平成27年度 茨城大学大学院 理工学研究科
博士前期課程 入学試験問題

情報工学専攻

アルゴリズムとプログラミング

注意事項

1. 試験開始の合図があるまで，この問題冊子の中を見てはいけません．
2. この試験に使用する解答用紙は ４枚です．枚数を確認して，解答用紙の表題の右側にある（ ）内に，（その１）から（その４）までを記入してください．
   問題１の問１，問２に対し（その１）を，
   問題１の問３，問４に対し（その２）を，
   問題２の問１，問２に対し（その３）を，
   問題２の問３，問４に対し（その４）を使用してください．
3. 解答用紙の全てのページに，解答科目名と受験番号を記入してください．
   受験番号に記入漏れや記入間違いがあると，試験は無効になりますので，手元の受験票で確認のうえ注意深く記入してください．
   【上記の２と３に記載の事項は，試験開始前に必ず記入してください．】
4. 試験中に問題冊子の印刷不鮮明，ページの落丁・乱丁等に気が付いた場合は，手を上げて試験監督に知らせてください．
5. 携帯電話の電源は必ず切ってカバン等に入れ，身に着けないでください．また，携帯電話を時計として使用することはできません．
6. 試験中に辞書類・電卓を使用することはできません．

英語対応表

エイトクイーン	eight-queen
クイーン	queen
チェス盤	chessboard
マス（升目）	square
行	row
列	column
配列	array
要素	element
再帰呼出し	recursive call
双方向リスト	doubly-linked list
データ構造	data structure
先頭ノード	head node
最後尾ノード	tail node
双方向循環リスト	circular doubly-linked list
自然数	natural number
格納する	store
消去する	delete
構造体	structure
構造体型	structure type
定義する	define
ダミーノード	dummy node
関数	function
前後のポインタ	previous and next pointer
先頭アドレス	beginning address
末尾	tail
追加する	add
標準入力	standard input
繰り返す	repeat
削除する	delete
逆の順番	reverse order

アルゴリズムとプログラミング

問題 1　エイトクイーンとは，8 × 8 マスのチェス盤上に 8 つのクイーンを配置する問題である．クイーンは縦，横，斜めの方向の任意のマス目に移動することができる．この問題では，それぞれのクイーンの移動可能なマスに他のクイーンが存在しないように配置しなければならない．

図 1-1 にそのような配置，すなわちこの問題の解の例を示す．「Q」と記述されたマスがクイーンの位置を示す．なお，C 言語の配列要素と対応させるため，0 から始まる行番号と列番号を付記している．

エイトクイーンの解は全部で 92 種類ある．ここでは 92 種類の解すべてを表示するプログラムを作成する．
まず，クイーンは一行に 1 つしか配置できないため，各行のクイーンが第何列にあるかを示すことで，チェス盤上のすべてのクイーンの配置を C 言語の配列で表現することができる．たとえば図 1-1 の解は {4,0,3,5,7,1,6,2} と表現できる．また，まだクイーンが第何列にあるかが決まっていないことを-1 の値で示すことにする．たとえば，第 3 行以降の位置が決まっていない場合は，{4,0,3,-1,-1,-1,-1,-1} のように表現される．

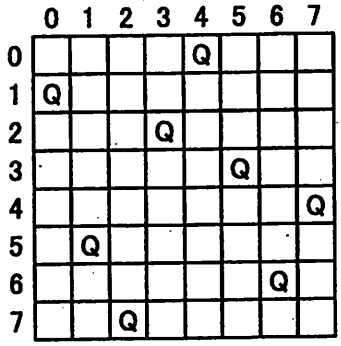

図 1-1

この配列のすべての要素を 0~7 まで変化させた組み合せの中に解が存在する．この処理は配列要素の 1 つを 0 から 7 まで変化させる関数を再帰呼び出しすることで実現できる．

ただし，すべての組み合わせを生成する必要はない，組み合わせを生成する過程で 1 つでも条件を満たさないクイーンが現れれば，残りのクイーンがどこにあっても解とはならないからである．

この確認は，縦と斜めの並びのどこにクイーンが配置済みであるかを記録する配列 flagc，flagl，flagr を用いれば容易に実現できる．図 1-2 に例示するように，まだクイーンが配置されていない並びには 0，配置されている並びには 1 を記録することにする．flagl には左斜め上に位置するマスにクイーンが配置済みであるかどうかが記録され，第 7 行第 0 列が flagl[0]，第 6 行第 0 列と第 7 行第 1 列が flagl[1]，...，第 0 行第 7 列が flagl[14] へそれぞれ対応するものとする．同様に flagr は右斜め上に位置するマスへの配置状態が記録される．すでに {4,0,-1,-1,-1,-1,-1,-1} のように第 1 行まで配置済みの場合，flagc は {1,0,0,0,1,0,0,0}，flagl は {0,0,0,0,0,0,1,0,0,0,0,1,0,0,0}，flagr は {0,1,0,0,1,0,0,0,0,0,0,0,0,0,0} となる．第 2 行はこれらの配列を調べれば，第 0，1，2，4，6 列にクイーンを配置できないことがわかる．

リスト 1 にエイトクイーンのすべての解を表示するプログラムを示す．関数 void eight_queen(int row) は第 row 行以降のクイーンの位置を決定し，すべての行のクイーンの位置が決定したらその解を表示する関数である．最初に関数 int main(void) から呼び出されるときには，row には 0 が与えられる．関数 void eight_queen(int row) 内で宣言される board はチェス盤上の各行の第何列にクイーンが配置されているかを記録する配列である．また，flagc，flagl，flagr は，上述のとおり，縦と斜めの並びにすでにクイーンが配置されているかどうかを記録する配列である．空欄 (1) には，第 row 行第 col 列のマスにクイーンを配置できるかどうかを判定する条件式が入る．また，空欄 (2) には，第 row 行第 col 列のマスにクイーンを配置する処理と第 row+1 行以降のクイーンの配置を決める再帰呼び出しが含まれる．さらに，再帰呼び出しの

直前には flagc，flagl，flagr の各配列の値の設定，再帰呼び出しの直後には値を元に戻す処理も必要となる．そして，空欄 (3) には条件を満たす解を図 1-3 のように表示する処理が入る．

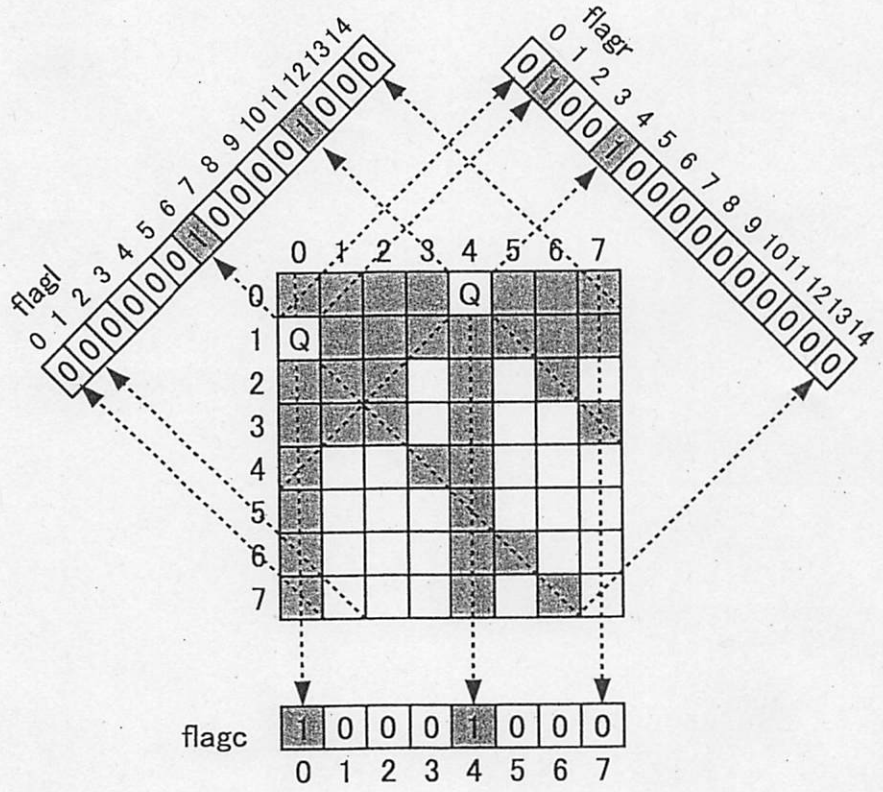

図 1-2

```
+-+-+-+-+-+-+-+-+
| | | | |Q| | | |
+-+-+-+-+-+-+-+-+
|Q| | | | | | | |
+-+-+-+-+-+-+-+-+
| | | |Q| | | | |
+-+-+-+-+-+-+-+-+
| | | | | |Q| | |
+-+-+-+-+-+-+-+-+
| | | | | | | |Q|
+-+-+-+-+-+-+-+-+
| |Q| | | | | | |
+-+-+-+-+-+-+-+-+
| | | | | | |Q| |
+-+-+-+-+-+-+-+-+
| | |Q| | | | | |
+-+-+-+-+-+-+-+-+
```

図 1-3

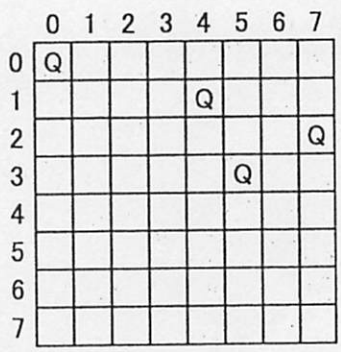

図 1-4

リスト１：エイトクイーンのすべての解を表示するプログラム

```
#include <stdio.h>

void eight_queen(int row);

int main(void) {
    eight_queen(0);
    return 0;
}

void eight_queen(int row) {
    static int board[] = {-1,-1,-1,-1,-1,-1,-1,-1};
    static int flagc[] = {0,0,0,0,0,0,0,0};
    static int flagl[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    static int flagr[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    int col;

    if (row < 8) {
        for(col = 0; col < 8; col++) {
            if ( (1) ) {
                (2)
            }
        }
    } else {
        (3)
    }
    return;
}
```

以上の説明に基づき，以下の問１～問４に答えよ．

問１ 図1-4は，リスト１のプログラムで最初に見つかる解の第３行までを示したものである．この解を完成させ，図または説明文中の配列表現（{4,0,3,5,7,1,6,2}のような表現）で示せ．

問２ リスト１の空欄(1)に適切な記述を示せ．

問３ リスト１の空欄(2)に適切な記述を示せ．なお変数が不足する場合には，新しい変数を宣言して使用してもよい．

問４ リスト１の空欄(3)に適切な記述を示せ．なお変数が不足する場合には，新しい変数を宣言して使用してもよい．

問題２　双方向リストとは，保持する情報と前後のノードへのポインタからなるノードが相互に連結されたデータ構造である．この双方向リストにおいて，先頭と最後尾のノードを相互に連結させ，環状につないだデータ構造を双方向循環リストと呼ぶ．双方向循環リストの構造を図 2-1 に示す．

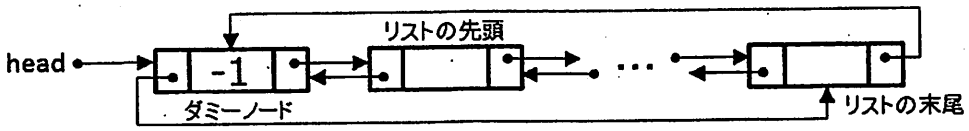

図 2-1: 双方向循環リスト

この双方向循環リストに n 個の自然数を格納し，指定した値を持つノードをすべて消去するプログラムを C 言語で作成する．このプログラムの一部をリスト２に示す．このプログラムではノードを表現するために構造体 _node を定義し，この構造体型の別名として node 型を定義する．node 型を用いた双方向循環リストの作成について，以下の問１～問４の設問に答えよ．

問１ 双方向循環リストを作成するために，図 2-2 に示すようなダミーノードを導入し，それを node 型ポインタ head が指すリストの先頭ノードとすることを考える．ダミーノードは関数 `node *createDummy(void)` を呼び出して作成される．この関数の中では，関数 `node *createNode(-1)` によって対象外の数値である −1 が格納されたノードを作成し，その先頭アドレスを返す処理が行われる．ここで使用する関数 `node *createNode(int val)` は，新しいノードを作成する関数 `node *nodeAlloc(void)` で得たひとつのノードに対して，value フィールドに自然数 val を格納し，前後へのポインタは自分自身を指し，その先頭アドレスを返す．関数 `node *createNode(int val)` を作成せよ．

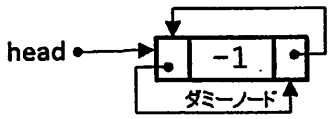

図 2-2: ダミーノード

問２ 問１で取得したダミーノードだけからなる双方向循環リストにデータを入力する．関数 `void createList(node *head)` はリストの先頭アドレス head を入力とし，入力データが格納されたノードをリストの末尾に追加する処理を繰り返し行う．ここで，リストの末尾とは，直後のノードへのポインタの値が head の値と同じノードを意味する．この関数内は，関数 scanf でキーボードから自然数を読み込んで新しいノードに格納し，リストに連結する処理を，入力時に関数 scanf が EOF を返すまで繰り返すという処理内容となっている．この処理を行う関数 `void createList(node *head)` 作成せよ．

問３ 問２で双方向循環リストに入力したデータの中から，特定の値を持つすべてのノードを削除する．あらかじめ関数 `int main(void)` で消去する自然数 q を入力し，双方向循環リストからこの値を削除するために，関数 `void removeNumber(head, q)` を呼び出す．関数 `void removeNumber(node *head, int val)` はリストの先頭アドレス head と削除する値 val を入力とし，値 val が格納されたノードをすべて削除する処理を行う．関数 `void removeNumber(node *head, int val)` を作成せよ．

問4 双方向循環リストに入力されたデータに対して，リストの末尾から先頭に向かってノードに格納された値をすべて出力する関数 `void printList_reverse(node *head)` を作成せよ．ここで，リストの先頭とは，直前のノードへのポインタの値が head の値と同じノードを意味する．関数 `void printList_reverse(node *head)` は，入力がリストの先頭アドレス head で，入力された順序と逆の順番で格納データを出力する処理を行う．ただし，ダミーノードは出力に含めないものとする．

リスト2 双方向循環リストに自然数を格納し，指定した値を持つノードを消去するプログラム

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

typedef struct _node {
    int value;          // 正の整数値を格納する変数
    struct _node *prev; // 直前のノードへのポインタ
    struct _node *next; // 直後のノードへのポインタ
} node;

node *createDummy(void);
node *createNode(int);
void createList(node *);
void removeNumber(node *, int);
void printList_reverse(node *);
node *nodeAlloc(void);

int main(void) {
    node *head;
    int q;

    // -1 の値を持つダミーノードを作成する
    head = createDummy();

    // 双方向循環リストにデータを入力する
    createList(head);
    // 標準入力の EOF 状態を解除する．
    clearerr(stdin);

    // 特定の値を持つデータを削除する
    printf("消去する値を入力\n");
    scanf("%d", &q);
    removeNumber(head, q);
```

(次のページに続く)

リスト２　双方向循環リストに自然数を格納し，指定した値を持つノードを消去するプログラム(続き)

```
    // 逆順でデータ表示する
    printList_reverse(head);

    return 0;
}

// ダミーノードを作成する
node *createDummy(void) {
    node *dummynode;
    dummynode = createNode(-1);
    return dummynode;
}

// 新しくノードを作成する
node *createNode(int val) {
    // 問１
}

// 双方向循環リストにデータを入力する
void createList(node *head) {
    // 問２
}

// 双方向循環リストのデータから，特定の値を持つデータを削除する
void removeNumber(node *head, int val) {
    // 問３
}

// 双方向循環リストの末尾から先頭に向かって要素の値を出力する
void printList_reverse(node *head) {
    // 問４
}

// ノード作成のためにメモリを割り当てる
node *nodeAlloc(void) {
    return (node *)malloc(sizeof(node));
}
```