平 成 ２４ 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２３年８月９日（火）
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、和英辞書などの辞書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．解答用紙は試験終了後に３枚とも提出せよ。

８．問題冊子、草稿用紙は試験終了後に持ち帰ってよい。

解析・線形代数

（解の導出過程も書くこと）

[1] 次の複素数zに関する方程式について、以下の問いに答えよ。

$$z^3 = -2 + 2\sqrt{3}i \tag{1}$$

(a) $-2+2\sqrt{3}i$ を極形式で表せ。

(b) (1) の解をすべて求めよ。

(c) (b) で得られたすべての解を複素平面上に図示せよ。

[2] 次の行列Aについて、以下の問いに答えよ。

$$A = \begin{pmatrix} a & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -a \end{pmatrix}$$

(a) 行列 A のすべての固有値を a を使って記せ。

(b) 行列 A が対角化可能でないような実数a をすべて求めよ。

[3] xy平面上の円$x^2+y^2=4$ を z軸方向に動かしてできる円柱の $0 \leq z \leq x$ の部分の立体Tについて、以下の問いに答えよ。

(a) 立体 T の概形を図示せよ。

(b) 立体 T の体積を求めよ。

Translation of technical terms

複素数	complex number	実数	real number
方程式	equation	平面	plane
極形式	polar form	円	circle
解	solution	軸	axis
複素平面	complex plane	円柱	cylinder
行列	matrix	立体	solid
固有値	eigenvalue	概形	rough sketch
対角化可能	diagonalizable	体積	volume

確率・統計（解の導出過程も書くこと。）

［1］連続確率変数 X の累積分布関数は $F_X(x) = P\{X \le x\}$ で与えられる。区間 [0,1] で定義された、二つの独立な確率変数 X_1, X_2 の累積分布関数 $F_{X_1}(x)$, $F_{X_2}(x)$ が図1で与えられる時、以下の問いに答えよ。

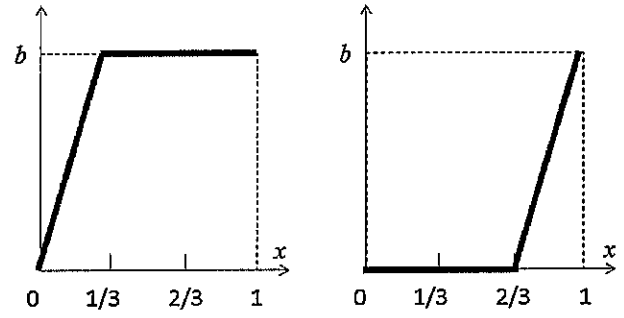

図1： 累積分布関数 $F_{X_1}(x)$（左） $F_{X_2}(x)$（右）

（1）図1中の b の値を定めよ。

（2）X_1 ならびに X_2 の確率密度関数 $f_1(x), f_2(x)$ をそれぞれ求め図示せよ。

（3）X_1 と X_2 それぞれの平均と分散を求めよ。

（4）X_1 と X_2 の和で与えられる確率変数 $Y = X_1 + X_2$ の平均と分散を求めよ。

（5）（4）で与えられる Y の累積分布関数 $F_Y(y)$ は X_1, X_2 の結合密度関数 $f_{12}(x_1, x_2)$ を用いて

$$F_Y(y) = \int_{-\infty}^{\infty} \int_{-\infty}^{y-x_1} f_{12}(x_1, x_2)\, dx_2 dx_1$$

で与えられる。このことを利用して Y の確率密度関数 $f_Y(y)$ を求め図示せよ。

［2］AとB二つの工場で同一の製品を製造しており、それぞれの工場で不良品が発生する確率はA工場で 0.01、B工場では 0.05 である。またそれぞれの工場で製造される製品の数の比は、$x : (1-x)$ である。（ただし $0 < x < 1$ であり、A工場を x、B工場を $1-x$ とする。）二つの工場で製造された製品を集め、無作為に合計 100 個の製品を抜き出して検査したところ不良品が 4 個だけ見つかった。以下の問いに答えよ。

（1）製品が不良品である確率 p を、x を用いて表せ。

（2）製品 100 個中に不良品が 4 個だけ見つかる確率 q を、p を用いて表せ。

（3）q の最大値を与える p を求めよ。

（4）（3）の結果から、二つの工場で製造された製品数の比 $x : (1-x)$ を推定せよ。

累積分布関数　cumulative distribution function
確率密度関数　probability density function
確率変数　random variable
平均　mean
分散　variance
結合密度　joint probability density
不良品　defective product
無作為に　randomly
検査　inspection
最大値　maximum value
比　ratio
推定　estimate

プログラミング

有向グラフ $G = (V, E)$ は，頂点の集合 V と有向辺の集合 $E \subseteq V \times V$ から構成される．V の有向辺 $(v_i, v_j) \in E$ を e とするとき，e の始点 v_i を $\mathsf{src}(e)$，終点 v_j を $\mathsf{dst}(e)$ と書き，e は v_j の入力辺であるという．ここで，G は孤立点を持たない．すなわち，すべての頂点 v に対して，$v = \mathsf{dst}(e)$ または $v = \mathsf{src}(e)$ となる e が存在する．

アルゴリズム A は有向グラフ $G = (V, E)$ を入力とし，V の要素からなるリスト L を出力する．

プログラム P は，アルゴリズム A を図 1 の有向グラフ G1 に対して適用した C 言語プログラムである．(プログラム P において，各行の先頭の数字は行番号を示し，プログラムには含まれない．)

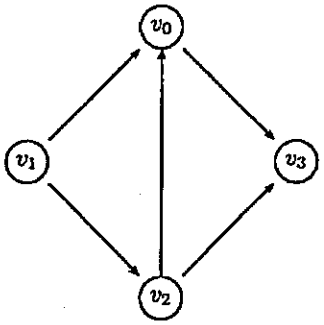

図 1: 有向グラフ G1

```
K ← 有向辺の集合
L ← 空リスト
S ← 入力辺を持たない頂点の集合
while S が空でない do
  S から頂点 n を選択し削除する
  L の先頭に n を追加する
  foreach src(e) = n となる辺 e do
    K から e を削除する
    m ← dst(e)
    if m のどの入力辺も K に含まれない then
      m を S に追加する
    fi
  od
od
L の要素を先頭から順に出力
```

アルゴリズム A

(1) アルゴリズム A における集合 S は，プログラム P では配列 workset で実現されている．S に v_i が属することは，workset ではどのように表現されているか説明し，プログラム P の (a) に適当な代入文を書け．

(2) 配列 edges は，2次元配列であり，v_x から v_y に有向辺が存在するときには edges[x][y] の値を 1 とし，存在しないときには 0 とする．プログラム P の (b-1),(b-2) に配列 edges に対する条件を書け．

(3) プログラム P の実行出力を書け．

(4) つぎに 11 行目の edges の値を変更してアルゴリズム A を図 2 のグラフ G2 に適用する C 言語プログラムを構成する．

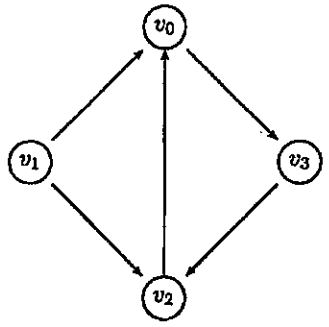

図 2: 有向グラフ G2

変更後の 11 行目において以下の (c) を埋めよ．

```
int edges[N][N]= (c) ;
```

さらに 11 行目を上記のように変更したプログラムの実行出力を書け．

(5) プログラム P の N と edges の値を変更して，さまざまなグラフ $G = (V, E)$ に対して適用したプログラムを実行したとき，V の要素で出力されない頂点があるのは，適用するグラフ G がどのような性質をもつ場合か．

(6) (5) の場合に，59 行目の直前で配列 edges はどのような性質を持つか，簡潔に記述せよ．

(7) プログラム P の N と edges の値を変更して，グラフ $G = (V, E)$ を適用したとき，V の要素がすべて出力される場合の計算量(けいさんりょう)を V の要素数 N で示し，その理由を説明せよ．

プログラムP

```
 1: #include <stdlib.h>
 2: #include <stdio.h>
 3:
 4: #define N 4
 5:
 6: typedef struct node {
 7:   int data;
 8:   struct node *next;
 9: } node;
10:
11: int edges[N][N] = {{0,0,0,1},{1,0,1,0},{1,0,0,1},{0,0,0,0}};
12: int workset[N];
13: node *sorted = NULL;
14:
15: node *cons(int a, node *list) {
16:    node *n = (node *)malloc(sizeof(node));
17:    n->data = a;
18:    n->next = list;
19:    return n;
20: }
21:
22: int isroot(int x) {
23:   int y;
24:   for (y = 0; y < N; y++) {
25:     if ( (b-1) ) return 0;
26:   }
27:   return 1;
28: }
29:
30: int pick() {
31:   int x;
32:   for (x=0; x < N; x++) {
33:     if (workset[x]) {
34:       (a) ;
35:       return x;
36:     }
37:   }
```

```
38:     return -1;
39:   }
40:
41:   void main(void) {
42:     int a,x;
43:     for(x = 0; x < N; x++) {
44:       if(isroot(x)) {
45:         workset[x]=1;
46:       } else {
47:         workset[x]=0;
48:       }
49:     }
50:     while ((a=pick()) != -1) {
51:       sorted = cons(a,sorted);
52:       for (x=0;x < N;x++) {
53:         if ( (b-2) ) {
54:           edges[a][x] = 0;
55:           if (isroot(x)) workset[x] = 1;
56:         }
57:       }
58:     }
59:     while (sorted != NULL) {
60:       printf("%d\n",sorted->data);
61:       sorted = sorted->next;
62:     }
63:   }
```

専門用語訳:

有向(ゆうこう)グラフ	directed graph	頂点(ちょうてん)	vertex	孤立点(こりつてん)	degree-0 vertex
有向辺(ゆうこうへん)	directed edge	配列(はいれつ)	array	代入文(だいにゅうぶん)	assignment statement
計算量(けいさんりょう)	complexity				

計算機理論

P_i, Q_i $(i = 1, 2, \ldots)$ を命題記号(proposition symbol) とする．以下で扱う論理式(formula) において，結合子 (connective) $\neg$, $\wedge$, $\vee$, $\rightarrow$ は，それぞれ，否定(negation)，連言 (conjunction)，選言(disjunction)，含意 (implication) である．結合の強さはこの順であり，$\wedge, \vee, \rightarrow$ はそれぞれ右結合(right associative) であるとして論理式の括弧を省略できるものとする．

(1) 論理式 [1], [2] はそれぞれ充足可能(satisfiable) か否かを答えよ．充足可能である場合は，その論理式を真にする解釈(interpretation)（命題記号に真，偽を割り当てる関数）も示せ．

$$P_1 \wedge \neg P_3 \wedge (P_1 \rightarrow P_2) \wedge (P_2 \rightarrow P_3) \qquad [1]$$

$$((P_3 \wedge \neg P_1) \vee (P_3 \wedge \neg P_4)) \wedge (P_4 \rightarrow P_3) \wedge (P_3 \rightarrow P_2) \wedge (P_2 \rightarrow P_1) \qquad [2]$$

(2) L_i, L_i' $(i = 1, 2, \ldots)$ は $P_j (j = 1, 2, \ldots)$ から構成されるリテラル (literal) とする．$k = 1, 2, \ldots$ に対して論理式 $\Phi(k)$, $\Psi(k)$ を [3], [4] で定める．

$$\Phi(k) = \bigvee_{1 \leq i \leq k} (L_i \wedge L_i') \qquad [3]$$

$$\Psi(k) = \left(\bigvee_{1 \leq i \leq k} Q_i \right) \wedge \bigwedge_{1 \leq i \leq k} (Q_i \rightarrow L_i \wedge L_i') \qquad [4]$$

以下の問い (a)〜(e) に答えよ．

(a) $\Phi(3)$ と $\Psi(3)$ の連言標準形 (conjunctive normal form, CNF) をそれぞれ求めよ．

(b) I は $\Phi(k)$ を真とする解釈であるとする．$\Psi(k)$ を真とする解釈の一つ I' を I を用いて表せ．

(c) $\Psi(k)$ を真とする解釈は $\Phi(k)$ も真とすることを示せ．

(d) $\Phi(k)$ の CNF に含まれる節 (clause) の個数を k で表せ．また，$\Phi(10)$ の CNF に含まれる節の一つを書け．

(e) $\Psi(k)$ の CNF に含まれる節の個数を k で表せ．

(3) 有向グラフ (directed graph) $G = (V, E)$ について考える．ここに，V は頂点 (vertex) の集合であり，$V = \{v_1, \ldots, v_{2n}\}$ $(n > 0)$ とする．E は辺(edge) の集合であり，$E \subseteq V \times V$ である．以下の問い (a)〜(d) に答えよ．

(a) G において v_2 が v_1 から辺をたどって到達可能あることと，論理式 [8] が充足不能 (unsatisfiable) であることとが必要十分条件となるように φ_1 を与えよ．

$$\boxed{\varphi_1} \wedge \bigwedge_{(v_i, v_j) \in E} (P_i \rightarrow P_j) \qquad [5]$$

(b) 論理式 [6] が充足不能であることと，G においてすべての k $(1 \leq k \leq n)$ に対して v_{2k} は v_{2k-1} から到達可能であることとが互いに必要十分条件になるように φ_2 を与えよ．

$$\left(\boxed{\varphi_2} \right) \wedge \bigwedge_{(v_i, v_j) \in E} (P_i \rightarrow P_j) \qquad [6]$$

(c) 論理式 [7] の φ_3 は，問い (2) の $\Phi(k)$ から $\Psi(k)$ への変形を論理式 [6] の φ_2 に適用して得られるものである．φ_3 を与えよ．

$$\left(\boxed{\varphi_3} \right) \wedge \bigwedge_{(v_i,v_j)\in E} (P_i \to P_j) \qquad [7]$$

(d) 論理式 [7] の CNF に含まれる節の個数を n と辺の個数 m で表せ．

以上

ハードウェア

[1] 同期式順序回路 (synchronous sequential circuit) により構成した3ビット乗算器 (multiplier) のデータパス (data path) を図1に，制御回路 (control circuit) の動作を表す有限状態機械 (finite state machine) を図2に示す。

図1において，積レジスタ (product register) は，次の2つの機能を持つ。

- 右シフト：1ビット右にシフト (shift) し，最上位ビット (most significant bit) に0を書き込む
- 書き込み：加算器 (adder) の出力を上位4ビットに書き込む

回路を動作させる前に，乗数 (multiplier) を積レジスタの下位3ビットに，被乗数 (multiplicand) を被乗数レジスタ (multiplicand register) に格納する。積レジスタの上位3ビットは0に初期化する。

図2において，初期状態 (initial state) はA，終了状態 (final state) はGである。「／」の前は状態遷移 (state transition) の条件 (condition) を，「／」の後はその時の動作 (action) を示す。条件が「—」は，無条件に状態遷移することを示す。また，積レジスタの最下位ビット (least significant bit) を，「最下位ビット」と表記している。なお，クロック (clock) 毎に1つの状態遷移を行うものとする。

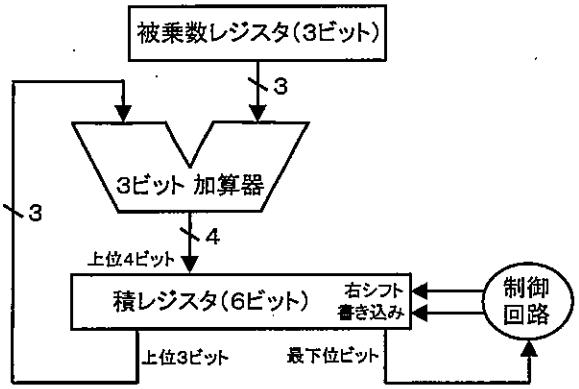

図1. 乗算器のデータパス

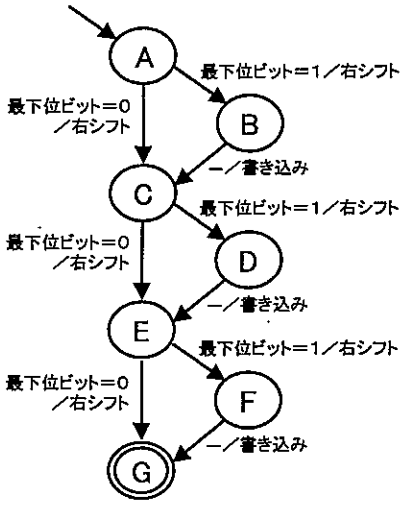

図2. 制御回路の動作を表す有限状態機械

この乗算器に関する以下の問いに答えよ。

(1) この乗算器が，初期状態から開始して，終了状態に到達するまでに必要なクロック数の，全入力パターンに対する平均値を求めよ。

(2) 図2の各状態に以下のように状態コード (state code) を割り当てた場合の出力関数 (output function) と次状態関数 (next state function) を，真理値表 (truth ta-

ble）の形で記述せよ。なお，乗算器が終了状態に到達した後のことは考慮しなくてよいものとし，ドントケア（don't care）についても記述すること。

状態	状態コード
A	0 0 0
B	0 0 1
C	0 1 0
D	0 1 1
E	1 0 0
F	1 0 1
G	1 1 0

（3）出力関数と次状態関数を簡単化（simplify）し，制御回路を設計して，その回路図（logic diagram）を描け。ただし，初期化と終了判定の機能を持たせる必要はない。

［2］マイクロプロセッサ（microprocessors）のキャッシュ・アーキテクチャ（cache architecture）について以下の問いに答えよ。

（1）次の用語をそれぞれ100文字程度で説明せよ。

(a) ダイレクトマップ・キャッシュ（direct mapped cache）

(b) セットアソシアティブ・キャッシュ（set associative cache）

(c) フルアソシアティブ・キャッシュ（fully associative cache）

（2）1ブロックが16バイトで，サイズが2Kバイトのキャッシュを考える。ダイレクトマップ・キャッシュ，2-wayセットアソシアティブ・キャッシュ，4-wayセットアソシアティブ・キャッシュに対し，アドレス中におけるタグ部（tag part）のビット数をそれぞれ求めよ。ただし，アドレスは32ビットで，バイトアドレス（byte addressing）を採用しているものとする。

（3）（2）の3種類のキャッシュにおいて，以下の(a)，(b)それぞれのケースとなるデータアクセスパターン（data access pattern）を，アドレスリストの具体例を示して述べよ。また，そのようになる理由を説明せよ。なお，キャッシュのリプレースメント方式（replacement method）はLRU（Least Recently Used）方式を採用しているものとする。

(a) キャッシュヒット率（cache hit ratio）について，4-wayセットアソシアティブ・キャッシュが最も高くなり，ダイレクトマップ・キャッシュが最も低くなるようなデータアクセスパターン

(b) キャッシュヒット率について，4-wayセットアソシアティブ・キャッシュと2-wayセットアソシアティブ・キャッシュが同じになり，ダイレクトマップ・キャッシュが最も低くなるようなデータアクセスパターン

ソフトウェア

インターネットでデータ交換する際に Base64 と呼ばれるエンコーディングが使われることがある．あるデータを Base64 形式に変換する手順は以下の 2 つからなる．

1. 元データを6bitずつに分割する．6bitに満たない分は0を追加して6bitにする．
2. 各6bitの値(0～63)を，別途定める変換表を用いて文字に変換する．4文字単位で出力し，4文字に満たない分は = 記号を追加して4文字にする．

このための関数encode()をリスト1とリスト2の2種類の方法で書いた．リスト2はリスト2-1，リスト2-2，リスト2-3の3つのファイルからなる．リスト2では，記憶クラス指定子staticを用いて静的局所変数を定義している．静的局所変数はその関数を抜けても変数領域は解放されず，値が保持される． これらのプログラムについて以下の問に答えよ．

(1) モジュール設計の尺度にモジュールの結合度がある．結合度には，外部結合，共通結合，スタンプ結合，制御結合，データ結合，内容結合の6つのレベルがある．これらを良いとされる順に左から並べ(a)～(e)を埋めよ．

[(a)] > [(b)] > [(c)] > [(d)] > [(e)] > 内容結合

(2) スタンプ結合とはどのような結合度であるか，それはどのような場合に起こりやすいかを説明せよ．

(3) このリスト2のencode()とmain()の結合度は上記の６つのレベルのどれにあたるか，その定義とともに説明せよ．

(4) リスト1とリスト2のプログラムを再利用性の観点で比較し，設計の良し悪しを論じよ．

(5) サイクロマティック複雑度はプログラムの複雑度の尺度であり，エッジが交差しない制御フローグラフの閉じた領域の個数に 1 を加えることで得られる．リスト2のすべての関数についてそれぞれサイクロマティック複雑度を求めよ．

(6) サイクロマティック複雑度とプログラムテストの関係を論じよ．

用語

エンコーディング：encoding
変換表(へんかんひょう)：translation table
記憶(きおく)クラス指定子(していし)：storage classes specifier
静的局所変数(せいてききょくしょへんすう)：static local variables
変数領域(へんすうりょういき)：space for the variable
モジュール設計(せっけい)：module design
外部結合(がいぶけつごう)：external coupling
共通結合(きょうつうけつごう)：common coupling
スタンプ結合(けつごう)： stamp coupling
制御結合(せいぎょけつごう)：control coupling
データ結合(けつごう)：data coupling
内容結合(ないようけつごう)：content coupling
サイクロマティック複雑度(ふくざつど)：cyclomatic complexity
制御(せいぎょ)フローグラフ：control flow graph
プログラムテスト：program testing

リスト1

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

#define ETX '\003'     /* End of Text */

const unsigned char *base64 = (char *)
   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
unsigned char *str;
int srcsize, pos, out;

unsigned char encode( ){
    pos++;
    switch( out++ % 4){
    case 0:
      if( pos-- > srcsize ){
        return ETX;
      }
      return base64[(str[pos]>>2) & 0x3f];
    case 1:
      if( pos == srcsize ) {
        return base64[((str[pos-1]<<4)|0) & 0x3f ];
      } else {
        return base64[((str[pos-1]<<4)|(str[pos]>>4)) & 0x3f ];
      }
    case 2:
      if( pos < srcsize ) {
        return base64[((str[pos-1]<<2)|(str[pos]>>6)) & 0x3f ];
      } else if ( pos == srcsize ){
        return base64[(str[pos-1]<<2) & 0x3f];
      } else {
        return '=';
      }
    case 3:
      if ( pos <= srcsize ) {
        return base64[ str[pos-1] & 0x3f ];
      } else {
        return '=';
      }
    }
}

main( argc, argv ){
  unsigned char c;
  unsigned char *src = "Hello";
  str = src;                /* set parameter */
  srcsize = strlen( src );
  pos = out = 0;
  printf("input : %s\n", src);
  printf("output: ");
  while((c=encode()) != ETX ){
    putchar(c);
  }
  putchar('\n');
}
```

リスト 2-1 (ファイル名:encode.c)

```
#include "encode.h"

unsigned char encode( int size, unsigned char *source ){
  static char *base64 = (char *)
   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
  static unsigned char *str;
  static int srcsize, pos, out;
  if( size > 0 ){
    srcsize = size;
    str = source;
    pos = out = 0;
    return 0;
  } else {
    pos++;
    switch( out++ % 4){
    case 0:
     if( pos-- > srcsize ){
        return ETX;
      }
     return base64[(str[pos]>>2) & 0x3f];
    case 1:
      if( pos == srcsize ) {
        return base64[((str[pos-1]<<4)|0) & 0x3f ];
      } else {
        return base64[((str[pos-1]<<4)|(str[pos]>>4)) & 0x3f ];
      }
    case 2:
      if( pos < srcsize ) {
        return base64[((str[pos-1]<<2)|(str[pos]>>6)) & 0x3f ];
      } else if ( pos == srcsize ){
        return base64[(str[pos-1]<<2) & 0x3f];
      } else {
        return '=';
      }
    case 3:
      if ( pos <= srcsize ) {
        return base64[ str[pos-1] & 0x3f ];
      } else {
        return '=';
      }
    }
  }
}
```

リスト 2-2 (ファイル名:encode.h)

```
#define ETX '\003'          /* End of Text */
unsigned char encode( int size, unsigned char *source );
```

リスト 2-3 (ファイル名:main.c)

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

main( argc, argv ){
  unsigned char c;
  unsigned char *src = "Hello";
  encode( strlen(src), src );
  printf("input : %s\n", src);
  printf("output: ");
  while((c=encode(-1, "")) != ETX ){
    putchar(c);
  }
  putchar('\n');
}
```