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1～3に答えよ。

C

本問は，図1に示す路線構成の鉄道模型における列車の運行をシミュレーションするプログラムである。表1は，図1の説明である。

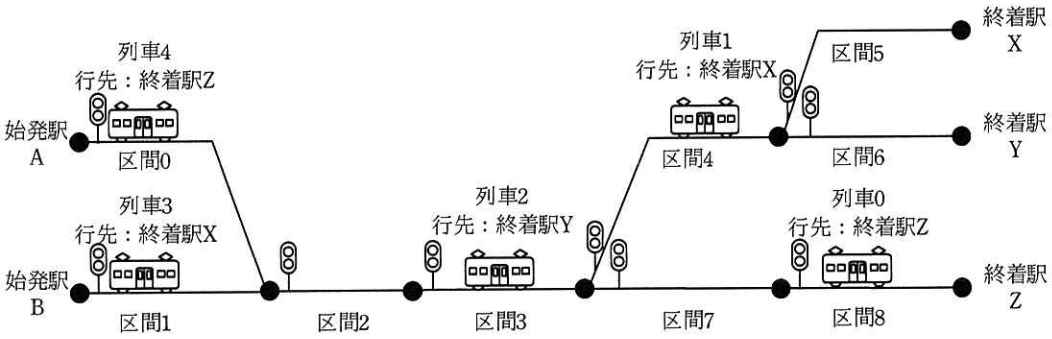

図1　鉄道模型の路線構成及び列車位置

表1　図1の説明

区間	区間の出口で接続する区間		区間内の列車（行先）
区間0	区間2	—	列車4（終着駅Z）
区間1	区間2	—	列車3（終着駅X）
区間2	区間3	—	—
区間3	区間4	区間7	列車2（終着駅Y）
区間4	区間5	区間6	列車1（終着駅X）
区間5	—	—	—
区間6	—	—	—
区間7	区間8	—	—
区間8	—	—	列車0（終着駅Z）

〔列車を進行させるルール〕

(1) 各列車は，あらかじめ決められた始発駅から，あらかじめ決められた終着駅を行先として，路線を後戻りすることなく進行する。

(2) 一つの区間には，同時に一つの列車しかいることができない。

(3) 各区間の入口には信号機があり，赤又は緑を表示する。

(4) 各列車は，終着駅を出口とする区間にいるときは無条件に進行できる。終着駅を出口とする区間にいる列車が進行した場合，その列車は路線上から取り除かれる。

(5) 各列車は，現在いる区間の出口で接続する区間（以下，次区間という）が次の条件をいずれも満たしたとき，次区間に進行できる。

① 次区間の信号機の表示が緑である。

② 次区間に進行すると，最終的には行先とする終着駅に到達できる。

(6) 次の二つの処理を繰り返し実行して各列車を進行させる。

① 路線を構成する全ての区間の信号機それぞれについて，区間内に列車がいるときは表示を赤に，列車がいないときは表示を緑にする。

② 路線を構成する全ての区間それぞれについて，次の処理を行う。

(ア) 終着駅を出口とする区間に列車がいる場合は，(4)に従って列車を進行させ，路線上から取り除く。

(イ) (ア)で述べた以外の区間に列車がいる場合は，(5)に従って進行できる列車を進行させ，進行した区間の信号機の表示を赤にする。

〔プログラム１の説明〕

(1) 構造体 block_info は，鉄道模型の区間を表現する。

```
struct block_info {
    struct train_info* train;
    struct block_info* next[2];
    int signal;
};
```

train： 区間内にいる列車を表現した，構造体 train_info へのポインタである。区間内に列車がいないとき，NULL を設定する。

next： 次区間を表現した，構造体 block_info へのポインタを格納する配列（配列の大きさは 2）である。終着駅を出口とする区間では，いずれの要素にも NULL を設定する。次区間が一つしかなければ，そのポインタは最初の要素に設定し，2 番目の要素には NULL を設定する。

signal： 信号機に表示する色を示す。次のいずれかの定数値をとる。

RED	表示は赤
GREEN	表示は緑

(2) 構造体 train_info は，鉄道模型の列車を表現する。

```
struct train_info {
    int number;
    struct block_info* dest;
};
```

number： 列車の番号である。

dest： 列車の終着駅を出口とする区間を表現した，構造体 block_info へのポインタである。

(3) 関数 set_signals の仕様は，次のとおりである。

機能： 全ての区間の信号機それぞれについて，区間内に列車がいるときは表示を赤に，列車がいないときは表示を緑にする。

引数：

blocks	全ての区間を表現した，構造体 block_info の配列
nblocks	blocks の要素数

〔プログラム 1〕

```
#include <stddef.h>

#define RED   (0)
#define GREEN (1)

struct train_info {
    int number;
    struct block_info* dest;
};

struct block_info {
```

```
    struct train_info* train;
    struct block_info* next[2];
    int signal;
};

void set_signals(struct block_info[], int);

void set_signals(struct block_info blocks[], int nblocks) {
    int i;
    struct block_info* block;
    for (i = 0; i < nblocks; i++) {
        block = &blocks[i];
        if ( [ a ] ) {
            block->signal = GREEN;
        }
        else {
            block->signal = RED;
        }
    }
}
```

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 に関する解答群

ア　block->next[0]->train == NULL

イ　block->next[1]->train == NULL

ウ　block->next[0]->train == NULL || block->next[1]->train == NULL

エ　block->next[0]->train == NULL && block->next[1]->train == NULL

オ　block->signal == RED

カ　block->train == NULL

キ　block->train != NULL

設問2　関数 proceed は，進行できる全ての列車を進行させるプログラムである。プログラム 2 中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔プログラム２の説明〕

(1) 関数 proceed の仕様は，次のとおりである。

機能： 全ての区間それぞれについて，進行できる列車がいるかどうかを判定し，進行できると判定された列車を進行させる。終着駅を出口とする区間にいる列車を進行させた場合，その列車を路線上から取り除く。出口で他の区間と接続する区間にいる列車を進行させた場合，進行した区間の信号機の表示を赤にする。

引数： blocks 全ての区間を表現した，構造体 block_info の配列

nblocks blocks の要素数

(2) 関数 proceed から呼び出される関数 find_block の仕様は，次のとおりである。

機能： 引数 block で示す区間が，引数 dest で示す区間に最終的に到達できる経路上にあるかどうかを判定する。

引数： block 判定対象の区間を表現した，構造体 block_info へのポインタ

dest 列車の終着駅を出口とする区間を表現した，構造体 block_info へのポインタ

返却値： 到達できるとき，1

到達できないとき，0

〔プログラム２〕

```
#include <stddef.h>

void proceed(struct block_info[], int);
int find_block(struct block_info*, struct block_info*);

void proceed(struct block_info blocks[], int nblocks) {
    int i, j;
    struct block_info* block;
    for (i = nblocks - 1; i >= 0; i--) {
        block = &blocks[i];
        if (block->train == NULL) {
            continue;
        }
        if ( [ b ] ) {
            block->train = NULL;
            continue;
```

```
            }
            for (j = 0; j < 2; j++) {
                if (block->next[j] == NULL) {
                    continue;
                }
                if (block->next[j]->signal == RED) {
                    continue;
                }
                if (find_block(block->next[j], block->train->dest) == 1) {
                    block->next[j]->train = block->train;
                    block->next[j]->signal = RED;
                    block->train = NULL;
                    [ c ];
                }
            }
        }
    }

    int find_block(struct block_info* block, struct block_info* dest) {
        int i;
        if (block == dest) {
            return 1;
        }
        for (i = 0; i < 2; i++) {
            if (block->next[i] == NULL) {
                continue;
            }
            if (find_block( [ d ] ) == 1) {
                return 1;
            }
        }
        return 0;
    }
```

b に関する解答群

ア　block->train->dest == NULL　　イ　block->train->dest != NULL

ウ　block == block->train->dest　　エ　block != block->train->dest

c に関する解答群

ア　break　　イ　continue　　ウ　return

dに関する解答群

ア　dest, block　　イ　dest, block->next[i]

ウ　block, dest　　エ　block, block->next[i]

オ　block->next[i], dest　　カ　block->next[i], block

C

設問3　図1に示した鉄道模型の路線構成及び列車位置を表現した，構造体block_infoの配列を第1引数とし，その要素数9を第2引数として，プログラム3を実行した。プログラム3の実行が終了したときの状態について述べた次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，blocks[n]は区間nを示す。

〔プログラム3〕

```
void start(struct block_info[], int);

void start(struct block_info blocks[], int nblocks) {
    int i;
    set_signals(blocks, nblocks);
    for (i = 0; i < 4; i++) {
        proceed(blocks, nblocks);
        set_signals(blocks, nblocks);
    }
}
```

列車4がいるのは区間［　e　］である。また，区間［　e　］の出口で接続する区間は［　f　］。

eに関する解答群

ア　0　　イ　2　　ウ　3　　エ　7

fに関する解答群

ア　一つあり，その信号機は緑を表示している

イ　一つあり，その信号機は赤を表示している

ウ　二つあり，それらの信号機はいずれも緑を表示している

エ　二つあり，それらの信号機はいずれも赤を表示している

オ　二つあり，それらの信号機は，一方が緑を，もう一方が赤を表示している