平成２８年度

名古屋大学大学院情報科学研究科
情報システム学専攻
入学試験問題

専門

平成２７年８月６日（木）
１２：３０～１５：３０

注意事項

１．試験開始の合図があるまでは、この問題冊子を開いてはならない。

２．試験終了まで退出できない。

３．外国人留学生は英語で解答してよい。また、和英辞書などの辞書を１冊に限り使用してよい。電子辞書の持ち込みは認めない。

４．問題冊子、解答用紙3枚、草稿用紙3枚が配布されていることを確認せよ。

５．問題は(１)解析・線形代数、(２)確率・統計、(３)プログラミング、(４)計算機理論、(５)ハードウェア、(６)ソフトウェアの６科目がある。(４)~(６)の３科目から少なくとも１科目を選択して解答し、(１)~(３)を含めた６科目から合計３科目を選択して解答せよ。
なお、選択した科目名を解答用紙の指定欄に記入せよ。

６．解答用紙は指定欄に受験番号を必ず記入せよ。解答用紙に受験者の氏名を記入してはならない。

７．解答用紙に書ききれない場合は、裏面を使用してもよい。
ただし、裏面を使用した場合は、その旨、解答用紙表面右下に明記せよ。

８．解答用紙はホッチキスを外さず、試験終了後に3枚とも提出せよ。

解析・線形代数

（解の導出過程も書くこと）

[1] 次の極座標の方程式で表される曲線について、以下の問いに答えよ。

$$r = 1 + \cos\theta \qquad (0 \le \theta \le 2\pi)$$

(a) 曲線の概形を図示せよ。

(b) 曲線の長さを求めよ。

[2] 次の2次形式について、以下の問いに答えよ。ただし、$x = \begin{pmatrix} x \\ y \end{pmatrix}$, $x' = \begin{pmatrix} x' \\ y' \end{pmatrix}$ とし、${}^t x$ で x の転置を表す。

$$Q(x, y) = x^2 - 2\sqrt{3}xy - y^2 \tag{1}$$

(a) 式 (1) は、対称行列 A を用いて $Q(x, y) = {}^t x A x$ と表せる。A のすべての固有値を求めよ。

(b) (a) の A を対角化する直交行列を求め、A を対角化せよ。

(c) 式 (1) は、ある線形変換 $x = Ux'$ により標準形、すなわち、$x'y'$ の項を含まない2次形式で表せる。式 (1) を標準形に直せ。

(d) $Q(x, y) = 2$ の概形を図示せよ。

[3] 次の条件を満たす複素数 $z = x + iy$ について、以下の問いに答えよ。ただし、R は実数の集合、i は虚数単位を表す。

$$z + \frac{1}{z} \in R$$

(a) $z + \frac{1}{z} = u + iv$ とおくとき、u, v を x, y を用いて表せ。

(b) z が複素平面上でえがく図形を求め、その概形を図示せよ。

(c) $|z - 1| \le 2$ であるとき、$|z - 2 + i|$ の最大値および最小値を求めよ。

Translation of technical terms

極座標	polar coordinates	方程式	equation
曲線	curve	概形	rough sketch
長さ	length	2次形式	quadratic form
転置	transpose	対称行列	symmetric matrix
固有値	eigenvalue	対角化	diagonalization
直交行列	orthogonal matrix	線形変換	linear transformation
標準形	normal form	複素数	complex number
実数	real number	集合	set
虚数単位	imaginary unit	複素平面	complex plane
図形	shape	最大値	maximum value
最小値	minimum value		

確率・統計　解の導出過程も書くこと.

[1] じゃんけん(ルールは下記※を参照)で3人の順位(1〜3位)を決めたい. 但し, じゃんけんの各回の試行は独立で, 3人とも3種類の手を等確率で出すものとする. これについて, 以下の問いに答えよ.

(1) 1回目のじゃんけんで, 3人があいことなる確率を求めよ.

(2) 2回目のじゃんけんで, 3人の順位が全て決まる確率を求めよ.

(3) n 回目のじゃんけんで, はじめて1位が決まる確率を求めよ. 但し, n は正整数とする.

※じゃんけんとは, 複数の人が同時にグー・チョキ・パーの3種類の手のいずれかを出すことで, 勝敗を決める手段である. グーはチョキに勝ち, チョキはパーに勝ち, パーはグーに勝つ. 全員が同じ手を出した場合や, 3人が異なる手を出した場合は, あいことする. まず, 3人のじゃんけんで1人が勝った場合は, 勝った人が1位となり, その後負けた2人でじゃんけんをして, その勝敗で2位と3位を決める. 一方, 3人のじゃんけんで1人が負けた場合は, 負けた人が3位となり, その後勝った2人でじゃんけんをして, その勝敗で1位と2位を決める. また, あいことなった場合は, 勝敗が決まるまで同じメンバーでじゃんけんを繰り返す.

[2] 互いに独立な確率変数 X, Y について, 以下の問いに答えよ.

(1) X, Y の期待値と分散が, それぞれ $E(X) = 2$, $V(X) = 1$, $E(Y) = 5$, $V(Y) = 9$ で与えられるとき, 確率変数 $W = (X - 2Y)^2$ の期待値 $E(W)$ を求めよ.

(2) X, Y がともに区間 $[1,2]$ における連続一様分布に従うとき, 確率変数 $Z = \max\{X, Y\}$ の確率密度関数 $f_Z(z)$ を求めよ.

[3] 確率変数 X, Y の同時確率密度関数 $f_{X,Y}(x, y)$ が次式で与えられている. 但し, c は定数とする. これについて, 以下の問いに答えよ.

$$f_{X,Y}(x,y) = \begin{cases} ce^{-x-y}, & 0 \le x \le y \\ 0, & \text{その他} \end{cases}$$

(1) c の値を求めよ.

(2) Y の周辺確率密度関数 $f_Y(y)$ を求めよ.

(3) X と Y が独立であるか否かを, 理由とともに答えよ.

[4] ある町で収穫されたリンゴの重さの母平均 μ [g]を推定したい. 但し, 母標準偏差は50[g]であるとわかっている. また, 標準正規分布を $f(x)$ としたとき,

$$\int_{-1.96}^{1.96} f(x)dx = 0.95$$

とする. これについて, 以下の問いに答えよ.

(1) 無作為に抽出した100個の標本の重さの平均は350[g]であった. μ の95%の信頼区間を示せ.

(2) 95%の信頼区間の幅が7[g]以下になるように μ を推定するには, 何個以上の標本を抽出する必要があるか答えよ.

Translation of technical terms

じゃんけん rock paper scissors, 試行 trial, 独立 independence, 手 hand gesture, 等確率 equal probability, あいこ draw, 確率 probability, 正整数 positive integer, グー rock, チョキ scissors, パー paper, 確率変数 random variable, 期待値 expectation, 分散 variance, 区間 interval, 連続一様分布 continuous uniform distribution, 確率密度関数 probability density function, 同時確率密度関数 joint probability density function, 定数 constant, 周辺確率密度関数 marginal probability density function, 母平均 population mean, 推定 estimation, 母標準偏差 population standard deviation, 標準正規分布 standard normal distribution, 無作為に randomly, 抽出 sampling, 標本 sample, 信頼区間 confidence interval

プログラミング

2分木は各頂点が高々2つの頂点を子に持つ根付き木である．2分木の各頂点はデータを保持し，その子は左右で区別される．以下では各頂点がデータとして整数を保持する2分木を考える．図1に2分木の例を示す．

図 1: 2分木の例

C言語プログラムでは以下に定義される構造体により2分木を表現することができる．

```
struct vertex {
  int value;
  struct vertex *left;
  struct vertex *right;
};
```

この構造体は1つの頂点を表現し，その頂点が保持する整数を格納するメンバ value，左の子へのポインタを格納するメンバ left，右の子へのポインタを格納するメンバ right から構成される．この構造体で表現された頂点の左の子，右の子が存在しない場合はメンバ left，right にそれぞれ値 NULL を代入する．

ソースコード1は上述の構造体を利用して表現された2分木を取り扱うC言語プログラムである．大域変数 tree は2分木を参照するポインタ変数であり，宣言時の初期値を NULL とする．すなわち，頂点を持たない空の2分木を NULL で表す．insert 関数，eliminate 関数はポインタ tree が指す2分木に対して引数で与えられた整数を保持する頂点をそれぞれ挿入，削除する関数である．create 関数は引数で与えられた整数を保持する頂点を作成する関数である．display 関数は引数で与えられたポインタが指す2分木（もしくは2分部分木）に対して頂点が保持する整数を先行順で出力する関数であり，再帰的に定義されている．

ソースコード1について以下の問いに答えよ．

[1] main 関数を 18 行目の末尾まで実行した時点，ならびに 19 行目の末尾まで実行した時点におけるポインタ tree が指す2分木を図1の記法に従いそれぞれ図示せよ．

[2] 21 行目の末尾まで実行した時点において tree が指す2分木が図1 (b) に示される2分木の根を参照するように 21 行目にデータの挿入・削除の操作を記述したい．21 行目の［　（ア）　］に書くべき命令の列を1つ示せ．ただし，命令の列は以下の条件を満たすものとする．

- 命令は insert(x); または eliminate(y); のみである．ただし，x および y は整数である．
- insert 関数や eliminate 関数の引数に与えられる整数には同じものは高々1回しか現れない．

[3] 三項演算子 ? : は「式1?式2:式3」のように記述されて式を構成する.「式1」を評価した結果が0でないときは「式2」を，0であるときは「式3」を評価した結果を式の値とする．例えば，式 x <= 0 ? x+1 : x+2 を評価した値は x の値が −1 のときには 0 となり，5 のときは 7 となる．この三項演算子を用いて，53 行目から 56 行目のコード

```
if( x < p->value )
  p = p->left;
else
  p = p->right;
```

と同様の動作をする代入文を以下の［　(イ)　］を埋めて完成させよ．

```
p = [　　(イ)　　];
```

[4] ソースコード 1 の eliminate 関数の定義にはメモリへの不正アクセスを発生させる可能性がある．例えば，19 行目と次の行の間で eliminate(10); を実行した場合に不正アクセスが発生する．eliminate 関数の定義の 1 箇所に命令を追加してその問題を解決せよ．なお，解答は「○○行目と次の行の間に△△を挿入」という形式で記すこと．

[5] ソースコード 1 の eliminate 関数では不要になったメモリを解放していない．このままで挿入・削除の実行を繰り返した場合にどのような問題が起こりうるかを説明せよ．さらに，不要になったメモリを解放するように，eliminate 関数の定義の中の 2 箇所にそれぞれ命令を追加せよ．解答は「○○行目と次の行の間に△△を挿入」という形式で記せ．また，メモリの解放には標準ライブラリ関数である free 関数を使用すること．

[6] 21 行目の末尾まで実行した時点でポインタ tree が指す 2 分木は図 1 (b) になっている．このあと，22 行目を実行した際に出力される文字列を示せ．

[7] display 関数を実行した際に数が大きい順に出力されるようその定義を変更したい．以下の関数定義の中の［　(ウ)　］～［　(オ)　］を埋めて変更せよ．頂点に保持される整数を出力する場合には printf("%d,", p->value) を記述すること．

```
void display(struct vertex *p){
  if( p == NULL ) return;
  [　　(ウ)　　];
  [　　(エ)　　];
  [　　(オ)　　];
  return;
}
```

[8] 2 分木に指定した数が存在しているかどうかを探索する member 関数を以下の要件を満たすように作成したい．

- 引数として整数 x を受け取る．
- ポインタ tree が指す 2 分木に x を保持する頂点が存在する場合には 1 を，そうでない場合には 0 を返す．

これらの条件を満たすように以下の関数定義の中の (カ) ～ (ケ) を埋めてmember関数の定義を完成させよ.

```
int member(int x){
  struct vertex *p;
  p = tree;
  while( p != NULL ){
    if( (カ) ) return 1;
    if( (キ) )
      (ク) ;
    else
      (ケ) ;
  }
  return 0;
}
```

Translation of technical terms

2分木	binary tree	関数	function
頂点	vertex	部分木	subtree
子	child	先行順	preorder
根付き木	rooted tree	出力する	print out
C言語	C programming language	再帰的に	recursively
構造体	structure	操作	operation
メンバ	member	命令	statement
ソースコード	source code	列	sequence
プログラム	program	三項演算子	ternary operator
大域変数	global variable	式	expression
格納する	store	代入文	assignment statement
参照する	refer	メモリ	memory
ポインタ	pointer	不正アクセス	illegal access
変数	variable	発生させる	raise
宣言	declaration	実行する	execute
初期値	initial value	解放する	free
空	empty	標準ライブラリ関数	standard library function
引数	argument	文字列	string
挿入する	insert	関数定義	function definition
削除する	delete	探索する	search

ソースコード 1: 2分木を処理するC言語プログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct vertex {
5    int value;
6    struct vertex *left;
7    struct vertex *right;
8  };
9
10 struct vertex *tree = NULL;
11
12 struct vertex *create(int x);
13 void insert(int x);
14 void eliminate(int x);
15 void display(struct vertex *p);
16
17 int main(){
18   insert(3); insert(12); insert(18); insert(11); insert(14);
19   eliminate(12); insert(2); insert(3);
20   insert(6); eliminate(3); insert(5);
21   [          (ア)          ]
22   display(tree);
23   return 0;
24 }
25
26 struct vertex *create(int x){
27   struct vertex *p;
28   p = (struct vertex *) malloc(sizeof(struct vertex));
29   p->value = x;
30   p->left = NULL;
31   p->right = NULL;
32   return p;
33 }
34
35 void insert(int x){
36   struct vertex *p;
37
38   if( tree == NULL ){
39     tree = create(x);
40     return;
41   }
42
43   p = tree;
44   do{
45     if( p->value == x ) break;
46     else if( x < p->value && p->left == NULL ){
47       p->left = create(x);
48       break;
49     }else if( p->value < x && p->right == NULL ){
50       p->right = create(x);
51       break;
52     }
53     if( x < p->value )
54       p = p->left;
55     else
```

```
56        p = p->right;
57    }while(1);
58    return;
59  }
60
61  void eliminate(int x){
62    struct vertex *f, *p, *q;
63
64    p = tree;
65    if( p == NULL ) return;
66    do {
67      f = p;
68      if( x < p->value )
69        p = p->left;
70      else if( p->value < x )
71        p = p->right;
72    }while( x != p->value );
73
74    if( p->left == NULL || p->right == NULL ){
75      if( p->right == NULL )
76        q = p->left;
77      else
78        q = p->right;
79      if( p == tree )
80        tree = q;
81      else{
82        if( f->left == p )
83          f->left = q;
84        else
85          f->right = q;
86      }
87    }else{
88      q = p->right;
89      f = q;
90      while( q->left != NULL ){
91        f = q;
92        q = q->left;
93      }
94      p->value = q->value;
95      if( q == f )
96        p->right = q->right;
97      else
98        f->left = q->right;
99    }
100   return;
101 }
102
103 void display(struct vertex *p){
104   if( p == NULL ) return;
105   printf("%d,", p->value );
106   display(p->left);
107   display(p->right);
108   return;
109 }
```

計算機理論

[1] 記号列wの長さを$||w||$と表す．またwにおける記号aの出現回数を$||w||_a$と表す．例えば，$||aaba|| = 4$, $||aaba||_a = 3$, $||aaba||_b = 1$である．以下では，アルファベットを$\Sigma = \{a, b\}$とする．記号列$w_1, w_2 \in \Sigma^*$に対して，$||w_1||_a = ||w_2||_a$かつ$||w_1||_b = ||w_2||_b$が成り立つとき，$w_1 \sim w_2$とかく．例えば，$aaba \sim baaa$である．

言語$L_1, L_2 \subseteq \Sigma^*$が次の2つを満たすとき，記号等価であるという．

- すべての$w_1 \in L_1$について，ある$w_2 \in L_2$が存在して，$w_1 \sim w_2$である．
- すべての$w_2 \in L_2$について，ある$w_1 \in L_1$が存在して，$w_2 \sim w_1$である．

例えば，$\{ab, aaba\}$と$\{ab, ba, baaa\}$は記号等価である．

(1) $\{\varepsilon, ab, ba, baaa, abaa, aabb, baba\}$と記号等価な言語のうちで要素数が最小のものを1つ示せ．

(2) 以下の言語と記号等価な正規言語が存在する場合は，そのうちの1つを表す正規表現を示せ．存在しない場合は，その理由を述べよ．ただし，以下の言語がいずれも正規言語ではないことを使ってもよい．

(a) $\{a^i b^i : i \geq 0\}$

(b) $\{ww : w \in \Sigma^*\}$

(c) $\{a^{i^2} : i \geq 0\}$

(3) mを非負整数とする．$\{w \in \Sigma^* : ||w|| \leq m\}$と記号等価な言語のうちで要素数が最小のものを考える．その要素数を求めよ．求める過程も示せ．

(4) nを非負整数とする．$\{w \in \Sigma^* : ||w||_b = 1, ||w|| \leq n\}$と記号等価な言語のうち要素数が最小の言語が何通りあるかを求めよ．求める過程も示せ．

Translation of technical terms

記号列	string	記号等価	letter equivalent
長さ	length	正規言語	regular language
出現回数	number of occurrences	正規表現	regular expression
アルファベット	alphabet	非負整数	nonnegative integer

[2] 一階述語論理について問いに答えよ.

以下，演算子の優先順位を高い順に，$\forall, \exists, \neg, \wedge, \vee, \supset$ とする．必要に応じて，変数の記号として，$x, y, z, x', y', z', \cdots$，スコーレム定数として $a, b, \cdots$，スコーレム関数として，$f, g, \cdots$ を用いること.

(1) 次の一階述語論理式 F について問いに答えよ.

$$\neg\forall x((\exists y P(x,y) \supset \forall y Q(x,y)) \wedge \exists y \forall z P(z,y) \supset \exists y Q(x,y))$$

(a) 上記の論理式 F を冠頭連言標準形に変換せよ．ここで得られた式を P_F とする.

(b) P_F を節集合に変換せよ． この節集合を S_F とする.

(c) F が充足可能であるとき，S_F が充足可能であることを説明せよ.

(d) S_F が充足可能であるとき，F が充足可能であることを説明せよ.

(e) 節集合 S に対して導出原理を適用して得られる分解節 C を S に加えた節集合 $S \cup \{C\}$ が充足不能ならば，S は充足不能である．$(*)$
S_F に対して導出原理を適用していくと空節が導かれる．ここで，F が充足不能であることを $(*)$ を用いて説明せよ.

(2) 次の一階述語論理式 G について問いに答えよ.

$$\forall z(\exists x \forall y P(x,y,z) \supset Q(z)) \supset (\forall z \forall y \exists x P(x,y,z) \supset \exists z Q(z))$$

上記の論理式を偽とする $\{0,1\}$ を台にもつ P,Q に対する解釈を 1 つ示せ．解釈は P,Q に対する真理値表の形で示せ.

Translation of technical terms

一階述語論理	first order logic	導出原理	resolution principle
演算子	operator	分解節	resolvent
優先順位	priority	充足不能	unsatisfiable
スコーレム定数	Skolem constant	空節	empty clause
スコーレム関数	Skolem function	台	carrier
冠頭連言標準形	prenex conjunctive normal form	解釈	interpretation
節集合	set of clauses	真理値表	truth table
充足可能	satisfiable		

ハードウェア

[1] 多くのプロセッサは，プロセッサの内部に，高速で小容量の記憶装置であるレジスタを持っている。メインメモリと比較した場合，レジスタは演算器に近い場所に実装されていることや，メインメモリはDRAMで実装されていることが多いなどの理由から，レジスタは，演算器からのアクセス時間が短いという特徴がある。これに関して，次の問いに答えよ。

(1) メインメモリに対して直接演算を行う命令と，レジスタに対して演算を行う命令があった場合に，後者の方が高速である理由として，上で述べた「アクセス時間が短い」以外の理由を1つ挙げて説明せよ。

(2) レジスタの数の大小が，命令セットの設計に与える影響について考察せよ。

(3) アクセス時間と容量の面で，レジスタとメインメモリの中間に位置付けられる記憶装置の名称を答えよ。また，その記憶装置の有無が，命令セットの設計に与える影響について考察せよ。

[2] 2以上の整数M，N (M>N)，m，nに対し，M進数m桁符号なし整数からN進数n桁符号なし整数への変換を考える。

(1) すべてのM進数m桁符号なし整数を表現するために必要なnの最小値n^*をm, M, Nであらわせ。なお、実数xを超えない最大の整数を$\lfloor x \rfloor$，x以上の最小の整数を$\lceil x \rceil$で表すものとする。

これにより，与えられたM進数m桁符号なし整数Kは N進数1桁符号なし整数 k_i (i=0,..., n^*-1)を用い，
$K=\sum_{i=0}^{n^*-1}(k_i N^i)$と一意に表せる。

$(20)_3 \div (2)_3 = (10)_3$ あまり $(0)_2$ ・・・最下位ビット
$(10)_3 \div (2)_3 = (1)_3$ あまり $(1)_2$
$(1)_3 \div (2)_3 = (0)_3$ あまり $(1)_2$ ・・・最上位ビット
図1

図1に2桁の3進数$(20)_3$を3桁の2進数$(110)_2$に変換する例を示す。次々と$(2)_3$で除算することにより求めるアルゴリズムが知られている。

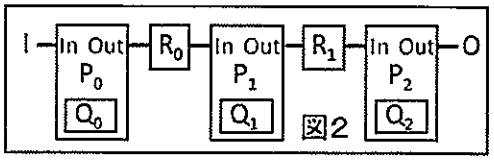

図2

このアルゴリズムを図2のようなパイプライン型演算器によって実現することを考える。図はn=3の例である。演算ブロック P_i (i=0,...,n-1)は同じ演算ブロック P を用いている。P はM進数1桁の入力In と出力Out，N進数1桁の状態Q を有する。P_iの状態を Q_iとする。P_0の入力はM進数1桁の外部入力Iと，P_{n-1}の出力はM進数1桁の外部出力Oと，それぞれ接続する。P_iと P_{i+1} (i=0,...,n-2)の間にはM進数1桁を保持するパイプラインレジスタ R_iを配置し，P_iの出力，P_{i+1}の入力と接続する。なお，時刻tにおけるI, O, Q_i, R_iの値をそれぞれ I_t, O_t, Q_{it}, R_{it}とする。

演算ブロックPは，各時刻 t>0 において入力Inと前状態 Q_{t-1}を用い，出力Outと次状態 Q_tを計算する。終了コードとよぶ特殊な値Eを定義し，Pは値Eが入力されると

値 E を出力し，状態を保持する（すなわち $Q_t = Q_{t-1}$）。

このパイプライン型演算器を次のように動作させる。まず時刻 0 において，すべての P_i に対し Q_i, R_i を $Q_{i0}=R_{i0}=0$ に初期化する。毎時刻 t においては，変換したい M 進数符号なし整数 K を上位の桁より 1 桁ずつ入力 I_t に与える。m 桁の入力が完了した後は終了コード E を入力し続ける。そして十分時間が経過したとき，$i<n^*$に対しては $Q_i=k_i$, $i \geqq n^*$に対しては $Q_i=0$ が保持されるようにする。例えば図 1 の例の場合，$(20)_3$ を $I_1=(2)_3$, $I_2=(0)_3$, $I_t=E$ $(t>2)$のように入力し，十分時間が経過したとき，$Q_{0\infty}=(0)_2$, $Q_{1\infty}=(1)_2$, $Q_{2\infty}=(1)_2$, $Q_{i\infty}=(0)_2$ $(i>2)$となるようにする。

(2) $(00)_3$ から$(21)_3$ までの入力を想定し，M=3, m=2, N=2, n=3 に対する上記パイプライン型演算器を設計した。これに $(20)_3$ を入力したところ，時刻 0 から 5 における I_t, O_t, R_{it}, Q_{it} の値が表 1 のようになった。このとき表 1 の①から⑮を回答せよ。

表 1

t	I_t	Q_{0t}	R_{0t}	Q_{1t}	R_{1t}	Q_{2t}	O_t
0		$(0)_2$	$(0)_3$	$(0)_2$	$(0)_3$	$(0)_2$	
1	$(2)_3$	①	②	③	④	⑤	⑥
2	$(0)_3$	$(0)_2$	⑦	⑧	⑨	⑩	⑪
3	E	$(0)_2$	E	$(1)_2$	⑫	⑬	⑭
4	E	$(0)_2$	E	$(1)_2$	E	$(1)_2$	⑮
5	E	$(0)_2$	E	$(1)_2$	E	$(1)_2$	E

(3) 演算ブロック P について，出力 Out と次状態 Q_t を，入力 In，前状態 Q_{t-1}, M, N を用いた式であらわせ。なお，In=E の場合について示す必要はない。

(4) 出力 O に終了コード E が現れる時刻 t*を m, n, M, N を用いて答えよ。

次に，上述のアルゴリズムを用いて $(00)_3$ から$(21)_3$ までの範囲の 3 進数を 3 ビットの 2 進数に変換するため，図 3 のような回路L を考える。図 3 において D-FF は D フリップフロップを表し，D は入力，Q は前クロックでの D の値を出力する。見やすさのためクロック回路および初期化回路は省略している。組合せ回路 X は 3 本の 1 ビット入力 i1, i0, ib，3 本の 1 ビット出力 o1, o0, ob をもつ。組合せ回路 Y は 2 本の 1 ビット入力 i1, i0，1 本の 1 ビット出力 ne をもつ。回路 L は，2 本の 1 ビット外部入力 T1, T0, 4 本の 1 ビット外部出力 B2, B1, B0, NFIN をもつ。

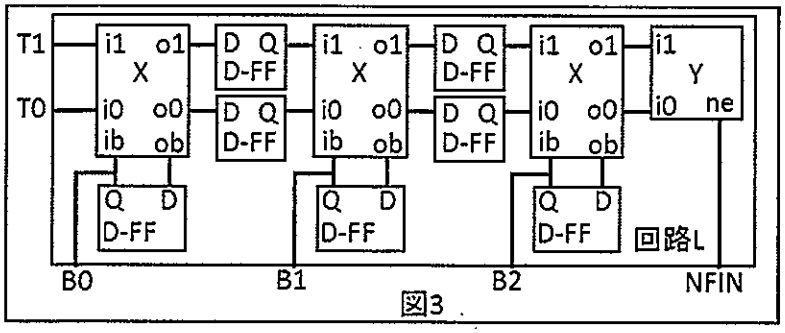

図3

表 2

入力	T1	T0
0	0	0
1	0	1
2	1	0
E	1	1

表 3

時刻	T1	T0
1	1	0
2	0	0
>2	1	1

このとき回路 L を次のように動作させる。

- 時刻 0 において，すべての D-FF を 0 に初期化する。このとき B2=B1=B0=0，NFIN=1 を出力する。
- 変換したい 3 進数を入力 T1, T0 に与える。毎クロック，上位の桁より 1 桁ずつ与

え，コード化は表 2 のようにする。入力が終了した後は終了コード E (T1=T0=1) を与え続ける。例えば$(20)_3$を与える時には表 3 のように入力する。

- 出力 NFIN が 0 に変わった時，B2, B1, B0 の出力が変換後の 2 進数を表すようにする。このとき，B2 が最上位ビット，B0 が最下位ビットとなるようにする。例えば入力$(20)_3$に対して B2=B1=1，B0=0 となる。

(5) 回路 L が上記の動作をするよう組合せ回路 X，Y の真理値表，カルノー図を示せ。その上でカルノー図を用いた簡単化を行い，それに対応する論理ゲート回路を示せ。論理ゲートは NOT と 2 入力と 3 入力の NAND のみを使用せよ。なお，クロック回路や D-FF 初期化回路について考慮する必要はない。

Translation of technical terms

プロセッサ	processor	入力	input
記憶装置	memory	出力	output
レジスタ	register	状態	state
メインメモリ	main memory	外部入力	primary input
演算器	arithmetic unit	外部出力	primary output
アクセス時間	access time	初期化	initialization
命令	instruction	回路	circuit
整数	integer	組合せ回路	combinational circuit
(M)進数	(M-)ary	最上位ビット	most significant bit
桁	digit	最下位ビット	least significant bit
符号なし	unsigned	真理値表	truth table
最小	minimum	カルノー図	Karnaugh map
実数	real number	簡単化	simplification
最大	maximum	論理ゲート	logic gate
除算	division		

ソフトウェア

[1] DFD(Data Flow Diagram)に関する下の問いに答えよ.

(1) 図 1 の DFD の説明として正しいものをすべて選べ.

(a) 処理 C は処理 B よりも先に実行される.

(b) 処理 A の入力データはない.

(c) 処理 B は処理 A からのデータを入力とする.

(d) 処理 C が完了しないと処理 D は開始できない.

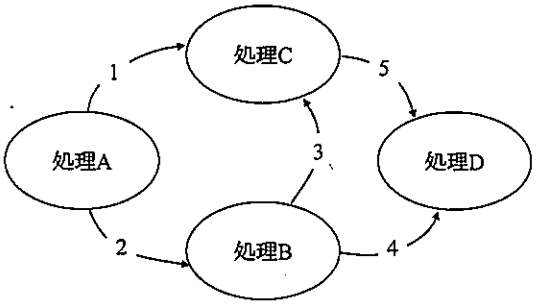

図 1 DFD

(2) 図 1 の DFD を構造図に変換せよ. 構造図は処理 C を中央（根）としデータフローを記述すること.

[2] UML(Unified Modeling Language)に関する下の問いに答えよ.

(1) 図 2 のクラス図C を汎化を用いないクラス図 C'に書き直せ. C と C'でクラスの属性名と操作名は同一となるようにせよ.

(2) クラス図 C をオブジェクト指向プログラミング言語で実装したプログラムのソースコードを S とする. 同様に, C'を実装したソースコードを S'とする.

クラス Element とクラス KeyValueElement に操作previous()を追加するときを考える。ソースコード S と S'の変更手順をそれぞれ 200 文字以内で示せ。なお、オブジェクト指向プログラミング言語は継承機能を持つものとする。また、クラス図 C と C'は変更しなくてよい。

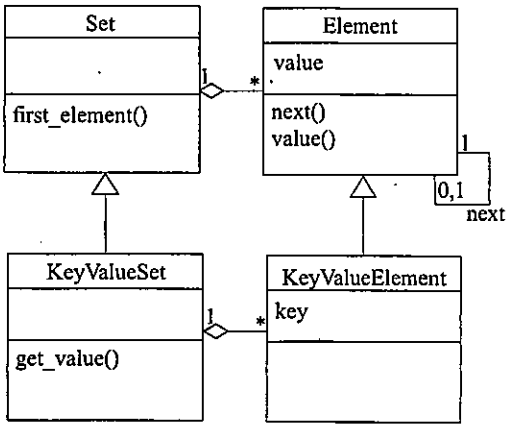

図 2 クラス図 C

Translation of technical terms

データ	data	属性名	attribute name
構造図	structure chart	操作名	operation name
中央	center	オブジェクト指向プログラミング言語	object-oriented programming language
根	root	実装	implementation
データフロー	data flow	プログラム	program
クラス図	class diagram	ソースコード	source code
汎化	generalization	操作	operator
クラス	class	継承機能	inheritance feature

[3] マルチプロセッサ上で実行される並行プロセスの排他制御について，以下の問いに答えよ．

(1) マルチプロセッサ上で複数のプロセスが以下のC言語で書かれた関数func1を同時に実行する場合において，変数shared_value を排他的にアクセスしたい．

```
int lock = 0;
int shared_value = 0;
void func1(void) {
   while (TestAndSet(&lock)) ;
   shared_value++;
   lock = 0;
}
```

この排他制御を実現するために必要な関数 TestAndSet を以下の空白(A)から(E)を埋めて完成させよ．なお，関数 TestAndSet はハードウェアによって不可分に実行されるものとし，変数 lock には 0 か 1 の値を設定する．

```
int TestAndSet(int *a) {
   int b;
   [ (A) ] = [ (B) ];
   [ (C) ] = [ (D) ];
   return [ (E) ];
}
```

(2) 下記の関数 Swap はハードウェアによって不可分に実行されるものとする．

```
void Swap(int *a, int *b) {
   int x = *a;
   *a = *b;
   *b = x;
}
```

この関数 Swap を用いて，(1)と同様に複数のプロセスが変数 shared_value へ排他的にアクセスするコードを以下の空白(F)～(I)を埋めて完成させよ．

```
int lock = 0;
int shared_value = 0;
void func2(void) {
   int key = [ (F) ];
   while ( [ (G) ] == 1)
      Swap( [ (H) ] , [ (I) ] );
   shared_value++;
   lock = 0;
}
```

Translation of technical terms

マルチプロセッサ	multiprocessor	関数	function
並行プロセス	concurrent process	変数	variable
排他制御	mutual exclusion	ハードウェア	hardware
C 言語	C language	不可分	atomic