正誤を，解答群の中から選べ。ただし，'0'～'9'の文字コードは0x30～0x39であるものとする。

(9)　数値定数 0500 と数値定数 500 は，同じ値である。

(10) 数値定数 0xde と数値定数 0XDE は，同じ値である。

(11) 次のようにして初期化された文字配列 s は，等価である。

```
char s[] = "DEC";
```

と

```
char s[] = {'D', 'E', 'C'};
```

(12) 次のようにして初期化された変数 value は，同じ値である。

```
int value = 8;
```

と

```
int value = '8' - '0';
```

(13) int 型変数 i に対して，「i++;」と「i += 1;」を行った場合の結果は同じ値となる。

解答群

ア　正しい　　イ　誤り

問３　次の for 文を使ったプログラムを実行したとき，printf 関数によって出力される値を，解答群の中から選べ。

＜プログラム＞

```
#include <stdio.h>
int main(void)
{
    int i, j, a, b;
    int data[] = {5, 3, 4, 6, 2, 9, 7, 8};

    a = 0;
    for (i = 7; i > 2; i--)
        a += data[i];
    printf("%d¥n", i);                          ･･･ (14)
    printf("%d¥n", a);                          ･･･ (15)

    a = 0;
    for (i = 1; i < 6; i += 2)
        a += data[i - 1];
    printf("%d¥n", i);                          ･･･ (16)
    printf("%d¥n", a);                          ･･･ (17)

    a = 0;
    for (i = 0; i < 6; i++) {
        if (data[i] + data[i + 1] < data[i + 2]) {
            a = data[i + 2];
            break;
        }
    }
    printf("%d¥n", i);                          ･･･ (18)
    printf("%d¥n", a);                          ･･･ (19)

    a = 0;
    b = 0;
    for (i = 0; i < 7; i += 2) {
        for (j = i + 1; j < 8; j++) {
            if (data[i] - data[j] > 0) {
                a += data[i] - data[j];
                b++;
            }
        }
    }
    printf("%d¥n", a);                          ･･･ (20)
    printf("%d¥n", b);                          ･･･ (21)
    return 0;
}
```

(14) の解答群

ア -1　　イ 0　　ウ 1　　エ 2　　オ 3

(15) の解答群

ア 26　　イ 32　　ウ 36　　エ 39　　オ 44

(16) の解答群

ア 4　　イ 5　　ウ 6　　エ 7　　オ 8

(17) の解答群

ア 11　　イ 18　　ウ 20　　エ 26　　オ 29

(18) の解答群

ア 1　　イ 2　　ウ 3　　エ 4　　オ 5

(19) の解答群

ア 0　　イ 6　　ウ 7　　エ 8　　オ 9

(20) の解答群

ア 7　　イ 8　　ウ 9　　エ 10　　オ 11

(21) の解答群

ア 1　　イ 2　　ウ 3　　エ 4　　オ 5

問４　演算子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。

C言語の等値演算子には，二つの値が等しいとき「真」となる `==` と，二つの値が等しくないとき「真」となる［(22)］がある。

算術演算子には，`+`（加算），`-`（減算），［(23)］（乗算），`/`（除算），`%`（剰余算）がある。ここで，被除数，除数ともに整数による除算では，［(24)］が商となるため，整数型変数 `a`，`b`（`b` ≠ `0`）について，`a % b` の結果は［(25)］の結果と同じ値となる。

論理演算子には，［(26)］（論理積），`||`（論理和），［(27)］（論理否定）がある。

(22) の解答群

ア　`<`　　イ　`<=`　　ウ　`>`　　エ　`>=`　　オ　`!=`

(23) の解答群

ア　`,`　　イ　`*`　　ウ　`--`　　エ　`++`　　オ　`!`

(24) の解答群

ア　小数点以下が切り捨てられた整数値
イ　小数点以下が切り上げられた整数値
ウ　小数第１位で四捨五入した整数値
エ　実数値

(25) の解答群

ア　`(a * b) - (a / b)`　　イ　`a + (a / b) * b`
ウ　`a - (a / b) * b`　　エ　`b + (a / b) * a`
オ　`b - (a / b) * a`

(26)，(27) の解答群

ア　`,`　　イ　`++`　　ウ　`--`　　エ　`&&`　　オ　`!`

問５　次のプログラムを実行したとき，printf 関数によって出力される値を，解答群の中から選べ。なお，解答群の␣は空白文字を表す。

＜プログラム＞

```
#include <stdio.h>

int main(void)
{
    printf("[%03d]¥n", 29);                  ･･･ (28)
    printf("[%5o]¥n", 03417);                ･･･ (29)
    printf("[%6s]¥n", "SIX");                ･･･ (30)
    printf("[%-3s]¥n", "THREE");             ･･･ (31)
    printf("[%-7.4s]¥n", "SEVENFOUR");       ･･･ (32)

    return 0;
}
```

(28) の解答群

ア　[␣29]　　イ　[029]　　ウ　[29]
エ　[29␣␣]　　オ　[290]

(29) の解答群

ア　[␣3417]　　イ　[␣6531]　　ウ　[03417]
エ　[06531]　　オ　[3417]

(30) の解答群

ア　[␣␣␣SIX]　　イ　[SIX]　　ウ　[SIXSIX]
エ　[SIX␣␣␣]　　オ　[SSSSIX]

(31) の解答群

ア　[␣␣␣THR]　　イ　[THR]　　ウ　[THR␣␣␣]
エ　[THREE]　　オ　[REE]

(32) の解答群

ア　[␣␣␣SEVE]　　イ　[␣␣␣FOUR]　　ウ　[FOUR␣␣␣]
エ　[SEVE␣␣␣]　　オ　[SEVENFO]

問６　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＞

暗号化前文字列（80 文字以内），行補正値（0～5），列補正値（0～11）を標準入力から入力し，暗号化テーブルと行補正値，列補正値により文字の変換（暗号化）を行い，暗号化前文字列と暗号化後文字列を標準出力に表示するプログラムである。

ここで，暗号化前文字列として入力できる文字は，英数字（A～Z，a～z，0～9）及び次に示す記号 10 文字とする。

【暗号化前文字列として入力できる記号】

記号：_ , . ' " - ( ) : ;

暗号化前文字列として使用できない文字を入力した場合は，不正な文字としてその文字を標準出力に表示し，不正な文字を除いた文字列を暗号化の対象として処理を行う。

なお，暗号化前文字列は，必ず 80 文字以内の文字列が入力されるものとし，空白類文字（標準関数 isspace が「真」を返す文字）は入力されない。また，行補正値，列補正値には正しい数値が入力されるものとする。

＜実行例：不正な文字が含まれていない場合＞

```
行補正値(0 - 5) : 3
列補正値(0 - 11) : 7
暗号化前文字列(80文字まで) : This_is_a_pen.
暗号化前文字列 : This_is_a_pen.
暗号化後文字列 : u'IwOIwObOU;-r
```

＜実行例：不正な文字（$, %, #）が含まれている場合＞

```
行補正値(0 - 5) : 3
列補正値(0 - 11) : 7
暗号化前文字列(80文字まで) : This$is%a#pen.
不正な文字 : $
不正な文字 : %
不正な文字 : #
暗号化前文字列 : This$is%a#pen.
暗号化後文字列 : u'IwIwbU;-r
```

暗号化テーブルは，6 行 12 列の変換前テーブル（tbl）と，6 行 12 列の変換後テーブル（cvt）の二つからなり，暗号化前文字列から 1 文字取得し，その文字と一致する文字が tbl[m][n]に格納されているとき，cvt[m + 行補正値][n + 列補正値]に格納されている文字を暗号化後の文字とする。ここで，m + 行補正値，n + 列補正値が配列要素の上限を超えた場

合は，最下行の下，最右列の右及びその下にも変換後テーブルが配置されているものとして処理を行う。

例えば，暗号化前文字＝'t'，行補正値＝4，列補正値＝6 のとき，暗号化前文字't'は tbl[3][9]に格納されているので，cvt[3][9]の要素の行方向に+4，列方向に+6 した位置である cvt[1][3]に格納されている文字'f'が暗号化後文字となる。

【変換前テーブル(tbl)】

	0	1	2	3	4	5	6	7	8	9	10	11
0	A	B	C	D	E	F	G	H	I	J	K	L
1	M	N	O	P	Q	R	S	T	U	V	W	X
2	Y	Z	a	b	c	d	e	f	g	h	i	j
3	k	l	m	n	o	p	q	r	s	t	u	v
4	w	x	y	z	_	0	1	2	3	4	5	6
5	7	8	9	,	.	'	"	-	(	)	:	;

【変換後テーブル(cvt)】

	0	1	2	3	4	5	6	7	8	9	10	11	0	1	2	3	4	5	6	7	8	9	10	11
0	U	B	t	w	N	2	j	z	R	G	-	X	U	B	t	w	N	2	j	z	R	G	-	X
1	)	9	F	f	o	D	d	0	e	c	,	O	)	9	F	f	o	D	d	0	e	c	,	O
2	3	Q	J	S	:	7	8	a	Z	q	W	r	3	Q	J	S	:	7	8	a	Z	q	W	r
3	C	s	m	h	g	"	L	Y	4	x	6	5	C	s	m	h	g	"	L	Y	4	x	6	5
4	1	A	u	(	T	i	p	V	H	y	v	l	1	A	u	(	T	i	p	V	H	y	v	l
5	K	;	n	M	'	I	E	.	k	b	_	P	K	;	n	M	'	I	E	.	k	b	_	P
0	U	B	t	w	N	2	j	z	R	G	-	X	U	B	t	w	N	2	j	z	R	G	-	X
1	)	9	F	f	o	D	d	0	e	c	,	O	)	9	F	f	o	D	d	0	e	c	,	O
2	3	Q	J	S	:	7	8	a	Z	q	W	r	3	Q	J	S	:	7	8	a	Z	q	W	r
3	C	s	m	h	g	"	L	Y	4	x	6	5	C	s	m	h	g	"	L	Y	4	x	6	5
4	1	A	u	(	T	i	p	V	H	y	v	l	1	A	u	(	T	i	p	V	H	y	v	l
5	K	;	n	M	'	I	E	.	k	b	_	P	K	;	n	M	'	I	E	.	k	b	_	P

＜処理手順＞

(1) 標準入力から，行補正値，列補正値，暗号化前文字列を順に入力する。

(2) 暗号化前文字列の文字を先頭から末尾まで順に 1 文字ずつ取り出して，変換前テーブル内の一致する文字位置（行，列）を求める。

(3) 一致する文字がない場合は，不正な文字としてその文字を標準出力に表示する。

(4) 一致する文字がある場合は，求めた文字位置の行を行補正値で，列を列補正値で補正し，補正後の位置にある変換後テーブルの文字を求めて，暗号化後文字列に追加する。

(5) 暗号化前文字列と暗号化後文字列を標準出力に表示する。

＜プログラム＞

```
#include <stdio.h>
#define COLS 12
#define ROWS 6
#define LENG 80

int main(void)
{
    char tbl[ROWS][COLS] = {
        {'A','B','C','D','E','F','G','H','I','J','K','L'},
        {'M','N','O','P','Q','R','S','T','U','V','W','X'},
        {'Y','Z','a','b','c','d','e','f','g','h','i','j'},
        {'k','l','m','n','o','p','q','r','s','t','u','v'},
        {'w','x','y','z','_','0','1','2','3','4','5','6'},
        {'7','8','9',',','.','¥'','¥"','-','(',')',':',';'}
    };

    char cvt[ROWS][COLS] = {
        {'U','B','t','w','N','2','j','z','R','G','-','X'},
        {')','9','F','f','o','D','d','0','e','c',',','O'},
        {'3','Q','J','S',':','7','8','a','Z','q','W','r'},
        {'C','s','m','h','g','¥"','L','Y','4','x','6','5'},
        {'1','A','u','(','T','i','p','V','H','y','v','l'},
        {'K',';','n','M','¥'','I','E','.','k','b','_','P'}
    };

    char instr[LENG + 1], encstr[LENG + 1];
    int i, j, col, row;
    int ofsCol, ofsRow, cnvCol, cnvRow;
    char ch;

    printf("行補正値(0 - %d) : ", ROWS - 1);
    scanf("%d", &ofsRow);
    printf("列補正値(0 - %d) : ", COLS - 1);
    scanf("%d", &ofsCol);
    printf("暗号化前文字列(%d 文字まで) : ", LENG);
    scanf("%s", instr);

    /* 暗号化 */
    i = 0;
    j = 0;
    while ( (33) ) {
        ch = '¥0';
        for (row = 0; row < ROWS && ch == '¥0'; row++) {
            for (col = 0; col < COLS && ch == '¥0'; col++) {
                if (instr[i] == tbl[row][col]) {
```

```
                    cnvRow = [ (34) ];
                    cnvCol = (col + ofsCol) % COLS;
                    ch = cvt[cnvRow][cnvCol];
                }
            }
        }
        if ( [ (35) ] )
            printf("不正な文字 : %c¥n", instr[i]);
        else
            [ (36) ];
        i++;
    }

    [ (37) ];
    printf("暗号化前文字列 : %s¥n", instr);
    printf("暗号化後文字列 : %s¥n", encstr);

    return 0;
}
```

(33) の解答群

ア instr[i] == '¥0'　　イ instr[i] != '¥0'
ウ instr[i++] == '¥0'　　エ instr[i++] != '¥0'

(34) の解答群

ア (row + ofsCol) % COLS　　イ (row + ofsCol) % ROWS
ウ (row + ofsRow) % COLS　　エ (row + ofsRow) % ROWS

(35) の解答群

ア ch == '¥0'　　イ ch == instr[i]
ウ ch != '¥0'　　エ ch != instr[i]

(36) の解答群

ア encstr[i] = ch　　イ encstr[i++] = ch
ウ encstr[j] = ch　　エ encstr[j++] = ch

(37) の解答群

ア encstr[i] = '¥0'　　イ encstr[i] = ch
ウ encstr[j] = '¥0'　　エ encstr[j] = ch