第59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 

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)　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。

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

(3)　Ｃ言語では，#include 前処理指令によって読み込む対象のソースファイル内に，さらに#include 前処理指令を記述することはできない。

(4)　標準ライブラリ関数 toupper は，ヘッダファイル ctype.h で宣言されている。

(5)　関数内の任意の場所で関数の実行を終了するためには，return 文を使用する。

(6)　switch 文の制御式は，整数型でなければならない。

(7)　標準ライブラリ関数 isdigit は，10 進数字かどうかを判定する。

(8)　文字列中に改行を指定する場合は「¥t」と記述する。

解答群

ア　正しい　　イ　誤り

問２　入出力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。

ファイルの入出力を行うには，標準ライブラリ関数 fopen を呼び出し，アクセス対象のファイルに関する情報を保持する構造体［　(9)　］のポインタを取得する。その後は，取得したポインタを指定して，対象ファイルの入出力を行う。

ここで，関数fopenは，次の二つの引数を指定する。

・ファイルの名前

・ファイルの使い方を表すモード

ファイルの使い方を表すモードには，読込み("［　(10)　］")，書込み("［　(11)　］")，追加("a")が指定できる。関数fopenは，処理に失敗した場合，［　(12)　］を返す。

オープンしたファイルに文字を出力するには，標準ライブラリ関数［　(13)　］を呼び出す。また，入出力を終えたファイルを閉じるには，標準ライブラリ関数［　(14)　］を呼び出す。

(9) の解答群

ア enum　　イ EOF　　ウ FILE　　エ NULL　　オ size_t

(10), (11) の解答群

ア b　　イ r　　ウ t　　エ w　　オ x

(12) の解答群

ア EOF　　イ NULL　　ウ stderr　　エ void　　オ 負の値

(13), (14) の解答群

ア fclose　　イ fgetc　　ウ fputc　　エ fputs　　オ ungetc

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>

int x = 2, y = 3, z = 4;

int func1(int x)
{
    int y = 1;

    x--;
    return x + y + z;
}

int func2(int y)
{
    x += 4;
    return x + y + z;
}

int func3(int z)
{
    z += 7;
    y -= 5;
    return x + y + z;
}

int main(void)
{
    printf("%d¥n", x + y + z);       … (15)
    printf("%d¥n", func1(x));        … (16)
    printf("%d¥n", func2(x));        … (17)
    printf("%d¥n", func3(y));        … (18)
    printf("%d¥n", x + y + z);       … (19)

    return 0;
}
```

(15) の解答群

ア 5　　イ 6　　ウ 7　　エ 8　　オ 9

(16) の解答群

ア 5　　イ 6　　ウ 7　　エ 8　　オ 9

(17) の解答群

ア 10　　イ 11　　ウ 12　　エ 13　　オ 14

(18) の解答群

ア 11　　イ 12　　ウ 13　　エ 14　　オ 15

(19) の解答群

ア 5　　イ 6　　ウ 7　　エ 8　　オ 9

問４　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。

Ｃ言語のビット演算子には，

| 演算子 ･･･ (20)
& 演算子 ･･･ (21)
^ 演算子 ･･･ (22)

などがある。

次のプログラムを実行すると，変数 a の値は (23) ，変数 b の値は (24) ，変数 c の値は (25) となる。

＜プログラム＞

```
int main(void)
{
    unsigned short x = 0x2a9f, y = 0xf05b;
    unsigned short a, b, c;

    a = x | y;
    b = x & y;
    c = x ^ (unsigned short)0xffff;

    return 0;
}
```

(20) ～ (22) の解答群

ア　ビットごとの排他的論理和
イ　ビットごとの論理積
ウ　ビットごとの論理和
エ　ビットシフト
オ　ビット反転

(23), (24) の解答群

ア　0x1afa
イ　0x201b
ウ　0x3a44
エ　0xdac4
オ　0xfadf

(25) の解答群

ア　0x1afa
イ　0x3a44
ウ　0xd560
エ　0xdac4
オ　0xdfa4

問５　Ｃ言語の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。

以下は，Ｃ言語の演算子（一部抜粋）について，優先順位が高い順に並べた表である。α に記述される演算子は［　(26)　］，β に記述される演算子は［　(27)　］，γ に記述される演算子は［　(28)　］となる。

優先度	演算子
高	（略）
↑	<< >>
|	< <= > >=
|	== !=
|	（略）
|	α
|	β
↓	γ
低	（略）

次のプログラムを実行すると，［　(29)　］となる。

＜プログラム＞

```
int main(void)
{
    int p = 10, q = -1, r = 8, a, b, c;

    if (p > r || (q == r && p < r)) {
        a = 1;
    } else {
        a = 0;
    }
    if ((p > r || q == r) && p < r) {
        b = 1;
    } else {
        b = 0;
    }
    if (p > r || q == r && p < r) {
        c = 1;
    } else {
        c = 0;
    }

    return 0;
}
```

(26) ～ (28) の解答群

ア　`++　--`
イ　`?:`（条件演算子）
ウ　`||`
エ　`&&`
オ　`sizeof`

(29) の解答群

ア　`a = 0, b = 0, c = 1`
イ　`a = 0, b = 1, c = 0`
ウ　`a = 1, b = 0, c = 1`
エ　`a = 1, b = 1, c = 0`
オ　`a = 1, b = 1, c = 1`

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 *kyushu[] = {
        "Fukuoka", "Nagasaki", "Saga", "Kumamoto",
        "Oita", "Kagoshima", "Miyazaki" };
    char *p, **pp, *q;
    int i, j, cnt;

    p = kyushu[5];
    printf("%s¥n", p);                          ･･･ (30)
    printf("%s¥n", p + 4);                      ･･･ (31)

    pp = kyushu;
    printf("%s¥n", *(pp + 3));                  ･･･ (32)
    printf("%s¥n", *pp + 3);                    ･･･ (33)

    cnt = 0;
    for (i = 0; i < 6; i++) {
        p = kyushu[i];
        for (j = i + 1; j < 7; j++) {
            q = kyushu[j];
            if (*(p + 1) == *(q + 1)) {
                cnt++;
            }
        }
    }
    printf("%d¥n", cnt);                        ･･･ (34)

    return 0;
}
```

(30) の解答群

ア　Kagoshima
イ　Miyazaki
ウ　Nagasaki
エ　Oita
オ　Saga

(31) の解答群

ア　goshima　イ　hima　ウ　Kumamoto　エ　mamoto　オ　shima

(32) の解答群

ア　Kumamoto　イ　mamoto　ウ　moto　エ　Oita　オ　Saga

(33) の解答群

ア　amoto　イ　Kumamoto　ウ　kuoka　エ　Saga　オ　uoka

(34) の解答群

ア　2　イ　3　ウ　4　エ　5　オ　6

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＞

関数Sortは，配列に格納されている数列（要素数＞0）を，一定間隔だけ離れた配列要素でグループ分けし，それらのグループ内の要素について大小比較をして，配列の要素を昇順に並べ替える。グループ分けの間隔（以下，グループ間隔という）の初期値は，配列要素数を2で割った商（小数点以下は切り捨て）とし，グループ内の並べ替えが終わった時点で，グループ間隔を半分にして，再度グループ内の並べ替えを行う。グループ間隔が0になった時点で並べ替え処理を終了する。

例えば，配列要素数が 9 の場合，グループ間隔の初期値は 4 となり，以下に示すグループ１～４それぞれで昇順に並べ替えを行う。なお，網掛け部分はグループの構成要素を示す。

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
並べ替え対象グループ１	■				■				■
並べ替え対象グループ２		■				■			
並べ替え対象グループ３			■				■		
並べ替え対象グループ４				■				■	

図１　「グループ間隔＝４」でのグループ分け

グループ間隔＝4 での並べ替えが終わったら，グループ間隔を半分（＝2）にしてグループ分けして，再度グループ内で昇順に並べ替えを行う。

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
並べ替え対象グループ１	■		■		■		■		■
並べ替え対象グループ２		■		■		■		■	

図２　「グループ間隔＝２」でのグループ分け

グループ間隔＝2 での並べ替えが終わったら，グループ間隔を半分（＝1）にしてグループ分けして，再度グループ内で昇順に並べ替えを行う。ここで，グループ間隔が 1 の場合は，配列全要素が並べ替え対象となる。

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
並べ替え対象グループ１	■	■	■	■	■	■	■	■	■

図３　「グループ間隔＝１」でのグループ分け

＜関数Sortの処理＞

(1) グループ間隔を，配列要素数 ÷ 2で初期化する。
(2) グループ間隔が 0 より大きい間，(3)～(7)の処理を繰り返す。
(3) グループ内の先頭要素位置（以下，グループ先頭位置という）を0で初期化し，グループ間隔より小さい間，1加算しながら(4)～(6)の処理を繰り返す。
(4) グループ内での大小比較の限界位置（以下，限界位置という）を，「配列要素数 － グループ間隔」で初期化し，グループ間隔以上の間，グループ間隔の値の分を減算しながら(5)～(6)の処理を繰り返す。
(5) グループ内で大小比較要素の前位置（以下，比較前位置という）をグループ先頭位置で初期化し，限界位置より小さい間，グループ間隔の値の分を加算しながら(6)の処理を繰り返す。
(6) 比較前位置の配列要素と「比較前位置 ＋ グループ間隔」の配列要素を比較し，比較前位置の配列要素の方が大きかった場合，要素の値を交換する。
(7) グループ間隔を半分にする。

表１　関数 Sort の引数と返却値の仕様

void Sort(int a[], int n)	
引　数	a　：並べ替え対象数列 n　：並べ替え対象数列の要素数
返却値	なし

＜プログラム＞

```
void Sort(int a[], int n)
{
    int i, j, k, w, span;

    span = n / 2;
    while ( (35) ) {
        for (i = 0; i < span; i++) {
            for (j = n - span; j >= span; j -= span) {
                for ( (36) ) {
                    if ( (37) ) {
                        w = a[k + span];
                        (38);
                        a[k] = w;
                    }
                }
            }
        }
        (39);
    }
}
```

(35) の解答群

ア　span < n
イ　span > 0
ウ　span > 1
エ　span >= 0

(36) の解答群

ア　k = i; k < j; k += span
イ　k = i; k < j; k -= span
ウ　k = j; k < i; k += span
エ　k = j; k < i; k -= span

(37) の解答群

ア　a[i] < a[k + span]
イ　a[i] > a[k + span]
ウ　a[k] < a[k + span]
エ　a[k] > a[k + span]

(38) の解答群

ア　a[k + span] = a[k]
イ　a[k + span] = a[k - span]
ウ　a[k + span] = w
エ　a[k - span] = a[k]

(39) の解答群

ア　span++
イ　span -= 2
ウ　span *= 2
エ　span /= 2

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（1文字以上）を，固有の識別子（1～8文字）により管理するシステム（以下，文字列管理システムという）である。

文字列管理システムは，管理エリアと呼ばれる構造体 MANAGEの配列（要素数64）と，データエリアと呼ばれる文字型の2次元配列及び「初期化／追加／参照／削除」の四つの機能をもつ関数群で構成される。

構造体MANAGEは，以下のように，識別子，文字列長，次要素番号で定義されている。次要素番号は，管理する文字列が256文字を超えた場合，超えた分の文字列を保持する管理エリアの要素番号を保持する。なお，次要素番号の-1は，次要素がないことを意味する。

```
typedef struct {
    char name[9];       /* 識別子 */
    int len;            /* 文字列長 */
    int next;           /* 次要素番号 */
} MANAGE;
```

管理エリアmArea，およびデータエリアdAreaは，以下のように領域確保されている。データエリアは256バイトのブロックが64個分確保されている。

```
#define BCOUNT  64
#define BSIZE   256
MANAGE mArea[BCOUNT];           /* 管理エリア */
char dArea[BCOUNT][BSIZE];      /* データエリア */
```

管理エリアの各要素は，それぞれの要素番号に対応するデータエリアの状態を保持する。データエリアは，実際の文字列データを保持する。

例えば，識別子 "TEST"の文字列A（80文字）は，以下のように管理される。

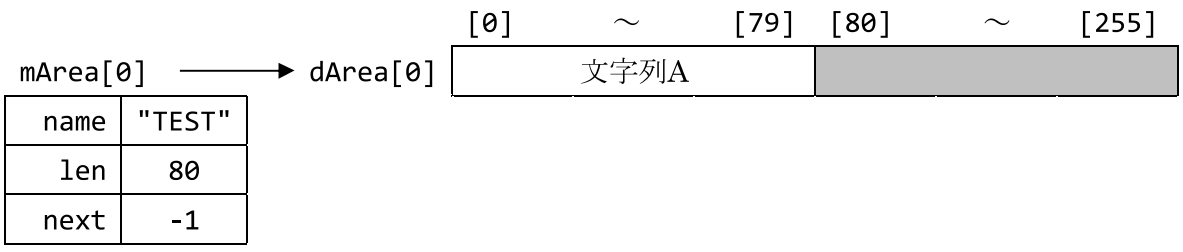

図１　識別子 "TEST" の文字列Ａの管理

また，識別子 "BEAT"の文字列B（532文字）は，以下のように管理される。

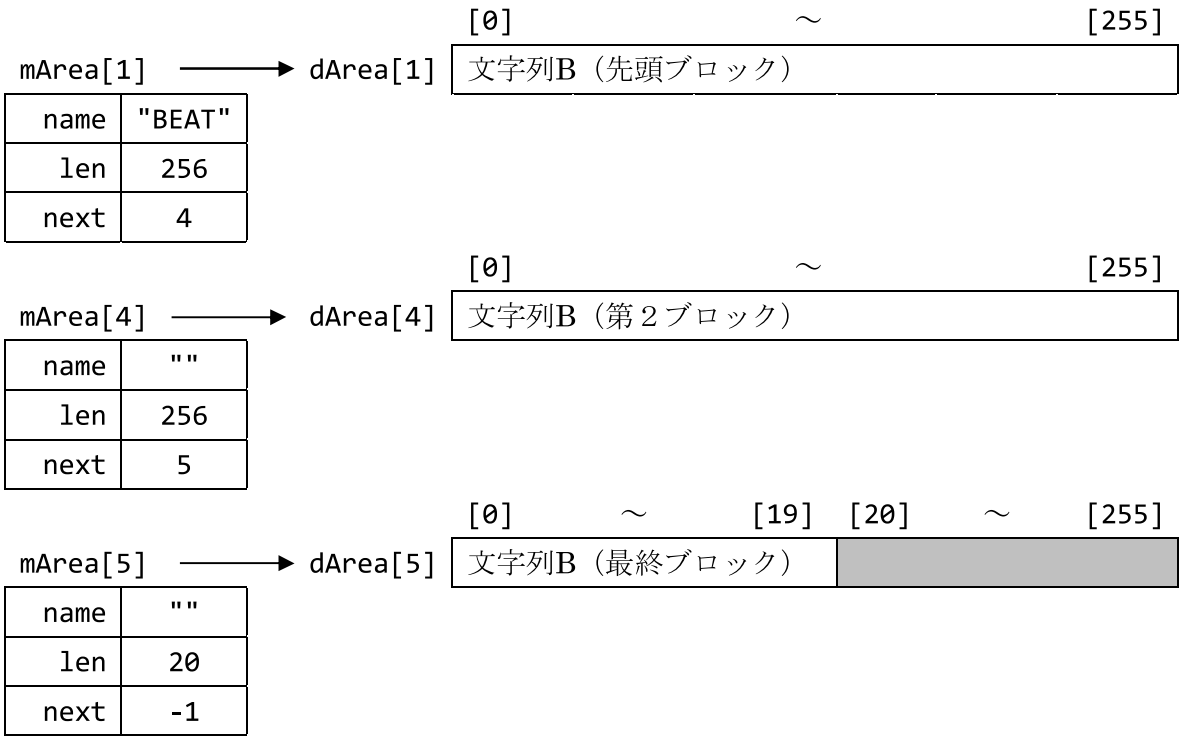

図２　識別子 "BEAT" の文字列Ｂの管理

以下，文字列管理システムの「初期化／追加／参照／削除」の機能を実現する関数Clear，関数Add，関数Get，関数Removeおよび共通関数Findについて説明しながら，管理エリアとデータエリアの仕様について説明する。

＜関数Clearの関数仕様＞

文字列管理システムの管理エリアを初期化する。

表１　関数 Clear の引数と返却値の仕様

void Clear(void)	
引　数	なし
返却値	なし

（処理内容）

管理エリアの全要素の「識別子」を空文字列に，「文字列長」を0に設定する。

「識別子」と「文字列長」の状態により，管理エリアの要素位置に対応するデータエリアの状態は以下のようになる。

表２　管理エリアの状態とデータエリアの状態の対応

識別子	文字列長	対応するデータエリアの状態
空文字列である	＞ 0	2ブロック目以降で使用中
	＝ 0	未使用
空文字列でない	＞ 0	先頭ブロックで使用中
	＝ 0	（存在しない状態）

＜共通関数Findの関数仕様＞

指定された識別子と一致する管理エリアの要素番号を返却する。

表３　関数 Find の引数と返却値の仕様

int Find(char name[])	
引　数	name　：識別子
返却値	0～　：指定された識別子と一致する管理エリアの要素番号 -1　：指定された識別子と一致する管理エリアの要素がなかった

（処理内容）

管理エリアの先頭要素から順に，管理エリアの「識別子」と引数で指定された「識別子」を比較し，一致したらその要素番号を返却する。一致しなかったら-1を返却する。

＜関数Addの関数仕様＞

引数で指定された「識別子」で，引数で指定された文字列を文字列管理システムに追加する。

表４　関数 Add の引数と返却値の仕様

int Add(char name[], char data[], int len)	
引　数	name　：識別子 data　：文字列 len　：文字列長
返却値	0　：同名の識別子が登録済みである -1　：文字列を登録するための十分なデータエリアがない 1　：正常終了

（処理内容）

(1) 共通関数Findを呼び出し，同名の識別子が登録済みであるか確認し，登録済みであれば，返却値0で関数を終了する。

(2) 文字列を登録するための十分なデータエリアがあるか，(2 － 1) ～(2 － 4) の処理で確認する。

(2 － 1) 文字列長をワーク変数に転写する。

(2 － 2) 文字列追加順要素番号配列（以下，追加配列という）の書込み位置を管理する変数（以下，書込位置という）を 0 に初期化する。

(2 － 3) 管理エリアの先頭から末尾まで，ワーク変数が 0 より大きい間，文字列長が 0 の要素を探す。見つかったら，追加配列の書込位置に，管理エリアの要素番号を格納し，書込位置を 1 加算し，ワーク変数から 256 を減算する。

(2 － 4) (2 － 3) の反復処理が終了した時点で，ワーク変数が 0 より大きかった場合，返却値-1 で関数を終了する。

(3) 文字列切出し先頭位置（以下，文字列切出位置という）を0に初期化する。また，追加配列の参照位置を管理する変数（以下，参照位置という）も0に初期化する。

(4) 参照位置が追加位置より小さい間，(4 － 1) ～(4 － 7) の処理を行う。

(4 － 1) 追加配列の参照位置の値を取り出し，参照位置が 0 の場合は，管理エリアの追加位置の「識別子」に，引数で指定された「識別子」を転写する。参照位置が 0 でない場合は，管理エリアの「参照位置 － 1」の「次要素番号」に追加位置を格納する。

(4 － 2) 管理エリアの追加位置の「次要素番号」に-1 を格納する。

(4 － 3) ワーク変数に文字列長を格納する。ただし，文字列長が 256 以上の場合は，ワーク変数に 256 を格納する。

(4 － 4) 管理エリアの追加位置の「文字列長」にワーク変数の値を格納する。

(4 － 5) データエリアの追加位置に，文字列切出位置からワーク変数に格納されている文字数分，文字列を複写する。

(4 － 6) 文字列切出位置にワーク変数の値を加算する。また，文字列長からワーク変数の値を減算する。

(4 － 7) 参照位置を 1 加算する。

(5) 返却値1で関数を終了する。

＜関数Getの関数仕様＞

引数で指定された「識別子」の文字列を参照する。

表５　関数 Get の引数と返却値の仕様

int Get(char name[], char data[])	
引　数	name　：識別子 data　：取得した文字列を格納する領域（十分な領域があるものとする） ※data の末尾には'¥0'は格納されない
返却値	0　　：識別子が登録されていない 1～　：取得した文字列の文字列長

（処理内容）

(1) 共通関数Findを呼び出し，識別子が登録されている要素番号を取得する。識別子が登録されていなければ，返却値0で関数を終了する。

(2) 文字列追加格納位置（以下，文字列格納位置という）を0に初期化する。

(3) データエリアの要素番号位置の文字列を，引数で指定された「文字列を格納する領域」の文字列格納位置へ，管理エリアの要素番号位置の「文字列長」分複写する。

(4) 文字列格納位置に管理エリアの要素番号位置の「文字列長」を加算する。

(5) 管理エリアの要素番号位置の「次要素番号」を，新たな要素番号位置とする。

(6) 要素番号位置が-1でない間，(3)～(5)の処理を繰り返す。

(7) 文字列格納位置を返却して，関数を終了する。

＜関数Removeの関数仕様＞

引数で指定された「識別子」の文字列を削除する。

表６　関数 Remove の引数と返却値の仕様

int Remove(char name[])	
引　数	name　：識別子
返却値	0　　：識別子が登録されていない 1　　：正常終了

（処理内容）

(1) 共通関数Findを呼び出し，識別子が登録されている要素番号を取得する。識別子が登録されていなければ，返却値0で関数を終了する。

(2) 管理エリアの要素番号位置の「識別子」を空文字列にする。

(3) 管理エリアの要素番号位置の「文字列長」に0を格納し，管理エリアの要素番号位置の「次要素番号」を，新たな要素番号位置とする。

(4) 要素番号位置が-1でない間，(3)の処理を繰り返す。

(5) 返却値1で関数を終了する。

＜プログラム＞

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

typedef struct {
    char name[9];       /* 識別子 */
    int len;            /* 文字列長 */
    int next;           /* 次要素番号 */
} MANAGE;

#define BCOUNT  64
#define BSIZE   256
MANAGE mArea[BCOUNT];           /* 管理エリア */
char dArea[BCOUNT][BSIZE];      /* データエリア */

void Clear(void)
{
    int i;

    for (i = 0; i < BCOUNT; i++) {
        mArea[i].name[0] = '¥0';
        mArea[i].len = 0;
    }
}

int Find(char name[])
{
    int i;

    for (i = 0; i < BCOUNT; i++) {
        if (strcmp(mArea[i].name, name) == 0) {
            break;
        }
    }
    if ( (40) ) {
        return -1;
    }

    return i;
}

int Add(char name[], char data[], int len)
{
    int i, j, k, pos, wlen, tc;
    int trace[BCOUNT];
```

```
    if (Find(name) != -1) {
        return 0;
    }

    wlen = len;
    i = 0;
    j = 0;
    while (i < BCOUNT && wlen > 0) {
        if (mArea[i].len == 0) {
            trace[j++] = i;
            (41);
        }
        i++;
    }
    if (wlen > 0) {
        return -1;
    }
    pos = 0;
    k = 0;
    while (k < j) {
        tc = trace[k];
        if (k == 0) {
            strcpy(mArea[tc].name, name);
        } else {
            mArea[trace[k - 1]].next = tc;
        }
        mArea[tc].next = -1;
        wlen = (42);
        mArea[tc].len = wlen;

        strncpy(dArea[tc], &data[pos], wlen);
        pos += wlen;
        len -= wlen;
        k++;
    }

    return 1;
}

int Get(char name[], char data[])
{
    int i, pos;

    i = Find(name);
    if (i == -1)
        return 0;
```

```
    pos = 0;
    do {
        strncpy(&data[pos], dArea[i], mArea[i].len);
        (43) ;
        i = mArea[i].next;
    } while ( (44) );

    return pos;
}

int Remove(char name[])
{
    int i;

    i = Find(name);
    if (i == -1) {
        return 0;
    }
    mArea[i].name[0] = '¥0';
    do {
        mArea[i].len = 0;
        i = mArea[i].next;
    } while ( (44) );

    return 1;
}
```

(40) の解答群

ア　i == 0
イ　i == BCOUNT
ウ　i != BCOUNT
エ　i < BCOUNT

(41) の解答群

ア　wlen++
イ　wlen--
ウ　wlen += BSIZE
エ　wlen -= BSIZE

(42) の解答群

ア　len == BSIZE ? len : BSIZE
イ　len != BSIZE ? len : BSIZE
ウ　len < BSIZE ? len : BSIZE
エ　len > BSIZE ? len : BSIZE

(43) の解答群

ア　pos++
イ　pos += mArea[i].len
ウ　pos -= BSIZE
エ　pos -= mArea[i].len

(44) の解答群

ア　i == -1
イ　i != -1
ウ　mArea[i].len != 0
エ　mArea[i].next == 0