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。

C

〔プログラム1の説明〕

U劇場の座席予約システムにおいて，希望する座席種別と座席数を指定すると，連続した空き（未予約）座席があるかどうかを調べるプログラムである。

(1)　U劇場の座席配置は図1に示すとおりである。

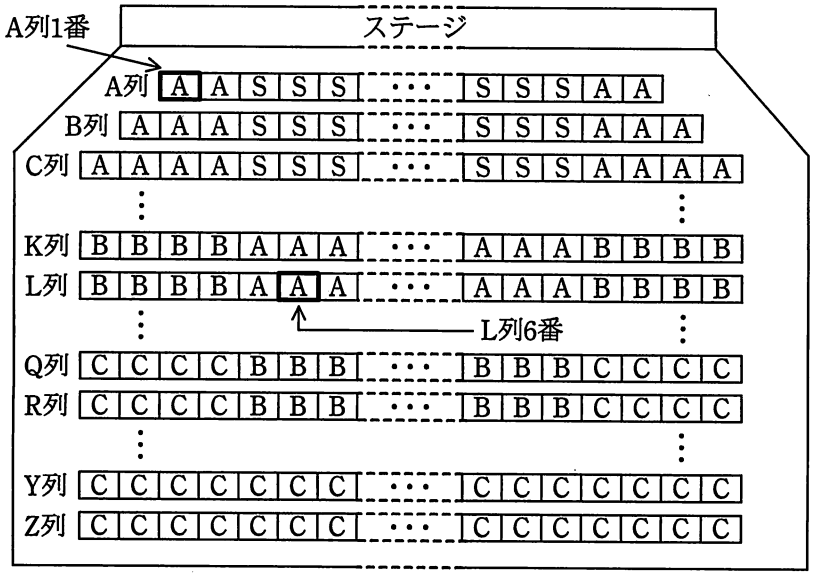

図1　U劇場の座席配置

① U劇場には，ステージ側から列名がA～Zの26列の座席があり，各列の座席数は異なる。

② 各列の座席は，ステージに向かって一番左から順に1，2，…と数え，列名のA～Zと，この数（以下，番という）を組み合わせた座席番号で識別される。例えば，“L列6番”の座席とは，12列目（L列）のステージに向かって一番左から6番目の座席を示す。

③ 各座席には，料金の違いを示す種別（以下，座席種別という）が割り振られている。座席種別は，料金が高いものから順にS，A，B，Cの四つがある。図1中の座席の文字は座席種別を表している。

(2) 空き座席を探すときには，ステージに向かって最前列の一番左（A列1番）から開始して，同一の列に希望する座席種別と座席数の連続した空きがあるかどうかを調べる。A列で見つからなかった場合は，順次B列，C列，…と後列を対象にして同様に調べる。

(3) 関数 check_seats の引数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。

class	希望の座席種別
num	希望の座席数
hall	座席表
pos	確認結果（POSITION 型の構造体へのポインタ）

① 連続した空き座席が見つかった場合，最初に見つかった空きの中で，ステージに向かって一番左に位置する座席の座席番号をposが指す構造体に格納する。

② 連続した空き座席が見つからなかった場合，pos が指す構造体の座席の列名にナル文字を格納する。

(4) 構造体 SEAT の構造は次のとおりである。

```
typedef struct {
    char seat_class; /* 座席種別 */
    int  reserved;   /* 予約状態 */
} SEAT;
```

(5) 構造体 POSITION の構造は次のとおりである。

```
typedef struct {
    char seat_row;   /* 座席の列名 */
    int  seat_no;    /* 座席の番 */
} POSITION;
```

(6) 座席表 hall の要素 hall[i][j]は SEAT 型の構造体であり，各座席の座席種別と予約状態が次のとおり格納されている。ここで，添字 i の 0～25 が列名 A～Z に，添字 j の 0～Ni－1が番 1～Ni(Niは，添字 i に対応する列の座席数)に対応している。

hall[i][j].seat_class：座席種別を表す“S”，“A”，“B”，“C”のいずれかの文字が格納されている。

hall[i][j].reserved：予約状態が格納されている。0 は空きを，1 は予約済を表す。

各列のステージに向かって一番右の座席の次の要素の座席種別 hall[i][Ni].seat_class には，ナル文字が格納されている。

〔プログラム 1〕

```
#define ROWNUM  26    /* 座席の列数 */

typedef struct {
    char seat_class;  /* 座席種別 */
    int  reserved;    /* 予約状態 */
} SEAT;
typedef struct {
    char seat_row;    /* 座席の列名 */
    int  seat_no;     /* 座席の番 */
} POSITION;

static char row_s[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

void check_seats(char, int, SEAT *[ROWNUM], POSITION *);

void check_seats(char class, int num, SEAT *hall[ROWNUM],
                 POSITION *pos) {
    int cnt, found = 0, no, row;

    for (row = 0; row < ROWNUM; row++) {
        cnt = 0;
        for (no = 0; hall[row][no].seat_class != '\0'; no++) {
            if ((hall[row][no].seat_class == class) &&
                (hall[row][no].reserved == 0)) {
                if (++cnt >= num) {
                    [ a ];
                    break;
                }
            } else {
                [ b ];
            }
        }
        if (found != 0) {
            break;
        }
    }

    if (found != 0) {
        pos->seat_row = [ c ];
        pos->seat_no  = [ d ];
```

```
    } else {
        pos->seat_row  = '\0';
    }
}
```

設問1　プログラム1中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　cnt = 0	イ　cnt--	ウ　cnt++
エ　found = 0	オ　found = 1	

c，dに関する解答群

ア　no	イ　no - num	ウ　no - num + 1
エ　no - num + 2	オ　row	カ　row + 1
キ　row_s[row]	ク　row_s[row + 1]	

設問2　関数 check_seats を使って，希望する座席種別の連続した空き座席があるかどうかを調べ，その結果を出力する。希望する座席種別の連続した空き座席がない場合には，ほかの座席種別で希望する座席数の連続した空き座席があるかどうかを調べ，その結果を出力するプログラムを作成した。作成したプログラムに関する説明文中の［　　］に入れる正しい答えを，解答群の中から選べ。

(1)　関数 check_service の引数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。

class	希望の座席種別
num	希望の座席数
hall	座席表
opt	結果の出力順

(2)　関数 check_service で使用している関数 print_seats の仕様は次のとおりである。

```
void print_seats(char class, int num, POSITION *pos)
```

機能：指定の座席種別の連続した空き座席があるかどうかの確認結果を出力する。

引数：class　座席種別
　　num　座席数
　　pos　確認結果（POSITION型の構造体へのポインタ）

〔プログラム2〕

```
#define ROWNUM  26    /* 座席の列数 */
#define CLSNUM   4    /* 座席種別数 */

typedef struct {
    char seat_class;  /* 座席種別 */
    int  reserved;    /* 予約状態 */
} SEAT;
typedef struct {
    char seat_row;    /* 座席の列名 */
    int  seat_no;     /* 座席の番 */
} POSITION;

static char class_s[CLSNUM] = "SABC";

void check_service(char, int, SEAT *[ROWNUM], int);
void check_seats(char, int, SEAT *[ROWNUM], POSITION *);
void print_seats(char, int, POSITION *);

void check_service(char class, int num, SEAT *hall[ROWNUM], int opt)
{
    int i;
    char c;
    POSITION pos;

    check_seats(class, num, hall, &pos);
    print_seats(class, num, &pos);
    if (pos.seat_row == '\0') {
        for (i =0; i < CLSNUM; i++) {
            if (opt == 0) {
                c = class_s[i];
            } else {
                c = class_s[CLSNUM - i - 1];
            }
            if (class != c) {
                check_seats(c, num, hall, &pos);
                print_seats(c, num, &pos);
            }
        }
    }
}
```

〔プログラム２の説明〕

(1) 希望する座席種別の連続した空き座席があるかどうかを調べ，その結果を出力する。

(2) 希望する座席種別の連続した空き座席がない場合には，希望する座席種別［ e ］に対して，結果の出力順 opt が 0 の場合は座席種別の料金が［ f ］ものから順に，結果の出力順 opt が 0 以外の場合は座席種別の料金が［ g ］ものから順に希望する座席数の連続した空きがあるかどうかを調べ，その結果を順次出力する。

C

eに関する解答群

ア　よりも料金の高いすべての座席種別
イ　よりも料金の安いすべての座席種別
ウ　よりも一つ料金の高い座席種別
エ　よりも一つ料金の安い座席種別
オ　を除くすべての座席種別

fに関する解答群

ア　高い　　イ　安い

gに関する解答群

ア　高い　　イ　安い