

大阪大学大学院情報科学研究科 博士前期課程

コンピュータサイエンス専攻
情報システム工学専攻
情報ネットワーク学専攻
マルチメディア工学専攻
バイオ情報工学専攻

【情報工学】

参考問題

【必須問題】アルゴリズムとプログラミング

(情報工学 1/15)

配点：(1) 32 点，(2) 30 点，(3) 20 点，(4-1) 20 点，(4-2) 23 点

図 1 に示す ANSI-C 準拠である C 言語プログラムは、以下に挙げる 4 個の制約を満たす重み付き無向グラフ (weighted undirected graph) G における 2 頂点 (vertex) 間の最短経路長 (shortest path length) を出力するプログラムである。

- ・頂点数を n として、頂点を $0 \sim n - 1$ の非負整数 (non-negative integer) で識別できること。
- ・すべての辺 (edge) の重みが非負整数であること。
- ・グラフ G は連結 (connected) であること。すなわち、任意の 2 頂点間に経路が存在する。
- ・無限大 (infinity) とみなす値を `INF` として、任意の 2 頂点間の最短経路長が `INF` 未満であること。

ここで、始点 (source) s から終点 (destination) d への最短経路とは、 s から d への経路上の辺の重みの和 (sum) が最小 (minimum) となる経路であり、 s と d が同一頂点である場合、その最短経路長は 0 である。本プログラムは、グラフ G の頂点数 n を変数 (variable) `n` に格納し、頂点 i から頂点 j への辺の重み $w_{i,j}$ を、配列 (array) `w` の要素 (element) `w[i*n+j]` に格納している。なお、存在しない辺に対しては重みの値を `INF` とし、任意の頂点 i に対して $w_{i,i} = 0$ とする。図 2 に、図 1 で扱うグラフを示す。以下の各問に答えよ。

- (1) 22～34 行目の `while` 文の処理において 1 巡目および 2 巡目の終了時における `Len[0] ~ Len[3]` の値を、解答用紙の太線内の空欄を埋めることにより答えよ。
- (2) 関数 `compute` が実現しているアルゴリズムの時間計算量 (time complexity) のオーダ表記 (order notation) を、頂点数 n を用いて答えよ。その理由も簡潔に説明せよ。
- (3) 最短経路長だけでなく、最短経路の一つを出力するように図 1 のプログラムを変更したい。例えば、始点 0 から終点 3 への最短経路であれば、以下のように出力したい。

```
Shortest Path Length from 0 to 3 is 3
- 0 - 1 - 3
```

そこで、5, 17, 27 および 47 行目の注釈を解除 (uncomment) し、下記に示す関数 `printpath` の定義を 12 行目の位置に挿入した。関数 `printpath` を呼び出して上記のような出力を得るためにには、
（ア）および（イ）をどのように記述すればよいか、適切な文をそれぞれ答えよ。

```
void printpath(int v) {
    if (v != -1) { printpath(Prev[v]); printf("- %d ", v); }
}
```

- (4) 図 1 の `main` 関数を変更することにより、頂点数を $n = 4$ に固定したときの一般のグラフにおけるすべての頂点対間 (all-pairs of vertices) の最短経路長を出力したい。その実現のために、44 行目以降に以下の変更のみを許す。

- ・関数 `compute` の呼び出しを追加・削除し、その実引数 (actual argument) を変更してよい。
 - ・関数 `printf` の呼び出しを追加・削除し、文字列や配列 `Len` の要素に限り、それらを出力してよい。
- なお、処理対象のグラフに合わせて 38～41 行目の値は変更されているものとする。また、最短経路は出力しなくてよい。以下の各小間に答えよ。

- (4-1) すべての頂点対間の最短経路長を出力するためには、関数 `compute` をどのように呼び出せばよいか、簡潔に説明せよ。呼び出し回数 T の値に言及し、できるだけ T の少ない方法を示すこと。
- (4-2) 22 行目を以下のように変更した。変更後の関数 `compute` をどのように呼び出せば、すべての頂点対間の最短経路長を出力できるか、簡潔に説明せよ。呼び出し回数 T の値に言及し、できるだけ T の少ない方法を示すこと。

```
while (allvisited(visited, n)==0) {
```

```
→ while (visited[d]==0) {
```

変更前

変更後

```

1 #include <stdio.h>
2 #define INF 255
3 #define MAXN 16
4
5 int Len[MAXN]; /* int Prev[MAXN]; */
6
7 int allvisited(int *a, int n) {
8     int i, r = 1;
9     for (i = 0; i < n; i++) { r *= a[i]; }
10    return r;
11 }
12
13 void compute(int *w, int n, int s, int d) {
14     int i, j, next, min, visited[MAXN];
15
16     for (i = 0; i < n; i++) {
17         Len[i] = INF; visited[i] = 0; /* Prev[i] = -1; */
18     }
19
20     i = s; Len[i] = 0; visited[i] = 1;
21
22     while (allvisited(visited, n) == 0) {
23         min = INF; next = d;
24         for (j = 0; j < n; j++) {
25             if (visited[j] == 1) continue;
26             if (Len[j] > Len[i] + w[i*n+j]) {
27                 Len[j] = Len[i] + w[i*n+j]; /* (ア) */
28             }
29             if (min > Len[j]) {
30                 min = Len[j]; next = j;
31             }
32         }
33         i = next; visited[i] = 1;
34     }
35 }
36
37 int main() {
38     int w[] = { 0, 2, 5, INF,
39                2, 0, 1, 1,
40                5, 1, 0, 2,
41                INF, 1, 2, 0 };
42     int n = 4;
43
44     compute(w, n, 0, 3);
45     printf("Shortest Path Length from 0 to 3 is %d\n", Len[3]);
46
47     /* (イ) printf("\n"); */
48
49     return 0;
50 }
```

図 1 : 2 頂点間の最短経路長を出力するプログラム

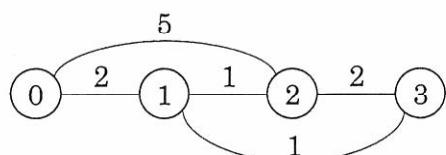


図 2 : 図 1 で扱うグラフ

配点： (1-1-1) 16 点, (1-1-2) 6 点, (1-1-3) 8 点, (1-2-1) 10 点, (1-2-2) 9 点, (1-2-3) 12 点
 (2-1) 14 点, (2-2-1) 20 点, (2-2-2) 30 点

(1) 計算機 (computer) における数の表現と演算に関する以下の各小間に答えよ。解答は全て解答用紙の太線内に書くこと。

(1-1) 計算機において整数を扱う場合、符号なし整数 (unsigned integer) と符号付き整数 (signed integer) があり、符号付き整数の表現方法には、符号絶対値表現 (signed magnitude representation), 1 の補数 (ones' complement) 表現, 2 の補数 (two's complement) 表現がある。この時、以下に答えよ。

(1-1-1) 整数を 10 ビットで表す場合、符号なし整数、符号絶対値表現、1 の補数表現、2 の補数表現それぞれで表すことができる数の最小値、最大値を 10 進数で示せ。また、それらに対応するビット列を示せ。

(1-1-2) 10 進数 -52 を 10 ビットの符号絶対値表現、1 の補数表現、2 の補数表現それぞれで表したビット列を示せ。

(1-1-3) 1010110011 が 10 ビットの符号なし整数、符号絶対値表現、1 の補数表現、2 の補数表現それぞれで表されたビット列である時、このビット列が表す数値を 10 進数で示せ。

(1-2) 整数の加算を行う回路に、桁上げ伝搬加算器 (リップル桁上げ加算器、ripple carry adder)、桁上げ先見加算器 (carry lookahead adder) がある。4 ビットのこれらの加算器の構成を、それぞれ、図 1、図 2 に示す。ここで、 FA_i ($i \in \{0, 1, 2, 3\}$) は全加算器 (full adder) であり、 s_i が加算出力、 c_{i+1} が桁上げ (carry) 出力である。また、 FA'_i は、 FA_i から桁上げ出力 c_{i+1} を取り除き、出力 $g_i = a_i b_i$ および出力 $p_i = a_i + b_i$ を付加した回路である。この時、以下に答えよ。

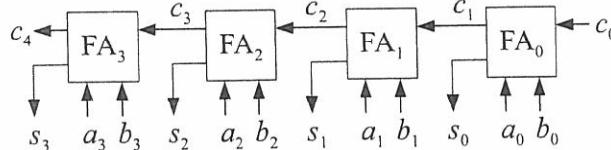


図 1. 桁上げ伝搬加算器の構成

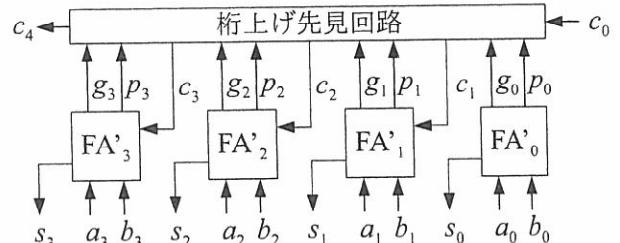


図 2. 桁上げ先見加算器の構成

(1-2-1) FA_i の s_i, c_{i+1} の論理式 (logic expression) を書け。ただし、 a_i, b_i, c_i を変数とする最簡積和形 (minimal sum-of-products expression) で書くこと。

(1-2-2) 図 2 の c_3 の論理式を書け。ただし、 $c_0, g_0, \dots, g_3, p_0, \dots, p_3$ を変数とする最簡積和形で書くこと。導出過程も示すこと。

(1-2-3) 全ての入力信号 a_i, b_i, c_i が揃ってから、 FA_i の出力信号 s_i, c_{i+1} が決定されるまでの遅延を、それぞれ、6 [ns], 4 [ns], FA'_i の出力信号 g_i, p_i, s_i が決定されるまでの遅延を、それぞれ、2 [ns], 2 [ns], 6 [ns] とする。また全ての入力 $c_0, g_0, \dots, g_3, p_0, \dots, p_3$ が揃ってから、桁上げ先見回路の出力 c_1, c_2, c_3, c_4 が決定されるまでの遅延を、それぞれ、4 [ns], 5 [ns], 6 [ns], 7 [ns] とする。

この時、全ての入力信号 $a_0, \dots, a_3, b_0, \dots, b_3, c_0$ が揃ってからそれぞれの加算器の出力が決定されるまでの最大遅延 (maximum delay) を求めよ。また、それぞれのクリティカルパス (最大遅延経路、critical path) の一つを、信号名を用いて $a_0 \rightarrow c_1 \rightarrow s_1$ の様に示せ。

(2) プロセス管理とスケジューリング (scheduling) に関する以下の各小間に答えよ。解答は全て解答用紙の太線内に書くこと。

(2-1) 以下の文中の空欄 (ア) ~ (キ) それぞれに最も適切な語を選択肢から一つ選び、その記号を答えよ。ただし、同じ選択肢を複数回用いてはならない。

单一プロセッサ (uniprocessor) を時分割制御するなどして、ユーザに複数プログラムが同時実行されているように見せかける多重化方式を (ア) という。例えば、入出力処理を多く含む (イ) 型プロセスを実行する場合、プロセッサによる処理から入出力処理へと切り替わる際に、そのプロセスが (ウ) されて実行中状態から (エ) へと遷移し、実行可能状態にある別のプロセスが (オ) されて、実行中状態へと遷移する。また、実行中状態のプロセスと比べて、処理時間が短いなどの理由で優先度の高いプロセスが実行可能キュー (ready queue) にある場合は、実行中状態のプロセスを (カ) して強制的に実行可能状態に遷移させ、優先度の高いプロセスを実行する。このように、プロセスを適切にスケジューリングすることで、(キ) やスループットの増加を図っている。

選択肢

- (a) マルチプログラミング (multi-programming), (b) 並列処理 (parallel processing), (c) ブロック (イベント待ち, block), (d) 横取り (preemption), (e) 実行中状態 (running state), (f) 実行可能状態 (ready state), (g) 待ち状態 (waiting state; ブロック状態, blocked state), (h) ディスパッチ (dispatch), (i) ウェイクアップ (wake-up), (j) プロセッサバウンド (processor bound; CPU バウンド, CPU bound), (k) I/O バウンド (I/O bound), (l) プロセッサ利用率 (processor utilization rate), (m) ターンアラウンド時間 (turn-around time)

(2-2) 表 1 にプロセス P1, P2, P3, P4, P5, P6 が生成される時刻、およびそれぞれのプロセスの処理時間を示す。

なお、プロセス P3 は、表 2 に示す通り、入出力装置による処理を含む三つのサブプロセス P3-A, P3-B, P3-C を、順次 (sequentially) 実行する I/O バウンド型プロセスであり、それ以外のプロセス P1, P2, P4, P5, P6 には入出力処理は含まれない。下記の仮定に留意し、次の (2-2-1) および (2-2-2) に答えよ。

仮定

1. 単一プロセッサを利用する。
2. あるプロセスが生成または横取りされた時刻から、そのプロセスが実行可能キューに格納される時刻までに要する処理時間を 0 とする。
3. あるプロセスが終了した、または横取りされた時刻から、次のプロセスがプロセッサにディスパッチされる時刻までに要する処理時間（プロセス切替えに関する処理時間）を 0 とする。

(2-2-1) 四つのプロセス P1, P2, P4, P5 を実行することを考える。横取り無し (non-preemption) として、以下の二つのスケジューリングアルゴリズム (algorithm) (a) および (b) を用いた場合のプロセス P1, P2, P4, P5 のターンアラウンド時間とその平均をそれぞれ求めよ。

- (a) 到着順 (first come first served) 方式
- (b) 最小処理時間順 (shortest processing time first) 方式

(2-2-2) 次に、(2-2-1) とは異なる組み合わせの四つのプロセス P1, P3, P4, P6 を実行することを考える。横取り有りとして、表 3 にある優先度順による二つのスケジューリングアルゴリズム (c) および (d) を用いた場合の (i) プロセス P1, P3, P4, P6 のターンアラウンド時間とその平均、および (ii) 全プロセス終了までのプロセッサ平均利用率[%]をそれぞれ求めよ。なお、最小残余時間 (shortest remaining time) は、プロセッサの処理残余時間と入出力装置の処理残余時間の和として考えるものとする。

表 1. プロセスの生成時刻と処理時間

プロセス	生成時刻	処理時間
P1	0	20
P2	5	40
P3	10	40
P4	25	30
P5	40	10
P6	45	10

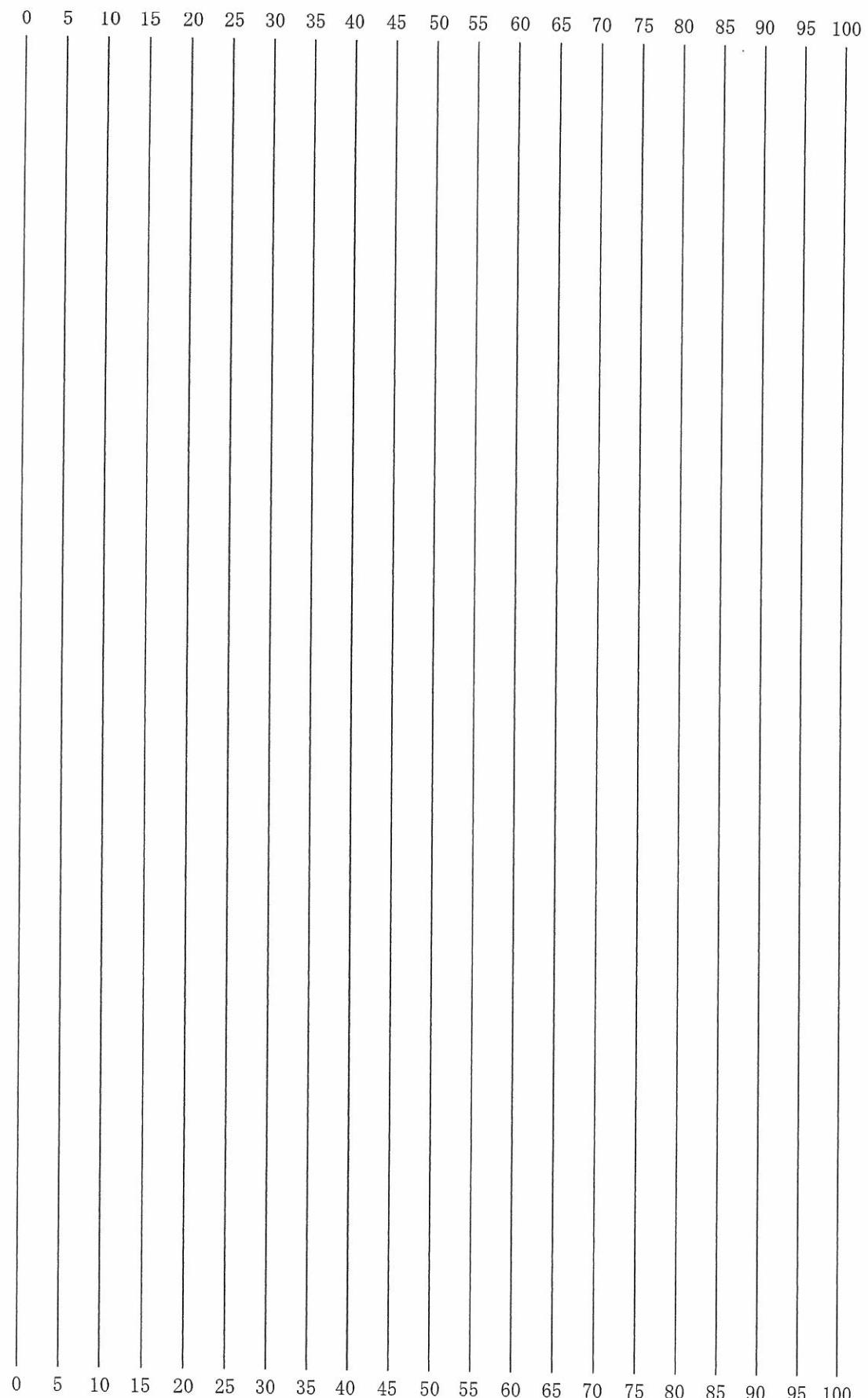
表 2. P3 のサブプロセスの利用資源と処理時間

サブプロセス	利用資源	処理時間
P3-A	プロセッサ	10
P3-B	入出力装置	20
P3-C	プロセッサ	10

表 3. 横取り有りの各スケジューリングアルゴリズムにおける優先度順

方式	第一優先	第二優先	第三優先
(c)	最小残余時間順	到着順	---
(d)	プロセッサバウンド型(CPU バウンド型)プロセス よりも I/O バウンド型プロセスを優先	最小残余時間順	到着順

(計算用紙)



3 【選択問題】離散構造

(情報工学 7/15)

配点:(1-1)~(1-2) 各 10 点, (1-3) 20 点, (2-1)~(2-2) 各 10 点, (2-3-1) 10 点, (2-3-2) 15 点, (3-1)~(3-4) 各 10 点

- (1) 命題論理 (propositional logic) に関する以下の各小間に答えよ. \rightarrow は含意 (implication) を表す論理演算子とする. また, 公理系 (system of axioms) X において, 論理式集合 (set of logical formulae) Γ から論理式 (logical formula) P が証明可能 (provable) であるとき, $\Gamma \vdash_X P$ と書くこととする.

命題論理の公理系 S および公理系 D を次のように定める.

[公理系 S] P, Q, R を任意の論理式を表わす変数とする. S の公理 (axiom) は次の A1 および A2 の 2 つである.

$$A1: P \rightarrow (Q \rightarrow P)$$

$$A2: (P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$$

また, S の推論規則は次の B1 である.

$$B1: P \text{ と } P \rightarrow Q \text{ から } Q \text{ を得る.}$$

[公理系 D] P, Q を任意の論理式を表わす変数とし, Γ を論理式の集合とする. D の公理は次の D1 のみである.

$$D1: \Gamma, P \vdash_D Q \text{ ならば } \Gamma \vdash_D (P \rightarrow Q)$$

また, D の推論規則は公理系 S の B1 と同じとする.

公理系 S と公理系 D が等価であることを示すには,

- (a) 公理系 S のもとで α を証明し,
(b) 公理系 D のもとで β を証明すればよい.

(1-1) 上記の空欄 α および β を埋めよ.

(1-2) 公理 D1 は一般に何とよばれているか答えよ.

(1-3) 上記の (b) に該当する証明を示せ.

- (2) 述語論理 (predicate logic) において, \Leftrightarrow , \rightarrow , \wedge , \vee , \neg を, それぞれ等価 (equivalence), 含意 (implication), 論理積 (conjunction, and), 論理和 (disjunction, or), 否定 (negation, not) を表す論理演算子とする. また, 空でない任意の有向グラフ (directed graph) $G = (V, E)$ (ただし $V, E (E \subseteq V \times V)$ はそれぞれ頂点 (vertex) および有向辺 (directed edge) の集合) において, 以下の述語を定義する.

$$n(x, y) \Leftrightarrow x \neq y \wedge x \in V \wedge y \in V \text{ であるとき真 (true), そうでないとき偽 (false)}$$

$$e(x, y) \Leftrightarrow (x, y) \in E \text{ であるとき真, そうでないとき偽}$$

以下の各小間に答えよ.

(2-1) $\exists y \forall x e(x, y)$ を真とする G において, $\forall x \exists y e(x, y)$ が真であることを説明せよ.

(2-2) 頂点数が 4 である G のうち, $\exists z \forall x \forall y (n(x, z) \wedge n(y, z) \rightarrow e(x, z) \wedge \neg e(z, x) \wedge \neg e(x, y) \wedge \neg e(y, x))$ を真とするものを図示せよ.

(2-3) V のある部分集合 Z が, 以下の (条件 X) を満たすものとする.

(条件 X) 集合 Z に属さない V の各頂点は, Z に属する頂点からの有向辺を少なくとも 1 本持つ
また, 以下の述語を定義する.

$$Z(x) \Leftrightarrow x \in Z \text{ であるとき真, そうでないとき偽}$$

以下の問 (2-3-1) および問 (2-3-2) に答えよ.

(2-3-1) (条件 X) を表す閉論理式 (closed formula) を A とする. A を, $Z(x)$ および $e(x, y)$ を用いて記述せよ.

- (2-3-2) $V = \{v_1, v_2, v_3, v_4\}$, $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\}$ とする。 v_2 および v_4 が集合 Z に含まれなければ、 $Z = \{v_1, v_3\}$ となることを、 A の変数に値を代入して導き出せる命題論理式を用い、導出原理 (resolution principle) により示せ。論駁過程 (refutation process) も示すこと。
- (3) X, Y を空でない有限集合 (finite set) とし、 f, g をそれぞれ X から Y への写像 (map) とする。 X から直積 $Y \times Y$ への写像 h を、 $x \in X$ について $h(x) = (f(x), g(x))$ で定義するとき、以下の各命題の真偽を答えよ。ただし真のときは証明を示し、偽のときは反例を示せ。
- (3-1) h が全射 (surjective) ならば、 f が全射かつ g が全射である
 - (3-2) f が全射かつ g が全射ならば、 h が全射である
 - (3-3) h が単射 (injective) ならば、 f が単射または g が単射である
 - (3-4) f が単射または g が単射ならば、 h が単射である

4

【選択問題】計算理論

(情報工学 9/15)

配点：(1-1-1) 15 点, (1-1-2) 15 点, (1-2-1) 10 点, (1-2-2) 20 点, (2-1) 20 点, (2-2) 20 点, (2-3) 25 点

(1) 以下のようなプログラム Q を考える。

1. プログラム Q は整数 (integer) $x \in \mathbf{N}$ を引数 (argument) として実行される。 \mathbf{N} は整数の集合を表す。
2. プログラム Q は、関数 (function) a, b, c, d を呼び出すことがある。関数 $a \sim d$ の中からは、関数 $a \sim d$ を呼び出すことはない。

アルファベット (alphabet) を $\Sigma = \{a, b, c, d\}$ とし、プログラム Q の実行において関数 $a \sim d$ がどのような順序で呼び出されるかを、呼び出される順にこれらの関数名を並べた Σ 上の文字列 (string) として表す。また、引数が x の場合におけるこの文字列を $P(x)$ と表記する。例えば、引数が x_0 のとき、プログラム Q は関数 a を呼び出し、次に関数 b を呼び出し、その後、終了するのであれば、 $P(x_0) = ab$ となる。

(1-1) Σ 上の言語 L_1 を、以下が成り立つような言語とする。

任意の $x \in \mathbf{N}$ について

$$P(x) \in L_1 \iff x \text{ が引数のときにプログラム } Q \text{ は関数 } a \text{ を 1 回以上呼び出す}$$

以下の (1-1-1) と (1-1-2) に答えよ。

(1-1-1) 下の遷移表 (transition table) で表される決定性有限オートマトン (deterministic finite automaton) を M_1 とする。 M_1 が言語 L_1 を受理するように、遷移表の空欄 (ア)～(ク) を状態 p または q で埋めよ。ただし、表の左端の列において、 \rightarrow で初期状態 (initial state) を、 $*$ で受理状態 (accepting state) を表している。

状態	入力記号				
		a	b	c	d
	$\rightarrow p$	(ア)	(イ)	(ウ)	(エ)
*	q	(オ)	(カ)	(キ)	(ク)

(1-1-2) 上記プログラム Q について下の式が成り立つとき、プログラム Q について成り立�性質を簡単に説明せよ。ただし、 $\overline{L_1}$ は L_1 の補集合 (complement) を、 \emptyset は空集合 (empty set) を表す。

$$\{P(x) \mid x \in \mathbf{N}\} \cap \overline{L_1} \neq \emptyset$$

(1-2) 正則表現 (regular expression) $(ab + cd)^*$ で表される言語を L_2 とする。以下の (1-2-1) と (1-2-2) に答えよ。

(1-2-1) $P(x) \in L_2$ であるような x を引数としてプログラム Q を実行した場合、関数 $a \sim d$ がどのような順序で呼び出されるか、簡単に説明せよ。

(1-2-2) 言語 $\overline{L_2}$ (すなわち L_2 の補集合) を受理する決定性有限オートマトン M_2 を求め、遷移表の形で示せ。初期状態を \rightarrow で、受理状態を $*$ で明示すること。また、どのようにして求めたのか簡単に説明せよ。

(2) 下の枠内に示す文脈自由言語 (context-free language) に対する反復補題 (pumping lemma) とその証明の概要 (proof outline) を読んで、以下の各小間に答えよ。

- (2-1) 任意に整数 (integer) k (≥ 0) が与えられる。構文木 (parse tree) の高さが $k+1$ である文字列 (string) z に関して、 z の長さの最大値 (maximum) を表す式を k を用いて示し、理由を簡単に説明せよ。構文木の高さとは、根 (root) から各葉 (leaf) への経路 (path) の長さの最大値と定める。根の子 (child) がすべて葉である構文木の高さは 1 である。
- (2-2) 証明の概要で「 $A_0, A_1, A_2, \dots, A_k$ に同一の変数が 2 回以上現れる」と述べている。 k と $|V|$ (V の要素数) がどのような関係にあれば同一の変数が 2 回以上現れるかを不等式 (inequality) で示し、理由を簡単に説明せよ。
- (2-3) 空欄 (ア) を式で埋め、理由を簡単に説明せよ。

反復補題: 言語 L を任意の文脈自由言語とする。 L に対してある定数 (constant) n が存在し、 L に属する長さ n 以上の任意の文字列 (string) z は以下の条件を満たす $z = uvwxy$ の形に分解 (decompose) できる。

1. $|vwx| \leq n$ である。
2. $vx \neq \epsilon$ である。ただし ϵ は空文字列 (empty string) である。
3. 任意の $i \geq 0$ に対し $uv^iwx^i y \in L$ である。

ただし文字列 s に対して、 $|s|$ は s の長さを表すものとする。

証明の概要: 言語 L を生成する文脈自由文法 (context-free grammar) を $G = (V, T, P, S)$ とする。なお V は変数 (variable, あるいは非終端記号 non-terminal symbol) の集合、 T は終端記号 (terminal symbol) の集合、 P は生成規則 (production rule) の集合、 S は開始記号 (start symbol) で $S \in V$ である。一般性を失うことなく、 G はチョムスキーベルト (Chomsky normal form) であると仮定する。チョムスキーベルトとは、生成規則の右辺が二つの変数 ($A \rightarrow BC$ の形)、あるいは一つの終端記号 ($A \rightarrow a$ の形) に限定したものをいう。

任意の $z \in L$ に対する構文木 (parse tree) において、根 (root) から葉 (leaf) への経路 (path) のうちで最も長いものを考える (構文木の根は開始記号 S であり、葉は終端記号である)。ただし最も長い経路が複数ある場合は、それらの中で任意のものを選ぶ。図 2-1 に示す通り、その経路に現れる変数を根から順に $A_0 (= S)$, A_1 , A_2 , ..., A_k とし、 A_k の子 (child) は葉で終端記号である。なお $A_0, A_1, A_2, \dots, A_k \in V$ である。

$n = \boxed{\text{(ア)}}$ とし、任意に $z \in L$ (ただし $|z| \geq n$) を選ぶと k の値は十分に大きく、 $A_0, A_1, A_2, \dots, A_k$ に同一の変数が 2 回以上現れる。そのような変数を $A \in V$ とし、図 2-2 で示す通り $A = A_p = A_q$ ($p < q$) とする。図 2-3 に示す通り $S \xrightarrow{*} uAy \xrightarrow{*} uvAxy \xrightarrow{*} uvwxy = z$ は文字列 z の導出 (derivation) であり、 $A \xrightarrow{*} vAx$ かつ $A \xrightarrow{*} w$ が成り立つ。

以上より、導出 $S \xrightarrow{*} uAy \xrightarrow{*} uv^iAx^iy \xrightarrow{*} uv^iwx^iy$ を構成できるので、 $uv^iwx^iy \in L$ が成り立つ。□

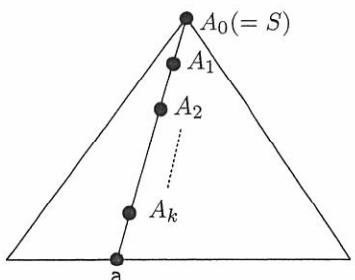


図 2-1

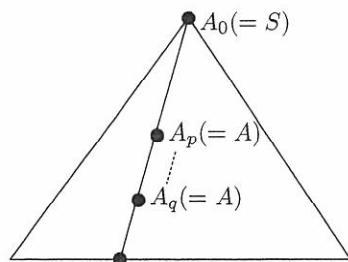


図 2-2

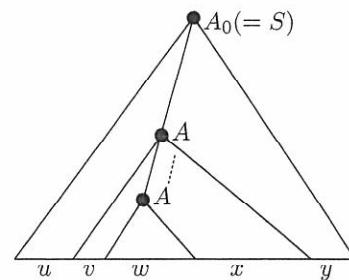


図 2-3

【選択問題】ネットワーク

(情報工学 11/15)

配点: (1-1-1) 5 点, (1-1-2) 14 点, (1-2) 21 点
(2-1) 18 点, (2-2-1) 10 点, (2-2-2) 10 点, (2-2-3) 15 点, (2-3) 32 点

- (1) 記憶がない情報源 (memoryless information source) の 2 元ハフマン符号化 (binary Huffman coding)について考える。

(1-1) 情報源記号が a, b , 各記号の生起確率 (occurrence probability) が $p(a) = 0.8, p(b) = 0.2$ である情報源 S_0 を考える。以下では通信路記号を 0, 1 とし, 2 元ハフマン符号化の復号木 (decoding tree) における通信路記号の割り当ては一通りだけを示せばよい。なお、復号木は符号の木 (code tree)とも呼ばれている。

(1-1-1) 情報源 S_0 の 2 元ハフマン符号化の復号木および平均符号語長 (average code length) を書け。

(1-1-2) 情報源 S_0 の記号を 2 個ずつまとめて出力する情報源、すなわち S_0 の 2 次拡大 (second extension) S_0^2 を考える。記号列 aa, ab, ba, bb それぞれの生起確率を求めよ。また、 S_0^2 の 2 元ハフマン符号化のすべての復号木と、情報源記号 1 個あたりの平均符号語長を求めよ。

- (1-2) 一般の記憶がない情報源 S の n 次拡大 (n -th extension) S^n について考える。以下では、 S^n の 2 元ハフマン符号化の、情報源記号 1 個あたりの平均符号語長を l_n とする。以下の文章の空欄 (あ) ~ (う) を埋めよ。

情報源 S の 2 を底とするエントロピー (entropy) を $H(S)$ とするとき、拡大情報源 S^n のエントロピーは (あ) である。また、シャノンの情報源符号化定理 (Shannon's source coding theorem, シャノンの第一定理ともいう) から、 $H(S) \leq l_n < H(S) + (い)$ が成り立つ。

この式を用いると、拡大情報源の情報源記号 1 個あたりの平均符号語長に関する以下の議論が成立する。今、ある正の整数 n_0 について、 l_{n_0} が $H(S)$ に等しくないとする。すなわち、ある $\delta > 0$ を用いて $l_{n_0} = H(S) + \delta$ と表せるとする。このとき、 $n \geq (う)$ を満たすように n を選べば、 l_n は l_{n_0} より必ず小さくなる。

- (2) トランスポート層プロトコル (transport layer protocol) についての以下の各小間に答えよ。

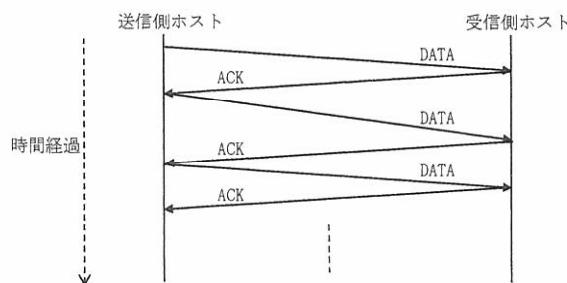
- (2-1) トランスポート層プロトコルが用いるフロー制御 (flow control), ふくそう制御 (congestion control), および再送制御 (retransmission control) に関する以下の文章の空欄 (あ) ~ (か) に当てはまる最も適切な語句を選択肢から選び、その記号を答えよ。なお、異なる空欄には異なる語句が当てはまる。

フロー制御は、送信側ホストと受信側ホストの (あ) や (い) などの違いを考慮し、(う) を決定するための制御である。ふくそう制御は、ネットワークが (え) になることで、ネットワーク内で発生するセグメントの廃棄や (お) の増加を防止し、ふくそう状態から回復するために、(う) を決定するための制御である。再送制御は、再送タイマ (retransmission timer) 等を利用し、ネットワーク内のセグメントの (か) を検出し、再送を行うための制御である。

選択肢

- | | |
|---------------------------------------|-----------------------------|
| (a) セグメントサイズ (segment size) | (g) IP アドレス (IP address) |
| (b) データ転送速度 (data transmission speed) | (h) 喪失 (loss) |
| (c) 過負荷 (overload) | (i) メモリサイズ (memory size) |
| (d) 転送遅延 (transmission latency) | (j) ポート番号 (port number) |
| (e) 経路 (route) | (k) スループット (throughput) |
| (f) ループ (loop) | (l) 処理速度 (processing speed) |

(2-2) 以下の図は、代表的なフロー制御手法であるストップアンドウェイトフロー制御 (stop and wait flow control) を用いたデータ転送の様子を表している。図中の矢印はセグメントの送信を意味し、DATA はデータセグメント (data segment) を、ACK は確認応答セグメント (acknowledgement segment) をそれぞれ表す。なお、ここでは、セグメントの喪失は発生しないものとする。



- (2-2-1) データセグメントが P [バイト] のデータと H [バイト] のヘッダから構成されるものとする。また、データセグメントをネットワーク層に受け渡してから、対応する確認応答セグメントをネットワーク層から受け取るまでの時間である、ラウンドトリップ時間 (round trip time) の平均値を T [秒] とする。上図における、データ転送の平均スループット (average throughput) [ビット/秒] を答えよ。
- (2-2-2) (2-2-1) の結果を用いて、データ転送のスループットの観点から、ストップアンドウェイトフロー制御の問題点を説明せよ。
- (2-2-3) ストップアンドウェイトフロー制御に比べてデータ転送のスループットを向上することができるフロー制御の一つである、スライディングウインドウフロー制御 (sliding window flow control) を用いたデータ転送の様子を、上図と同様の方法で示せ。なお、ウィンドウサイズを 3 [セグメント] とし、データ転送の様子のみ解答すること。
- (2-3) データセグメントの送信時に作動させる再送タイマの設定値が大きすぎる場合、および小さすぎる場合に発生する問題点をそれぞれ述べよ。また、再送タイマの設定値を適切に決定するために、TCP (Transmission Control Protocol) の一般的な実装 (implementation) において用いられる手法を説明せよ。

6

【選択問題】電子回路と論理設計

(情報工学 13/15)

配点：(1) 20 点, (2) 20 点, (3) 20 点, (4) 20 点, (5) 20 点, (6) 15 点, (7) 10 点

順序回路 (sequential circuit) について、以下の各間に答えよ。

- (1) 図 1 の状態遷移図 (state transition diagram) と等価な状態遷移出力表 (state transition table with output) を作成せよ。ただし、入力は x 、出力は y 、状態は S_0 から S_5 で表すこととする。
- (2) (1) の順序回路の状態を表 1 のように割り当てた場合、 x, Q_2, Q_1, Q_0 を変数とする y, Q_2^+, Q_1^+, Q_0^+ の最簡積和形 (最小積和形、minimal sum-of-products expression) の論理式 (logic expression) を導出せよ。ただし、現在の状態 (current state) を (Q_2, Q_1, Q_0) 、次状態 (next state) を (Q_2^+, Q_1^+, Q_0^+) で表すものとする。
- (3) (2) で求めた最簡積和形の論理式を用いて、図 2 の CL0, CL1, CL2, CLY 内の組合せ回路 (combinational circuit) をそれぞれ最小のゲート数で実現する。このときの CL0, CL1, CL2, CLY 内のゲート数 (gate count) をそれぞれ求めよ。ただし、論理ゲートは、2 入力 NAND ゲートおよび 3 入力 NAND ゲートのみが使用できるものとする。最小であることの証明は不要である。

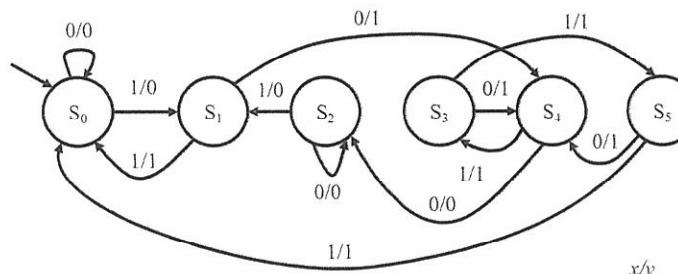


図 1

表 1

状態	状態割り当て		
	Q_2	Q_1	Q_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1

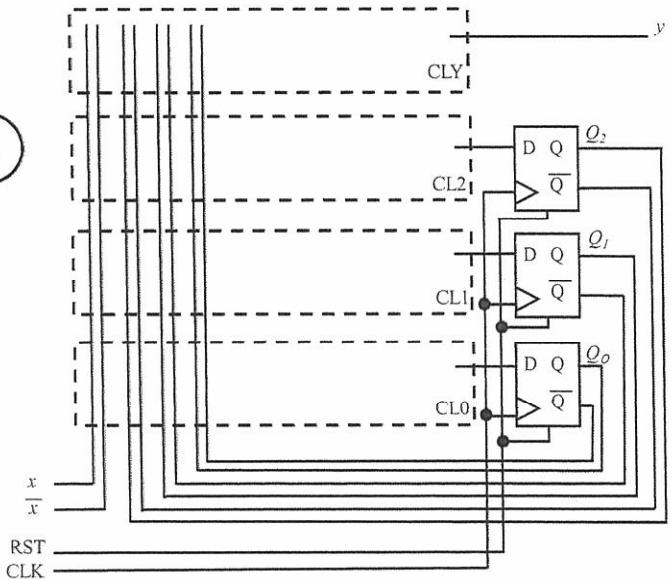


図 2

- (4) 図 1 の状態遷移図から共通化できる状態をまとめ、状態数最小化 (state minimization) を行う。共通化した状態を新たに S_0' , S_1' , S_2' , S_3' とした場合の状態割り当てを表 2 に示す。[A]から[F]を埋めて表 2 を完成させよ。ただし、□の中にはいる状態は、 S_0 から S_5 のいずれかもしくは状態がないことを表す記号 (-) である。
- (5) (4) の順序回路の状態を表 2 のように割り当てた場合、 x, Q_l, Q_o を変数とする y, Q_l^+, Q_o^+ の最簡積和形の論理式を導出せよ。ただし、現在の状態を (Q_l, Q_o) 、次状態を (Q_l^+, Q_o^+) で表すものとする。
- (6) 2 入力 NAND ゲートのハードウェアコスト (hardware cost) を 2, 3 入力 NAND ゲートのハードウェアコストを 3, D フリップフロップのハードウェアコストを 12 とする。状態数最小化前の(2)の順序回路と状態数最小化後の(5)の順序回路のハードウェアコストをそれぞれ求めよ。ただし、順序回路のハードウェアコストは、2 入力 NAND ゲート、3 入力 NAND ゲート、D フリップフロップのハードウェアコストの総和で求められる。
- (7) 順序回路においてはその状態数が減らせても、ハードウェアコストが必ずしも減るわけではない。その理由について、組合せ回路と記憶回路 (memory element) のハードウェアコストの観点から簡潔に説明せよ。

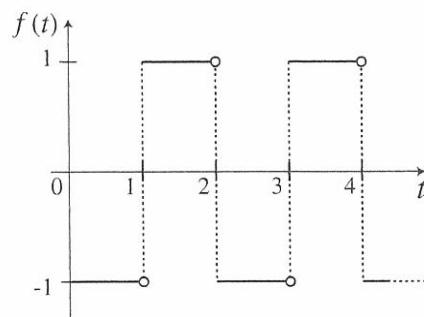
表 2

新状態	共通化された 状態	状態割り当て	
		Q_l	Q_o
S_0'	{ S_0 , [A] }	0	0
S_1'	{ S_1 , [B] }	0	1
S_2'	{ [C] , [D] }	1	0
S_3'	{ [E] , [F] }	1	1

配点: (1-1)30 点, (1-2)30 点, (2-1)20 点, (2-2)45 点

以下の各間に答えよ.

(1) 関数 $f(t) = (-1)^{n+1} \quad \{n \leq t < n+1 \quad (n = 0, 1, 2, \dots)\}$ は下記のように図示される. この関数 $f(t)$ について以下の小間に答えよ. ただし, 関数 f は $[0, +\infty)$ で定義される.



(1-1) $\int_0^\infty e^{-st} f(t) dt$ を級数 (series) の形式で求めよ. ただし s は複素変数 (complex variable) である.

(1-2) (1-1) の結果を用いて $f(t)$ のラプラス変換 (Laplace transform) $\mathcal{L}[f(t)]$ が収束する条件 (convergence condition) と, この条件が満たされたときの $\mathcal{L}[f(t)]$ を求めよ.

(2) 次の正弦波信号 (sine wave signal) $u(t)$ について, 以下の小間に答えよ. ただし, $V_m > 0$ とする.

$$u(t) = V_m \sin \frac{2\pi}{T} t$$

(2-1) $u(t)$ を全波整流 (full-wave rectification) した波形 (waveform) $v(t)$ を式で示し, 図示せよ. なお, 全波整流とは交流電流 (alternating current) の正・負両波とも整流 (rectification) し, 流れの向きを同じにすることである.

(2-2) $v(t)$ をフーリエ級数展開 (Fourier series expansion) せよ.

【必須問題】アルゴリズムとプログラミング

(情報工学 1/15)

配点：(1) 30 点, (2-1) 10 点, (2-2) 15 点, (2-3) 20 点, (2-4) 20 点, (3) 30 点

図に示す ANSI-C 準拠である C 言語のプログラム (program) は、配列 (array) `values` の要素 (element) に格納された非負整数 (non-negative integer) のデータ (data) を、 r 進数 (r base number) ($2 \leq r \leq 256$) とみなして整列 (sort) し、出力 (output) するプログラムである。以下の各間に答えよ。

- (1) プログラムの 10～26 行目の `for` 文の処理において 1 巡目 ($d=1$ のとき) および 2 巡目 ($d=2$ のとき) の終了時における配列 `buckets` の内容を、解答用紙の太線内の空欄を埋めることにより答えよ。なお、配列の各要素は 0 で初期化される。配列の各要素 `buckets[x][y]` において、 x は 3 以上の値、 y は 5 以上の値を取りうるが、これらの場合の内容は答えなくてよい。

<code>buckets[x][y]</code>	1 巡目 ($d=1$)					2 巡目 ($d=2$)				
	$y=0$	$y=1$	$y=2$	$y=3$	$y=4$	$y=0$	$y=1$	$y=2$	$y=3$	$y=4$
$x=0$										
$x=1$										
$x=2$										

- (2) プログラムで実現されている整列アルゴリズム (sorting algorithm) に関する以下の各小間に答えよ。

- (2-1) この整列アルゴリズムは、一般に何と呼ばれているか名称を答えよ。
- (2-2) 整列対象のすべてのデータを r 進数で表現したときの最大の桁数 (maximum digit number) を k とする。整列対象のデータの数を n としたとき、プログラムで実現されている整列アルゴリズムの時間計算量 (time complexity) のオーダー表記 (order notation) を、その理由とともに答えよ。
- (2-3) この整列アルゴリズムは、整列対象のデータ同士の比較を伴わない整列アルゴリズムである。この整列アルゴリズムでは、どのような工夫によって整列対象のデータ同士の比較を伴わない整列を可能にしているかを答えよ。
- (2-4) 整列対象のデータ同士の比較を伴う整列アルゴリズムと比較して、プログラムで実現されている整列アルゴリズムの特徴を、時間計算量および空間計算量 (space complexity) の観点から答えよ。
- (3) 配列 `values` の要素に格納されたデータを降順 (descending order) に整列するように、プログラムを変更することを考える。そのためには、下線 (ア) および下線 (イ) をどのように記述すればよいか、適切な式をそれぞれ答えよ。

```

1 #include <stdio.h>
2 #define MAXR 256
3 #define MAXN 1000
4 void sort(int values[], int numvalues, int r, int maxdigit){
5     int rd, d, b, i, n, j;
6     int buckets[MAXR][MAXN];
7     int numbucket[MAXR];
8
9     rd = 1;
10    for(d = 1; d <= maxdigit; d++){
11        for(b = 0; b < r; b++){
12            numbucket[b] = 0;
13        }
14        for(i = 0; i < numvalues; i++){
15            n = (values[i] / rd) % r;
16            buckets[n][numbucket[n]++] = values[i];
17        }
18
19        i = 0;
20        for( b = 0; b < r; b++ ){
21            (ア)
22            for( j = 0; j < numbucket[b]; j++ ){
23                (イ)
24                values[i++] = buckets[b][j];
25            }
26        }
27    }
28    int main(){
29        int i;
30        int values[5] = {12, 21, 1, 11, 2};
31
32        sort(values, 5, 10, 2);
33        for(i = 0; i < 5; i++){
34            printf("%d\n", values[i]);
35        }
36        return 0;
37    }

```

図 プログラム

配点： (1-1) 20 点, (1-2-1)~(1-2-3) 各 5 点, (1-2-4) 10 点, (1-3-1) 10 点, (1-3-2) 10 点
 (2-1-1) 10 点, (2-1-2) 15 点, (2-2-1) 10 点, (2-2-2) 10 点, (2-3) 15 点

(1) 計算機 (computer), 特に, 中央処理装置 (CPU: central processing unit) に関する以下の各小間に答えよ. 解答は全て解答用紙の太線内に書くこと.

(1-1) 以下の文章の空欄 (a)~(e) に当てはまる最も適切な語句を, 下記の選択肢から選び, 記号で答えよ.

計算機の構成方式のことを (a) と呼ぶ. 現在, 実用的に利用されている大部分の計算機は, 線形アドレス空間 (linear address space) を有するメインメモリ (main memory) 上にプログラム (program) 及びデータ (data) を置き, プログラムを逐次実行する, 等の (a) を採用している. この様な計算機は, (b) 計算機と呼ばれる.

(b) 計算機では, (c) が示すメインメモリアドレスから, プログラムの構成要素である機械語命令を読み出し, これをデコード (decode) して実行する. 機械語命令は, その種類を示す (d) 部と, 演算対象のデータである (e) の格納場所を示すアドレス部から成る.

【選択肢】

- | | |
|---------------------------------------|---------------------------------------|
| (ア) データフロー型 (dataflow architecture) | (イ) オペコード (operation code, opcode) |
| (ウ) プログラムカウンタ (program counter) | (エ) アーキテクチャ (architecture) |
| (オ) VLIW (very long instruction word) | (カ) オペランド (operand) |
| (キ) 命令レジスタ (instruction register) | (ク) メモリデータレジスタ (memory data register) |
| (ケ) スーパースカラ (superscalar) | (コ) ノイマン型 (von Neumann architecture) |

(1-2) 一つの機械語命令 (以下, 命令と略す) の処理を n 段のステージ (stage) に分け, 1 ステージを 1 クロックサイクル (clock cycle) で実行する同期型 CPU を考える. クロック周波数を f [Hz] とする. 以下の (1-2-1)~(1-2-4) に答えよ.

(1-2-1) m 個の命令をパイプライン (pipeline) 処理を用いて逐次的に実行する場合の実行時間 [s] を示せ.

(1-2-2) m 個の命令をパイプライン処理を用いて実行する場合の実行時間 [s] を示せ. 但し, パイプラインストール (pipeline stall) は無いものとする.

(1-2-3) m 個の命令をパイプライン処理を用いて実行する場合の単位時間あたりの命令実行数 [instructions/s] について, $m \rightarrow \infty$ における極限を示せ. 但し, パイプラインストールは無いものとする.

(1-2-4) パイプライン処理を用いた CPU の性能 (単位時間あたりの命令実行数) を高める手法の一つとして, ステージ数 n を増やす方法がある. この方法により CPU の性能を高めることが可能である理由を説明せよ.

(1-3) 一つの機械語命令 (以下, 命令と略す) の処理を, 命令フェッチ (IF: instruction fetch), デコード (D: decode), オペランドフェッチ (OF: operand fetch), 実行 (EX: execution), 結果の格納 (S: store) の 5 つのステージに分け, 1 ステージ 1 クロックサイクルのパイプライン処理を用いた同期型 CPU において, 以下の (1-3-1) および (1-3-2) に示す命令 1~3 を実行することを考える. 解答欄の命令 1 の例を参考に, 実行されるステージ名 “IF”, “D”, “OF”, “EX”, “S” を解答欄に記入せよ. パイプラインストールにより完了までに複数クロックサイクルが必要なステージは, 完了するクロックサイクルにステージ名を, それ以外のクロックサイクルに “→” を記入すること. なお, 以下の点を仮定する.

- パイプラインストールの原因としては, 構造ハザード (structural hazard) およびデータハザード (data hazard) を考える.
- 構造ハザードはメモリアクセス (memory access) の競合 (conflict) のみ考える.
- プログラムとデータは同じメインメモリ上にある.
- ある命令でレジスタに書き込まれた値を以降の命令で参照する場合, 前者の命令の格納 (S) が完了した後, 後者の命令のオペランドフェッチ (OF) が可能となる.

- (1-3-1) 命令 1 : MOV R1, (A) ; メインメモリアドレス A の内容をレジスタ R1 に転送
 命令 2 : MOV R2, (B) ; メインメモリアドレス B の内容をレジスタ R2 に転送
 命令 3 : ADD R1, R2 ; R1 + R2 の結果を R1 に代入
- (1-3-2) 命令 1 : MOV R1, (A) ; メインメモリアドレス A の内容をレジスタ R1 に転送
 命令 2 : INC R1 ; R1 + 1 の結果を R1 に代入
 命令 3 : MOV (B), R1 ; レジスタ R1 の内容をメインメモリアドレス B に転送

	クロックサイクル	0	1	2	3	4	5	6	7	8	9	10	11
(1-3-1)	命令1 : MOV R1, (A)	IF	D	OF	EX	S							
	命令2 : MOV R2, (B)												
	命令3 : ADD R1, R2												
(1-3-2)	クロックサイクル	0	1	2	3	4	5	6	7	8	9	10	11
	命令1 : MOV R1, (A)	IF	D	OF	EX	S							
	命令2 : INC R1												
	命令3 : MOV (B), R1												

(2) キャッシュメモリ (cache memory) 及びメインメモリ (main memory) で階層を形成しているメモリシステムを持つ計算機を考える。以下の各小間に答えよ。解答は全て解答用紙の太線内に書くこと。

(2-1) キャッシュメモリに存在する命令あるいはデータを読み出す際のアクセス時間 (access time) が 2 [ns] であり、キャッシュメモリに存在しない命令あるいはデータをメインメモリから読み出す際の、キャッシュメモリ及びメインメモリへのアクセス時間の和が 50 [ns] であるとする。また、メインメモリの容量はプログラムに対して十分大きく、プログラムの命令及びデータは全てメインメモリに格納されているものとする。以下の(2-1-1) 及び(2-1-2)に答えよ。

(2-1-1) プログラムを実行した結果、プログラムの命令あるいはデータを読み出す際の平均のキャッシュヒット率 (cache hit ratio, cache hit rate) が 80% であった。この時の、プログラムの命令あるいはデータを読み出す際の平均アクセス時間を求めよ。導出根拠も示せ。

(2-1-2) キャッシュメモリを、命令あるいはデータを読み出す際のアクセス時間が 2 [ns] のものから 4 [ns] のものに変更する。ただし、キャッシュメモリに存在しない命令あるいはデータをメインメモリから読み出す際の、キャッシュメモリ及びメインメモリへのアクセス時間の和は 50 [ns] のままであるとする。この時、命令あるいはデータを読み出す際の平均アクセス時間の増加を防止するためには、キャッシュメモリの容量を大きくするなどの方法により、キャッシュヒット率を高める必要がある。(2-1-1) で得られた平均アクセス時間を維持するために必要となるキャッシュヒット率を求めよ。導出根拠も示せ。

(2-2) 一般に、プログラムを実行するためにアクセスされる命令及びデータには、参照局所性 (locality of reference) がある。次の 2 種類の参照局所性のそれぞれについて説明せよ。

(2-2-1) 空間的参照局所性 (spatial locality of reference)

(2-2-2) 時間的参照局所性 (temporal locality of reference)

(2-3) キャッシュメモリとメインメモリで階層を形成することの利点をその理由と共に述べよ。

配点：(1-1)～(1-5) 各 6 点, (1-6) 7 点, (1-7) 7 点, (1-8-1) 6 点, (1-8-2) 15 点
 (2-1) 12 点, (2-2) 15 点, (2-3) 20 点, (2-4) 13 点

- (1) 情報論理 (mathematical logic) に関する以下の各小間に答えよ。ただし、論理式 (logic formula) の記述には以下の記号を用いる。 \rightarrow , \wedge , \vee , \neg はそれぞれ含意 (implication), 論理積 (conjunction, and), 論理和 (disjunction, or), 否定 (negation, not) を表す論理演算子とする。また、必要に応じて $\bigwedge_{1 \leq i \leq n} T_i$ (あるいは $\bigvee_{1 \leq i \leq n} T_i$) の表記を用いる。これは論理式 T_i の i を 1 から n (n は正整数) まで順次変えながら論理積 (あるいは論理和) で結合した式を意味している。なお、 $\bigwedge_{1 \leq i < j \leq n} T_{ij}$ は、 $\bigwedge_{1 \leq i \leq n-1} (\bigwedge_{i+1 \leq j \leq n} T_{ij})$ を意味する。

縦 n^2 マス×横 n^2 マスで構成されるパズル (puzzle) を考える。本パズルにおける規則は以下の通りである。

- 空いているマスに 1 から n^2 までの整数を入れる。
- 各列、各行および、太線で囲まれた $n \times n$ の各ブロック内に同じ整数を複数用いてはいけない。

図 1 は、 $n = 3$ における本パズルの問題例である。以降、このパズル問題を解くための制約式を和積形 (CNF: Conjunctive Normal Form) の命題論理式 (propositional formula) で与えることを考える。なお、和積形とは、一つ以上の和項の論理積で表される論理式である。また、和項とは、一つ以上のリテラル (literal) の論理和で表される論理式である。ここでリテラルとは、命題変数 (propositional variable) または命題変数の否定を意味する。 i ($1 \leq i \leq n^2$) 行 j ($1 \leq j \leq n^2$) 列のマスを $c(i, j)$ で表記する (行は上から順に $1, \dots, n^2$ 行目、列は左から順に $1, \dots, n^2$ 列目である)。

マス $c(i, j)$ に整数 k ($1 \leq k \leq n^2$) が入っているとき、かつ、そのときのみ真となる命題変数 x_{ijk} を導入する。以下の各小間に答えよ。

		列								
		1	2	3	4	5	6	7	8	9
行	1	7	3	2		1				
	2	4						2		
	3		5	1			3	8		
	4								1	
	5		8				2			
	6	9							7	
	7									
	8		2		6			3		
	9				7				9	

図 1: $n = 3$ の問題例

- (1-1) 図 1 において命題変数 $x_{115}, x_{214}, x_{841}$ の真偽をそれぞれ答えよ。
- (1-2) 縦 n^2 マス×横 n^2 マスで構成される本パズル問題における命題変数 x_{ijk} の総数はいくつか。 n を用いて示せ。
- (1-3) 「1 行 1 列目のマスには、1 以上 9 以下の整数が少なくとも一つ入る」ことを表す論理式を記述せよ。

- (1-4) CNF 式 $A(i, j)$ を「 i 行 j 列目のマスには、1 以上 n^2 以下の整数が少なくとも一つ入る」ことを表す論理式とする。論理式 $A(i, j)$ を記述せよ。
- (1-5) $A(i, j)$ を用いて、「どのマスについても、1 以上 n^2 以下の整数が少なくとも一つ入る」ことを表す論理式 A を示せ。
- (1-6) $\neg p \vee \neg q$ は、「 p, q がともに真になるということはない」ことを表す論理式である。これを用いて「どの行についても各整数は高々1回しか現れない」ことを表す CNF 式 B を作りたい。以下の空欄を埋めることで B を完成させよ。

$$B = \bigwedge_{1 \leq i \leq n^2} \bigwedge_{1 \leq j < l \leq n^2} \bigwedge_{1 \leq k \leq n^2} (\boxed{\quad})$$

- (1-7) 「どの列についても各整数が高々1回しか現れない」ことを表す CNF 式 C を示せ。

- (1-8) この小問では、 $n = 2$ として、図 2 の問題に着目する。

		1	2
1	2	4	
	1	2	4
			3

図 2: $n = 2$ の問題

図 2 の問題では、空いたマス（例えば 2 行 4 列目）にどのように整数を埋めても、パズルを解くことができない。ここでは、（命題 P）「図 2 の問題では、空いたマスにどのように整数を埋めても、パズルを解くことができない」ことを示したい。この命題 P を示すには、「太線で囲まれた $n \times n$ のどのブロックについても各整数が高々1回現れる」ことを意味する CNF 式を D 、図 2 における整数の配置を表す論理式を $Assign$ としたときに、（方針） $A \wedge B \wedge C \wedge D \wedge Assign$ の CNF 式が充足不能（unsatisfiable）であることを示せばよい。このとき、以下の (1-8-1)～(1-8-2) に答えよ。

- (1-8-1) 論理式 $Assign$ を示せ。

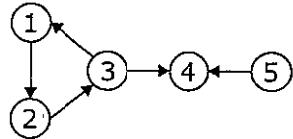
- (1-8-2) 上記の方針に沿って、命題 P が成り立つことを導出原理（resolution principle）を用いて具体的に示せ。

- (2) 空でない有限集合 (finite set) V 上の二項関係 (binary relation) E について、次のように二項関係 R_i と S_i ($i = 0, 1, 2, \dots$) を定義する。また、 V の各要素を頂点とし、 E の各要素を有向辺とする有向グラフ (directed graph) を G とする。ただし $(u, v) \in E$ に対しては、 u を有向辺の始点とし、 v を有向辺の終点とする。

$$\begin{aligned} R_0 &= S_0 = \{(v, v) \mid v \in V\} \\ R_{i+1} &= \{(u, v) \mid (\exists w \in V) [uR_i w \wedge wEv]\} \quad (i \geq 0) \\ S_{i+1} &= S_i \cup R_{i+1} \quad (i \geq 0) \end{aligned}$$

以下の各小間に答えよ。

- (2-1) 有向グラフ $G = (V, E)$ が下図の時、 R_3 を求めよ。



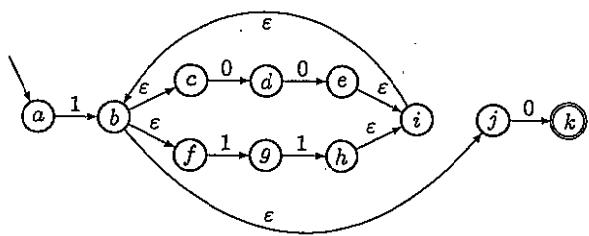
- (2-2) ある非負の整数 n が存在して $S_n = S_{n+1}$ であることを、背理法 (proof by contradiction) を用いて証明せよ。
- (2-3) 小問 (2-2) の n について、 S_n の逆関係 (inverse relation) を $S_n^{-1} = \{(u, v) \mid vS_n u\}$ とし、 $S = S_n \cap S_n^{-1}$ とする。 S が同値関係 (equivalence relation) であることを証明せよ。ただし、任意の非負整数 i, j について、 $((u, v) \in R_i) \wedge ((v, w) \in R_j)$ ならば $(u, w) \in R_{i+j}$ が成立することは、証明なしで用いてよい。
- (2-4) 小問 (2-3) の S について、 $v \in V$ の S による同値類 (equivalence class) を $[v]_S$ とする。 $[v]_S$ が、グラフ G について何を表すか簡潔に述べよ。

4 【選択問題】計算理論

(情報工学 8/15)

配点: (1-1) 5 点, (1-2) 5 点, (1-3) 5 点, (1-4) 25 点, (1-5) 20 点, (2-1) 25 点, (2-2) 20 点, (2-3) 20 点

- (1) 有限オートマトン (finite automaton) M は 5 項組 $M = (Q, \Sigma, \delta, q_0, F)$ で与えられる。ここで、 $Q, \Sigma, \delta, q_0, F$ は、それぞれ、状態 (state) の有限集合、入力記号 (input symbol) の有限集合 (アルファベット (alphabet))、状態遷移関数 (state transition function)、初期状態 (initial state) ($q_0 \in Q$)、受理状態 (accepting state) の集合 ($F \subseteq Q$) である。また、 M が受理 (accept) する言語 (language) (認識する言語) を $L(M)$ と表す。下の状態遷移図 (state transition diagram) に示す非決定性 (non-deterministic) 有限オートマトン M_1 について、以下の各小間に答えよ。なお、 $Q = \{a, b, c, d, e, f, g, h, i, j, k\}$, $\Sigma = \{0, 1\}$, $q_0 = a$, $F = \{k\}$ である。



- (1-1) M_1 が受理する言語 (language) を正規表現 (正則表現, regular expression) で示せ。
 (1-2) M_1 が受理する語 (word) を、1 文字目を最上位ビット (most significant bit) とする符号なし 2 進数 (unsigned binary number) とみなす。 M_1 が受理するすべての語の中で、7 番目に小さな数となる語を示せ。
 (1-3) M_1 の状態 i の ϵ -閉包 (ϵ -closure) を示せ。
 (1-4) $L(M_1) = L(M_2)$ を満たす、決定性 (deterministic) 有限オートマトン M_2 をサブセット構成 (subset construction) 法を用いて求め、状態遷移図で示せ。同値 (equivalent) な状態があっても全て残すこと。状態名は A, B, C, \dots とすること。導出過程と結果を示すこと。
 (1-5) M_2 の状態数を最小化した決定性有限オートマトン M_3 を求め、状態遷移図で示せ。状態名は A', B', C', \dots とすること。

(次ページに続く)

(2) 文脈自由文法(context-free grammar)により生成される文脈自由言語(context-free language)の閉包性(closure property)を考える。文脈自由文法 G は 4 項組 $G = (V, T, P, S)$ で表される。ただし、 V は変数(variable, あるいは非終端記号 non-terminal symbol)の集合、 T は終端記号(terminal symbol)の集合、 P は生成規則(production rule)の集合、 S は開始記号(start symbol)で $S \in V$ である。なお、生成規則の集合は ϵ -規則(ϵ -rule)を含んでよいものとし、変数の集合と終端記号の集合は共通の記号を含まないものと仮定する。文法 G が生成する言語を $L(G)$ と表記する。

任意の文脈自由文法 $G_1 = (V_1, T_1, P_1, S_1)$ と $G_2 = (V_2, T_2, P_2, S_2)$ が与えられる(ただし $V_1 \cap V_2 = \emptyset$ である)。

文脈自由言語は和(union)演算に関して閉じている。なぜなら、文脈自由文法 G_3 を以下の通りに構成すると、証明は省略するが、 $L(G_1)$ と $L(G_2)$ の和の言語 $L_3 = L(G_1) \cup L(G_2)$ は G_3 によって生成される。すなわち L_3 は文脈自由言語だからである。

- $G_3 = (V_3, T_3, P_3, S_3)$
- $V_3 = V_1 \cup V_2 \cup \{S_3\}$, ただし $S_3 \notin (V_1 \cup V_2)$
- $T_3 = T_1 \cup T_2$
- $P_3 = \{S_3 \rightarrow S_1, S_3 \rightarrow S_2\} \cup P_1 \cup P_2$

文脈自由言語は接続(concatenation)演算に関して閉じていることを、文法の構成により示したい。以下の各小間に答えよ。

(2-1) $L(G_1)$ と $L(G_2)$ を接続した言語 $L_4 = \{w_1 w_2 : w_1 \in L(G_1), w_2 \in L(G_2)\}$ を生成する文脈自由文法 $G_4 = (V_4, T_4, P_4, S_4)$, すなわち $L_4 = L(G_4)$ である G_4 を構成せよ。

(2-2) 上記(2-1)で構成した G_4 によって生成される任意の語 w は、上記(2-1)で定義した L_4 に属することを証明せよ。

(2-3) 上記(2-1)で定義した L_4 に属する任意の語 w は、上記(2-1)で構成した G_4 により生成されることを証明せよ。

5 【選択問題】ネットワーク

(情報工学 10/15)

配点: (1)15 点, (2-1)15 点, (2-2)25 点, (2-3)20 点,
(3-1)7 点, (3-2)10 点, (3-3)11 点, (3-4)12 点, (3-5)10 点

データリンク層 (data link layer) では、端末 (host) がフレーム (frame) を送信する際にプリアンブル (preamble) と呼ばれる特定のビット列を付与している。プリアンブルとフレームの構成は、図 1 に示すものとする。以下の各間に答えよ。

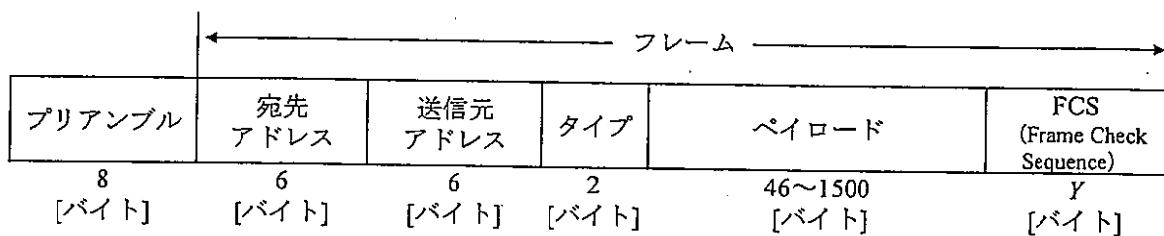


図 1 プリアンブルとフレーム構成

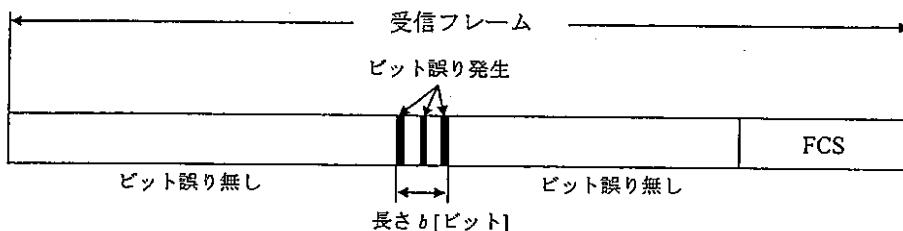
- (1) 伝送レート (transmission rate) X [bps] の伝送媒体 (transmission medium) を用いた場合に、上位層に対して提供できる最大の転送レート (transfer rate) を、 X および図 1 中の Y を用いて示せ。なお、インターフレームギャップ (inter-frame gap) は考えなくて良い。
- (2) プリアンブルに関する以下の各小間に答えよ。
 - (2-1) プリアンブルをフレームに付与する目的を述べよ。
 - (2-2) プリアンブルのビット列として最も適切なものを、以下の四つの選択肢から一つ選び、記号を答えよ。また、選んだビット列が、最も適切である理由を説明せよ。

選択肢 A : 00000111 00000110 00000101 00000100 00000011 00000010 00000001 00000000
 選択肢 B : 10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011
 選択肢 C : 11111111 11111111 11111111 11111111 00000000 00000000 00000000 00000000
 選択肢 D : 11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
 - (2-3) プリアンブルと同一のビット列がペイロード (payload) に含まれる場合に生じる問題を述べよ。また、その問題を回避するためにイーサーネット (Ethernet) で行われる方策を説明せよ。
- (3) 図 1 で与えられるフレームには、伝送誤り (transmission error) 検出を行うための FCS (frame check sequence) が含まれている。生成多項式 $G(x)$ を用いた巡回冗長検査 (cyclic redundancy check) によりフレームの伝送誤りが検出される。すなわち、生成多項式 $G(x)$ で定義される擬巡回符号 (pseudo-cyclic code) の符号語の冗長記号部分が FCS となり、誤り検出が行われる。以下の各小間に答えよ。ただし、生成多項式 $G(x)$ は次の多項式とする。

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

表1 $G(x)$ を生成多項式とする巡回符号の次数最小の符号語を表す多項式 $c(x)$

ハミング重み	$c(x)$
1	なし
2	なし
3	$1+x^{41678}+x^{91639}$
4	$1+x^{2215}+x^{2866}+x^{3006}$
5	$1+x^{89}+x^{117}+x^{155}+x^{300}$
6	$1+x^{79}+x^{85}+x^{123}+x^{186}+x^{203}$
7	$1+x^{45}+x^{53}+x^{74}+x^{80}+x^{120}+x^{123}$

図2 長さ b [ビット] のバースト誤りの例

- (3-1) FCS のサイズ Y [バイト] の値を答えよ.
- (3-2) 受信フレーム (誤り検出における受信語) の多項式表現を $R(x)$ とする. 巡回冗長検査において誤り無しと判断する条件式を, $R(x)$ と $G(x)$ を用いて書け.
- (3-3) 生成多項式 $G(x)$ を用いて検出できないビット誤り (bit error) の最小個数 r を考える. フレーム長が 1000 [バイト] である時の r を求めよ. 求めた過程も簡単に書け. 必要に応じて, $G(x)$ を生成多項式とする巡回符号の次数最小の符号語を表す多項式 (符号多項式) $c(x)$ をハミング重みごとにまとめた表1を用いてよい.
- (3-4) 図1のフレームの最大ペイロード長は 1500 [バイト] である. 最大ペイロード長をその 10 倍の 15000 [バイト] に増やすことの利点と欠点を, 巡回冗長検査の観点から簡潔に説明せよ.
- (3-5) 生成多項式 $G(x)$ を用いたバースト誤り (burst error) 検出に関する以下の文章中で, b にあてはまる数値と空欄 にあてはまる多項式を書け.

フレーム内で, 図2に示すような1つのバースト誤りが発生したとする. その長さが b [ビット] 以下であれば, 確実にそのバースト誤りを検出できる. しかし, 長さが $(b+1)$ [ビット] のときは, 検出できない場合がある. 例えば, 送信フレーム (符号語) を表す多項式が 0 と等しく小問(3-2)の多項式 $R(x)$ が と等しいときに, そのバースト誤りを検出できない.

配点: (1-1) 25 点, (1-2) 15 点, (1-3) 15 点, (2-1) 15 点, (2-2) 24 点, (2-3) 16 点, (3) 15 点

- (1) 二つの n ビット符号無し 2 進整数 (unsigned binary integer) $X = (x_{n-1}, \dots, x_0)$, $Y = (y_{n-1}, \dots, y_0)$ が入力され, $X > Y$ のとき出力 out に 1 を出力し, $X \leq Y$ のとき 0 を出力する比較器 (comparator) を設計する。1 ビットの比較器を図 1 のように直列 (serial) に接続した構造を考える。以下の各小間に答えよ。

- (1-1) 1 ビットの比較器は、各桁において x_i , y_i と下位桁の比較結果 c_i を入力して、比較結果 c_{i+1} を出力する。この 1 ビット比較器の真理値表 (truth table) を示せ。
- (1-2) x_i , y_i , c_i を変数とする c_{i+1} の最小積和形 (最簡積和形; minimal sum-of-products expression) を求めよ。
- (1-3) (1-1)(1-2)で考えた 1 ビット比較器を NAND ゲートと NOT ゲートのみを用いて実現せよ。

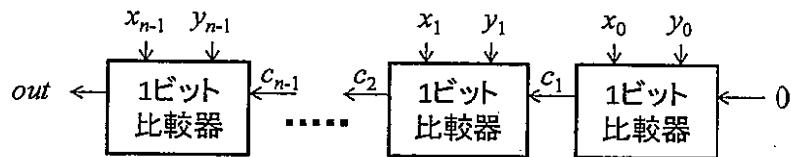


図 1

(2) エッジトリガ型 D フリップフロップ (edge-triggered D flip-flop) を 3 個用いた順序回路 (sequential circuit) について考える。3 個の D フリップフロップのそれぞれの入力を D_2, D_1, D_0 、出力を Q_2, Q_1, Q_0 とする。ここでは、 (Q_2, Q_1, Q_0) が $(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (0, 0, 0), \dots$ と遷移するカウンタ (counter) を設計する。 (Q_2, Q_1, Q_0) の初期値 (initial value) は $(0, 0, 0)$ とする。以下の各小間に答えよ。

- (2-1) カウンタの状態遷移表 (state transition table) を作成せよ。
- (2-2) D_2, D_1, D_0 を Q_2, Q_1, Q_0 の最小積和形で表せ。
- (2-3) (2-1)(2-2) で設計したカウンタにおいて、一時的な誤動作 (temporal error) により時刻 T にフリップフロップの記憶値が $(Q_2, Q_1, Q_0) = (1, 1, 0)$ となった。これ以降 3 クロックサイクルにおける (Q_2, Q_1, Q_0) の値の変化を示せ。但し、時刻 T 以外において順序回路は設計された通りに動作している。表 1 にならって解答すること。

表 1

時刻	(Q_2, Q_1, Q_0)
T	$(1, 1, 0)$
$T+1$	
$T+2$	
$T+3$	

- (3) 図 2 の CMOS 回路の出力 z を入力 a, b, c, d を用いた論理式で表せ。図中の V_{dd} は電源電位、 Gnd はグラウンド電位に接続されている。

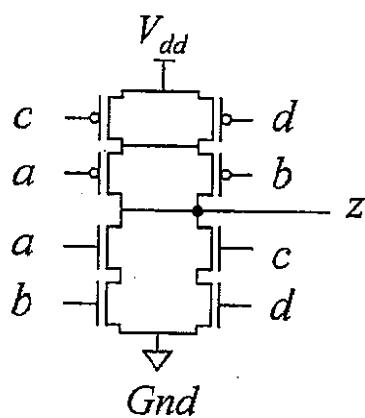


図 2

配点: (1-1)30 点, (1-2)25 点, (2-1)35 点, (2-2)35 点

以下の各間に答えよ。

(1) 以下の各小間に答えよ。

(1-1) ある関数 $f(t)$ のラプラス変換 (Laplace transform) を $\mathcal{L}[f(t)] = F(s)$ とし,

$$g(t) = \begin{cases} f(t-a) & t \geq a \\ 0 & t < a \end{cases}$$

であるとき, $\mathcal{L}[g(t)] = e^{-as}F(s)$ であることを証明せよ。ただし $a \geq 0$ とする。

(1-2) 以下の式のラプラス変換を求めよ。

$$h(t) = \begin{cases} \cos(t - 2\pi/3) & t \geq 2\pi/3 \\ 0 & t < 2\pi/3 \end{cases}$$

(次ページへ続く)

(2) 図1に示す周期 T である周期信号 (periodic signal) $x(t)$ に関する以下の各小間に答えよ。ただし n は整数 (integer number) である。また $0 < \tau < \frac{T}{2}$ である。

$$x(t) = \begin{cases} 0 & (n - 1/2)T < t \leq nT - \tau/2 \\ V & nT - \tau/2 < t \leq nT + \tau/2 \\ 0 & nT + \tau/2 < t \leq (n + 1/2)T \end{cases}$$

(2-1) $x(t)$ の複素型フーリエ級数 (complex form of Fourier series) を求めよ。

(2-2) $x(t)$ のパワー (power) P を、 $P \equiv \frac{1}{T} \int_{-T/2}^{T/2} |x(t)|^2 dt$ の定義に基づき計算すると、

$\frac{V^2\tau}{T}$ となる。これは、(2-1)で求めた複素型フーリエ級数を用いることでも求められる。これを以下の二つの式を用いて証明せよ。ただし、 c_k は $x(t)$ を複素型フーリエ級数表現したときの複素型フーリエ係数 (complex Fourier coefficient) とし ($k = 0, \pm 1, \pm 2, \dots$)、 $0 \leq a \leq \pi$ とする。

$$\frac{1}{T} \int_{-T/2}^{T/2} |x(t)|^2 dt = \sum_{k=-\infty}^{+\infty} |c_k|^2,$$

$$\sum_{k=1}^{+\infty} \frac{\sin^2(ka)}{k^2} = \frac{1}{2}a(\pi - a)$$

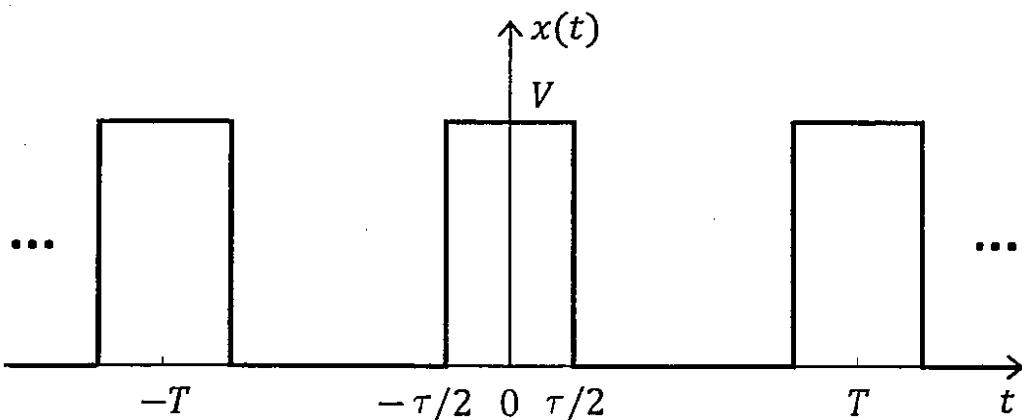


図1: 周期 T の信号 $x(t)$

【必須問題】アルゴリズムとプログラミング

(情報工学 1/15)

配点：(1-1) 20 点, (1-2) 20 点, (1-3) 25 点, (2-1) 32 点, (2-2) 28 点

(1)

図1に示すANSI-C準拠であるC言語のプログラム(program)は、重複のないN(自然数, $1 \leq N \leq 2000$ とする)個の整数を昇順に配列(array)Aの要素A[1]～A[N]に格納し、変数xの値が配列Aの要素に含まれているかどうかを2分探索法(binary search)を用いて探索するプログラムである。このプログラムは、探索の途中経過と、xの値が配列Aの要素に含まれる場合はその値が格納されている配列Aのインデックス(index)、xの値が配列Aの要素に含まれない場合は0を出力する。Nの値の設定、配列A[1]～A[N]に整数值を設定する処理、変数xの値を設定する処理、及び、これらの値が妥当かどうかを確認する処理は省略している。以下の各小間に答えよ。

(1-1) 図1のプログラムに対して、N, A[1]～A[N], xを下記のように設定した時の出力結果を書け。

N=8, A[1]=2, A[2]=5, A[3]=8, A[4]=10, A[5]=15, A[6]=20, A[7]=25, A[8]=30, x=8

(1-2) N=1000で、xの値がA[1]～A[N]のどこかに存在する場合、図1中のdo-whileループが実行される回数は最大何回か。その理由も簡潔に説明せよ。

(1-3) 図1のdo-whileループ中の□で囲われた部分を下記のように変更する。

変更前	変更後
<pre>if (A[mid]<x) left = mid+1; else right= mid-1;</pre>	<pre>if (A[mid]<x) left = mid; else right= mid;</pre>

変更前のプログラムは正しく結果が出力されるが、変更後のプログラムは正しく結果が出力されないことがある。そのようなN, A[1]～A[N], xの設定例を一つあげ、「正しく結果が出力されない」状況がどのようなものかを簡潔に説明せよ。

```
#include <stdio.h>  
/* この部分でNの値が設定される. */  
int main(void)  
{  
    int A[N+1];  
    int mid, left, right, x, index;  
    /* この部分に下記の処理が記述されているとする。  
     * 配列A[1]～A[N]の値が設定される。  
     * 変数xに探索すべき値が設定される。  
     * 配列A[1]～A[N], xの値が妥当かどうか確認される。  
    */  
    left = 1;  
    right = N;  
    do {  
        mid= (left+right)/2;  
        printf("%d %d %d\n", left, mid, right);  
        if (A[mid]<x) left = mid+1;  
        else right= mid-1;  
    } while (A[mid] != x && left <= right);  
    if (A[mid]==x) index=mid;  
    else index=0;  
    printf("%d\n", index);  
    return 0;  
}
```

図1 2分探索法のプログラム

- (2) 図 2 に示す ANSI-C 準拠である C 言語のプログラム (program) は、0-1 ナップサック問題 (0-1 knapsack problem) の解を求めて表示するプログラムである。なお、図の左端の数値は行番号を表す。0-1 ナップサック問題は、ある容量 (capacity) のナップサックが一つと、それぞれ大きさ (size) と価値 (value) が定められた複数の品物が与えられた時、ナップサックの容量を超えない範囲で価値の和が最大となる品物の組み合わせを求める問題である。本プログラムでは、四つの品物のそれぞれに他と重複しない番号が与えられており、番号 i ($1 \leq i \leq 4$) の品物の大きさと価値が、配列 size の要素 size[i] と配列 value の要素 value[i] としてそれぞれ格納されている。なお、各番号の品物はそれぞれ一つしかなく、また、分割できない。品物の大きさと価値は自然数とする。以下の各小間に答えよ。

(2-1) 22 行目の処理を実行する直前の配列 sack の内容を、解答用紙の表の空欄を埋めることにより答えよ。

sack[i][j]	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9	j=10	j=11	j=12	j=13	j=14	j=15
i=0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
i=1	0	-1	20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
i=2	0	-1	20	-1	-1	-1	30	-1	50	-1	-1	-1	-1	-1	-1	-1
i=3																
i=4																

(2-2) 図 2 のプログラム中の空欄 (ア), (イ) を適切な式 (expression) で埋めよ。

```

1 #include <stdio.h>
2 #define capacity 15      /* ナップサックの容量 */
3 #define item 4           /* 品物の個数 */
4 int main(void) {
5     int size[] = {0, 2, 6, 6, 2};
6     int value[] = {0, 20, 30, 15, 25};
7     int sack[item+1][capacity+1];
8     int i, j, max, index;
9
10    for (i = 0; i <= item; i++)
11        for (j = 0; j <= capacity; j++)
12            sack[i][j] = -1; /* ナップサックの初期化 */
13    sack[0][0] = 0;       /* 品物が入っていない(大きさの和が 0)のナップサックの総価値は 0 */
14
15    for (i = 1; i <= item; i++)
16        for (j = 0; j <= capacity; j++)
17            if (sack[i-1][j] != -1) {
18                if (sack[i-1][j] > sack[i][j]) sack[i][j] = sack[i-1][j];
19                if (j + size[i] <= capacity) sack[i][j+size[i]] = sack[i-1][j] + value[i];
20            }
21
22    max = 0; index = 0;
23    for (j = 0; j <= capacity; j++)
24        if (sack[item][j] > max) {max = sack[item][j]; index = j;}
25    for (i = item; i >= 1; i--)
26        if (index >= size[i] && (ア) == (イ)) {
27            printf("item %d is in a knapsack\n", i); index = index - size[i];
28        }
29    return 0;
30 }
```

図 2 0-1 ナップサック問題を解くプログラム

配点：(1-1) 5 点, (1-2) 40 点, (1-3) 15 点,

(2-1) 10 点, (2-2) 18 点, (2-3) 24 点, (2-4) 13 点

(1) 浮動小数点数 (floating point number) に関する以下の小間に答えよ。なお x を 2 進数 (binary number) として解釈する場合、 $[x]_2$ と表記する。角括弧 (square bracket) をつけない場合は 10 進数と解釈する。

(1-1) 0.625 を 2 進数として表記せよ。

(1-2) IEEE 754 で規定された半精度浮動小数点数表現 (representation of half precision floating point number) において、以下の値に最も近い値を表すビット列を解答用紙に書け。

- (a) $[1.101]_2$
- (b) 5
- (c) 0.125
- (d) 0.1

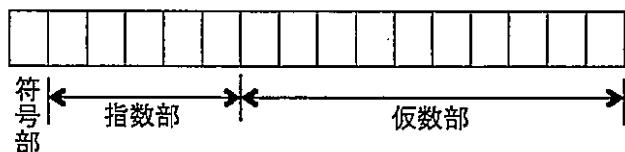
(1-3) 2 進数の浮動小数点数表現では、0.1 を誤差なく有限桁で表現できない理由を述べよ。

資料： IEEE 754 半精度浮動小数点数表現 簡略版 (simplified version)

符号部 (sign) を 1 ビット、指数部 (exponent) を 5 ビット、仮数部 (significand) を 10 ビットで表現。指数部が表す符号無し整数を e 、仮数部のビット列を f とすると、この形式が表す値は

$$(-1)^{\text{符号部}} \times 2^{e-15} \times [1.f]_2$$

である。ただし、 $e = 0, 31$ の時は上式をあてはめない非正規化数 (special value) として扱う。本問題では非正規化数を扱わないため、その説明を省略する。



図：IEEE 754 半精度浮動小数点数表現

(2) メモリ管理に関する以下の小間に答えよ。解答は全て解答欄の太線内に書くこと。

(2-1) 計算機 (computer) においては、読み書きの速度 (reading and writing speed)、容量 (capacity) が異なる複数種類の記憶素子 (memory device) や記憶装置 (memory unit) を連携動作させて、高速な読み書きと大容量の両立を図る、記憶階層 (memory hierarchy) の考え方が導入されている。以下に示す記憶素子または記憶装置 (ア)～(オ) を有する一般的な計算機において、これらを、読み書きが速く、容量の少ない順に整列し、記号で答えよ。

(ア) 主記憶装置 (main memory unit), (イ) 2 次キャッシュメモリ (secondary cache memory), (ウ) 補助記憶装置 (auxiliary memory unit), (エ) レジスタ (register), (オ) 1 次キャッシュメモリ (primary cache memory)

(2-2) 以下の文章の空欄 (a)～(i) に当てはまる最も適切な語句を、下記の選択肢から選び、記号で答えよ。同じ選択肢を複数回用いても良い。

記憶階層の一部を成す仮想記憶 (virtual memory) では、主記憶装置に置くべきプログラムやデータ (以下では“情報”と呼ぶ) の一部を (a) に置く。これにより、主記憶装置の持つアドレス空間よりもより広いアドレス空間を有するように、ユーザープログラムに見せかける効果がある。機械語命令 (machine instruction) の中に記載されるアドレスは (b) 、アドレスバスに送出されるアドレスは (c) と呼ばれる。

仮想記憶において、必要な情報が主記憶装置上に無く、(d) 上にある場合、この情報は主記憶装置に移動される。この動作は (e) と呼ばれる。この時、主記憶装置上に空き領域が無ければ、使用される可能性が低い情報が主記憶装置から (f) へ移動され、空き領域が作られる。この動作は (g) と呼ばれる。 (e) や (g) が頻繁に発生すると、計算機の処理性能が低下する。

仮想記憶の代表的な実現方法として、メモリを固定長 (fixed size) のブロック (block) で管理する (h) 方式と、可変長 (variable size) のブロックで管理する (i) 方式とがある。

【選択肢】

- | | | |
|--------------------|------------------------------|---------------------------------|
| (ア) レジスタ | (イ) スワップイン (swap in) | (ウ) フラグメンテーション (fragmentation) |
| (エ) キャッシュメモリ | (オ) 仮想アドレス (virtual address) | (エ) セグメンテーション (segmentation) |
| (キ) 補助記憶装置 | (ケ) スワップアウト (swap out) | (ケ) コンテクストスイッチ (context switch) |
| (コ) ページング (paging) | (サ) 実アドレス (real address) | (シ) 直接マッピング (direct mapping) |

(2-3) ページ枠 (page frame) 数が 4 のページング方式を採用した仮想記憶を考える。以下に示すページ番号 (仮想アドレス空間におけるページの番号) の順にメモリ参照が起きた時、ページ置換アルゴリズム (page replacement algorithm) が LRU (least recently used) 法と FIFO (first-in-first-out) 法のそれぞれについて、主記憶装置上に置かれる各ページ枠が保持するページの番号を答えよ。なお、初期状態では全てのページ枠は空であるとし、解答欄に示されている前半の例を参考に、後半の太線内について解答すること。また、ページフォルト (page fault) が発生する場合はページ番号を○で囲むこと。

参考ページ番号 : 0, 1, 2, 3, 3, 4, 2, 1, 0, 5, 1, 2

(2-4) ページング方式を採用した仮想記憶を有する計算機で、主記憶装置に入り切らないサイズの整数型配列 (array) A [W * H] を扱うプログラムを考える。以下に示す C 言語で書かれた二つのプログラム断片 (ア), (イ) の内、その処理時間が短くなる方を選び、記号で答えよ。また、その理由を仮想記憶と関連付けて説明せよ。なお、コンパイラ (compiler) による最適化は行われず、キャッシュメモリを有しない計算機で実行するものとする。

(ア)

```
for(x=0; x<W; x++) {
    for(y=0; y<H; y++) {
        sum = sum + x * A[x+(y*W)];
    }
}
```

(イ)

```
for(y=0; y<H; y++) {
    for(x=0; x<W; x++) {
        sum = sum + x * A[x+(y*W)];
    }
}
```

配点：(1-1-1)~(1-1-3) 各 12 点, (1-2-1) 10 点, (1-2-2) 10 点, (1-2-3) 15 点
 (2-1-1)~(2-1-3) 各 6 点, (2-2) 6 点, (2-3) 15 点, (2-4) 15 点

- (1) 一階述語論理 (first-order predicate logic) に関する以下の各小間に答えよ。ただし、一階述語論理式 (first-order predicate logic formula) の記述には以下の記号を用いる。 \forall, \exists はそれぞれ全称作用素 (universal quantifier), 存在作用素 (existential quantifier) であり、全称作用素と存在作用素を併せて限定作用素 (quantifier) と呼ぶ。 $\Leftrightarrow, \rightarrow, \wedge, \vee, \neg$ はそれぞれ等価 (equivalence), 合意 (implication), 論理積 (conjunction, and), 論理和 (disjunction, or), 否定 (negation, not) を表す論理演算子とする。特に断りのない限り、 u, w, x, y, z で変数 (variable) 記号、 a, b で定数 (constant) 記号、 f で関数 (function) 記号、 p, q で述語 (predicate) 記号を表わす。

- (1-1) 一階述語論理式の解釈 (interpretation) I は (D, C, F, P) の 4 項組で与えられる。ここで、 D は値集合、 C は各定数記号への D の要素への割り当て、 F は各 n 引数関数記号への $D^n \rightarrow D$ の要素の割り当て、 P は各 n 引数述語記号への $D^n \rightarrow B$ の要素の割り当てである。ただし、真 (true), 偽 (false) を表わす true, false の 2 値からなる集合を B とする。

例えば、一階述語論理式 $\forall x p(f(b, x), a)$ に対して、解釈 I_0 として

- D を非負整数 (nonnegative integer) 全体からなる集合とし、
- C として a, b それぞれへ非負整数值 0, 1 を割り当てる、
- F として 2 引数関数記号 $f(u, w)$ へ非負整数上の加算 $u + w$ を割り当てる、
- P として 2 引数述語関数 $p(u, w)$ へ非負整数上の比較演算 $u > w$ を割り当てるとき (例えば $4 > 3$ の値は true である)、

式 $\forall x p(f(b, x), a)$ の解釈 I_0 のもとでの評価値は true となる。

以下の (1-1-1)~(1-1-3) の各論理式が、(a) 恒真 (valid), (b) 充足可能 (satisfiable) であるが恒真ではない、(c) 充足不能 (unsatisfiable) のいずれであるか判断し記号で答えよ。また、(b) である場合は、 D を非負整数 (nonnegative integer) 全体からなる集合として、真にする解釈と偽にする解釈をそれぞれ一つずつ挙げよ。なお、定数記号や関数記号が含まれないものは、それぞれ C や F は考えなくてよい。

$$(1-1-1) (p(a) \wedge p(b)) \rightarrow \forall x p(x)$$

$$(1-1-2) \forall x (p(x) \vee q(x)) \rightarrow (\forall x p(x) \vee \forall x q(x))$$

$$(1-1-3) \neg \forall x \neg p(x) \Leftrightarrow \exists x p(x)$$

- (1-2) 以下で与えられる論理式 E が恒真 (valid) であることを示すため、まず、 E の否定をスコーレム化 (Skolemization) し、次いで、導出原理 (resolution principle) を適用し、その式が充足不能であることを示す。

$$A = \forall x \forall y (p(x, y) \rightarrow p(y, x))$$

$$B = \forall x \forall y \forall z ((p(x, y) \wedge p(y, z)) \rightarrow p(x, z))$$

$$C = \forall x \exists y p(x, y)$$

$$D = \forall z p(z, z)$$

$$E = (A \wedge B \wedge C) \rightarrow D$$

以下の (1-2-1)~(1-2-3) に答えよ。それぞれ導出過程も示すこと。なお、解答には記号 A, B, C, D を用いないこと。

- (1-2-1) $\neg E$ の冠頭標準形 (prenex normal form) を示せ。冠頭標準形は、すべての限定作用素が先頭にある閉論理式 (closed formula) である。ただし、閉論理式は連言標準形 (和積標準形 : conjunctive normal form) で示すこと。

- (1-2-2) (1-2-1) で得た論理式のスコーレム連言標準形 (Skolem conjunctive normal form) $\neg E'$ を求めよ.
 (1-2-3) (1-2-2) で得た論理式 $\neg E'$ をもとに、導出原理を用いて $\neg E'$ が充足不能であることを示せ.

- (2) 有限集合 (finite set) A について、 A の要素数を $|A|$ と書く。 A のべき集合 (power set) を $P(A)$ として、 $P(A)$ 上の 2 項関係 (binary relation) $R_{P(A)}$ を

$$R_{P(A)} = \{(a, b) \in P(A) \times P(A) \mid (a \subseteq b) \wedge (\forall c \in P(A))[(a \subseteq c) \wedge (c \subseteq b)] \rightarrow ((c = a) \vee (c = b))\}$$

とする。以下の各小間に答えよ。

- (2-1) 任意の有限集合 A について、 $R_{P(A)}$ の以下の各性質が成立するかどうか答えよ。証明は与えなくてもよい。

- (2-1-1) 反射性 (reflexivity)
- (2-1-2) 対称性 (symmetry)
- (2-1-3) 反対称性 (antisymmetry)

- (2-2) $|A| = 2$ の時、 $|R_{P(A)}|$ を求めよ。

- (2-3) $|A| = n$ の時、 $|R_{P(A)}|$ を求めよ。

- (2-4) $R_{P(A)}$ の反射的推移的閉包 (reflexive transitive closure) を $R_{P(A)}^*$ とする。 $|A| = n$ の時、 $|R_{P(A)}^*|$ を求めよ。

配点：(1-1) 10 点, (1-2) 15 点, (1-3) 20 点, (1-4) 15 点,
 (2-1) 25 点, (2-2) 40 点

- (1) 決定性有限オートマトン (deterministic finite automaton) M は 5 項組 $M = (Q, \Sigma, \delta, q_0, F)$ で与えられる。ここで、 $Q, \Sigma, \delta, q_0, F$ は、それぞれ、状態 (state) の有限集合、入力記号 (input symbol) の有限集合 (アルファベット (alphabet))、状態遷移関数 (state transition function)、初期状態 (initial state) ($q_0 \in Q$)、受理状態 (accepting state) の集合 ($F \subseteq Q$) である。また、 M が受理する言語 (認識する言語) を $L(M)$ と表す。有限オートマトンに関する以下の各小間に答えよ。

- (1-1) 決定性有限オートマトン $M = (Q, \Sigma, \delta, q_0, F)$ の状態 p, q ($p \in Q, q \in Q$) が区別不能 (indistinguishable) とは、任意の記号列 $w \in \Sigma^*$ に対し、 $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ が成り立つことであり、 $p \simeq q$ と表す。ここで、 Σ^* は、 Σ 上の記号列すべての集合 (空系列を含む) を表す。また、任意の状態 $r \in Q$ と任意の記号列 $x \in \Sigma^*$ に対し、 $\hat{\delta}(r, x) \in Q$ を以下のように定義する。

$$x = \varepsilon \text{ のとき, } \hat{\delta}(r, x) = r \quad (\varepsilon \text{ は空系列を表す})$$

$$x = ya \quad (y \in \Sigma^*, a \in \Sigma) \text{ のとき, } \hat{\delta}(r, x) = \delta(\hat{\delta}(r, y), a)$$

状態集合 Q 上の 2 項関係 (binary relation) \simeq が Q 上の同値関係 (equivalence relation) であることを証明せよ。

- (1-2) 決定性有限オートマトン $M_1 = (Q_1, \{0, 1\}, \delta_1, a, \{a, b\})$ が右の状態遷移表 (state transition table) で与えられたとき、状態集合 $Q_1 = \{a, b, c, d, e, f, g, h, i\}$ が同値関係 \simeq によって、どのような同値類 (equivalence class) に分割 (partition) されるか示せ。

- (1-3) $L(M_1) = L(M)$ となる決定性有限オートマトン M の中で、状態数最小のものを M_2 とする。 M_2 を状態遷移図 (state transition diagram) で示せ。状態遷移図では、初期状態、受理状態それが分かるように明示すること。

- (1-4) $L(M_1) = L(M)$ となる決定性有限オートマトン M の中で、(1-3) で示した M_2 の状態数が最小であることを証明せよ。

	0	1
a	e	d
b	f	d
c	d	f
d	i	g
e	g	i
f	g	h
g	c	b
h	a	c
i	b	c

- (2) 文脈自由言語 (context-free language) に対して次の反復補題 (Pumping Lemma) が知られている。

反復補題 (Pumping Lemma)

文脈自由言語 L に対し、ある非負整数 n が存在し、 L に属する n 以上の長さの任意の文字列 z に対して、ある部分文字列分解 $z = uvwxy$ が存在して、以下が成り立つ。

- (i) $|vwx| \leq n$
- (ii) $vx \neq \varepsilon$ ただし ε は空系列
- (iii) $i \geq 0$ なる任意の i に対して $uv^iwx^iy \in L$

以下の各小間に答えよ。解答用紙には①～⑦とそれらに対応する解を列挙すること。

- (2-1) 文脈自由言語 $L_A = \{a^m b^m \mid m \geq 5\}$ に対して反復補題が成り立つことを確かめたい。以下の文章の空白を埋め、 L_A に対し反復補題が成り立つことを示せ。空白①～⑦にはそれぞれ、指定された条件を満たす要素がある。例えば ④ 非負整数 に対応する要素は非負整数である 1, 40, 100 等である。

n として ④ 非負整数 を選ぶ。 $z = a^K b^K$ ただし、 $K \geq ④$ 非負整数 なる任意の文 z に対して $z = uvwxy$ なる部分文字列分解として $u = a^{K-1}$, $v = ⑤$ 記号列, $w = ⑥$ 記号列, $x = ⑦$ 記号列, $y = ⑧$ 記号列 を選ぶ。明らかに $|vwx| \leq n$ であり、 $vx \neq \varepsilon$, および $i \geq 0$ なる任意の i に対して ⑨ 記号列 $\in L_A$ が成り立つ。なお L_A を生成する文脈自由文法 (context-free grammar) は $(\{S_1, A_1\}, \{a, b\}, P_1, S_1)$ であり (表記の意味は下記参考を参照), ここで $P_1 = \{S_1 \rightarrow ⑩$ 記号列, $A_1 \rightarrow ab, ⑪$ 生成規則 } である。

- (2-2) 言語 $L_B = \{a^m b^m c^m \mid m \geq 0\}$ は文脈自由言語ではないことを反復補題を用いて証明したい。以下の証明の空白を埋めよ。空白①～⑦にはそれぞれ、(2-1) と同様指定された条件を満たす要素が入る。ただし、④, ⑤, ⑥ については終端記号の数量的性質に言及すること。また、⑦については使用した反復補題の条件を明示すること。証明

背理法 (帰謬法; proof by contradiction) で証明する。言語 L_B が文脈自由文法であると仮定する。このとき仮定より L_B に対して反復補題が成り立つ。以下、反復補題が成り立たないことを示し、矛盾を導く。

n として K という非負整数值を考える。以降の議論は K をパラメータとして考えているので、任意の値を K に与えても成立する。 L_B の定義より K を超える長さの文 z が存在する。具体的に $z = a^K b^K c^K$ を考える。 z を $uvwxy$ に分割する際、すべての考え得る分割パターンに対して、上記の矛盾を説明する必要がある。

場合 1: vwx が終端記号 c を含まない場合 :

このとき y は ⑦ 記号列 を部分文字列として含む。また反復補題の条件 (ii) が成り立つとすると、 v, x について以下が成り立つ。

④ v, x に関する条件

したがって $i = 0$ において L_B に含まれるはずの文 uwy を考えたとき、 uwy は L_B に属さない。なぜならば

① 証明

すなわち反復補題の条件 (iii) を満たすことはできず、反復補題を満たさない。

場合 2: vwx が ⑨ 終端記号に関する条件 場合 :

対象の対称性 (symmetry) より場合 1 と同様である。

場合 3: その他の場合、すなわち vwx が ⑩ 終端記号に関する条件 場合:

場合 3 はそもそも起こりえない。なぜならば

② 証明

いずれの場合であっても反復補題を満たさず、よって、矛盾を生じる。

以上の議論は仮定していた命題「言語 L_B は文脈自由文法である」が偽であることを示している。Q.E.D.

参考

文脈自由文法 (context-free grammar) G は 4 項組 $G = (N, T, P, S)$ で与えられる。ここで N, T, P, S は、それぞれ、非終端記号 (non-terminal symbol) 集合、終端記号 (terminal symbol) 集合、生成規則 (production rule) 集合、開始記号 (start symbol) である。

配点: (1-1)6 点, (1-2)8 点, (1-3-1)4 点, (1-3-2)22 点,
 (2-1)30 点, (2-2)20 点, (2-3)15 点, (2-4)20 点

(1) ビット誤り率 (bit error rate) p (ただし $0 < p < 1/2$) の 2 元対称通信路 (binary symmetric channel) における, 2 元符号 $C = \{00000, 11111\}$ を用いた通信を考える. C の各符号語は等確率で送信されるとする.

(1-1) 受信語における誤りの個数が 3 ビット, 4 ビット, 5 ビットになる確率を, それぞれ p を用いて表せ.

(1-2) 最尤復号法 (maximum likelihood decoding) を用いて (すなわち, 最大尤度基準に基づいて) 復号した場合の, 誤って復号する確率を, p を用いて表せ.

(1-3) 限界距離復号法 (bounded distance decoding) を用いて復号することを考える.

(1-3-1) 限界距離 d の最大値を求めよ.

(1-3-2) $p = 1/10$ とする. 誤って復号する確率を $1/1000$ 以下に抑えつつ, 正しく復号する確率を最大にするような限界距離 d を求めよ. また, そのときの, 正しく復号する確率を求めよ. 計算過程も示すこと.

(2) IP (Internet Protocol) の経路制御 (routing) に関する以下の各小間に答えよ.

(2-1) 経路制御プロトコルに関する以下の説明文の空欄 (あ) ~ (か) に当てはまる最も適切な語句を選択肢から選び, その番号を答えよ. なお, 異なる空欄には異なる語句が当てはまる.

IP アドレス (IP address) が割り当てられたホスト (host) がルーター (router) を介して接続される IP ネットワークにおいて, ルーターが経路情報 (routing information) を管理する手法として, (あ) と (い) がある. (あ) は, 経路情報をネットワークの各ルーターに手動で設定する手法である. これに対し, (い) は, 経路制御プロトコル (routing protocol) を用いてルーターが経路情報を交換および収集し, 経路を決定する手法であり, ホストやルーター等の機器の接続状況に変更が生じた際に経路情報が自動で更新される. インターネットの AS (Autonomous System) 内で用いられる経路制御プロトコルには, (う) 型プロトコルである OSPF (Open Shortest Path First) や, (え) 型プロトコルである RIP (Routing Information Protocol) がある. (う) 型プロトコルは, (え) 型プロトコルと比較して, (お) などの利点がある. 一方, AS 間で用いられる経路制御プロトコルとして (か) がある.

選択肢

- | | |
|--|---|
| (a) IS-IS protocol (Intermediate System to Intermediate System protocol) | (h) スタティックルーティング (static routing) |
| (b) コスト最小 (cost-minimized) | (i) 最適ルーティング (optimal routing) |
| (c) 距離ベクトル (distance vector) | (j) リンクステート (link state) |
| (d) ブロードキャスト (broadcast) | (k) プロトコルで交換される情報量が少ない |
| (e) RSVP (Resource reSerVation Protocol) | (l) 接続状況に変更が生じた際の経路収束時間が短い |
| (f) BGP (Border Gateway Protocol) | (m) ダイナミックルーティング (dynamic routing) |
| (g) IPIP (IP-within-IP encapsulation protocol) | (n) 経路計算量が少ない |
| | (o) HTTP (Hyper Text Transfer Protocol) |

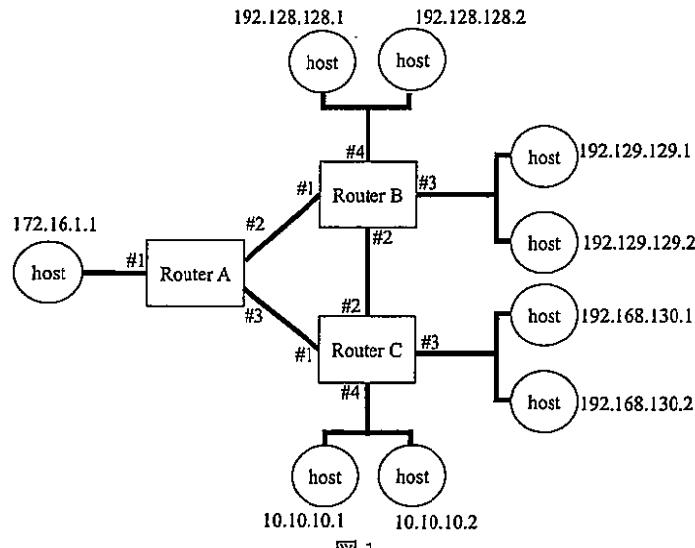


図 1

(2-2) 図 1 に示す IP ネットワークの構成を考える。図には、ホストの IP アドレスをドットアドレス表記で示している。また、#で示した番号はルーターのインターフェース番号である。クラスフルルーティング (classful routing) により経路情報を管理する場合に、ルーター A が保持する経路表 (routing table) を表 1 にならって示せ。経路表のエントリー数が最小となるように記載すること。ただし、ホスト間の経路は経由するルーター数が最小となるよう設定されるものとする。また、デフォルトゲートウェイは考えないものとし、図に記載のない IP アドレスに対する経路エントリーは保持しないものとする。必要に応じて、クラスフルルーティング (classful routing) とクラスレスルーティング (classless routing) に関する以下の説明文を参考にすること。

IPv4 における IP アドレス長は 32 ビットであるため、理論上約 40 億の機器に IP アドレスを割当てる事ができるが、IP アドレス毎に経路情報を保持すると経路情報が肥大化する。そのため、IP アドレスをネットワークアドレス (network address) 部とホストアドレス (host address) 部に分割し、ネットワークアドレス部に対する経路情報を管理する方法が採られている。IP の経路制御プロトコルが導入された当初は、ネットワークアドレス部の長さを固定とするアドレスクラス (address class) に基づいた経路制御プロトコルが用いられていた。例えば IP アドレスの最上位から 3 ビットを 110 とするアドレスクラス C の IP アドレスは、上位 24 ビットがネットワークアドレス部、下位 8 ビットはホストアドレス部と定められている。同様に、最上位ビットを 0 とするアドレスクラス A は上位 8 ビットがネットワークアドレス部であり、最上位ビットから 2 ビットを 10 とするアドレスクラス B は上位 16 ビットがネットワークアドレス部となる。このようなアドレス規則にもとづく経路制御をクラスフルルーティングと呼ぶ。最近ではアドレスクラスを撤廃し、サブネットマスク (subnet mask) を用いて指定される任意のネットワークアドレス部の長さにもとづいて経路を制御するクラスレスルーティングが導入されている。

表1

エントリー番号	宛先IPアドレス	サブネットマスク	出力先のインターフェース
1	172.16.0.0	255.255.0.0	#1
2			
3			
4			
5			
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

- (2-3) 図1のIPネットワークのルーターB, ルーターCに新たにインターフェース#5をそれぞれ追加し, ルーターBのインターフェース#5に H_B 台のホストを, ルーターCのインターフェース#5に H_C 台のホストを新たに接続することを考える。 H_B 台のホストに対して192.130.0で始まるアドレスクラスCのネットワークアドレスを新たに N_B 個割り当てた。また, H_C 台のホストに対しては, 192.1.0で始まるアドレスクラスCのネットワークアドレスを新たに N_C 個割り当てた。この時, 1) ホスト毎に経路表のエントリーを用意する場合, 2) クラスフルルーティングを行う場合, 3) クラスレスルーティングを行う場合のそれぞれについて, ルーターAが保持するホストに対する経路表の最小エントリー数を求めよ。必要に応じて H_B , H_C , N_B , N_C を用いること。(2-2)と同様に, ホスト間の経路は経由するルーター数が最小となるよう設定されるものとする。また, デフォルトゲートウェイは考えないものとする。
- (2-4) 図1のルーターCのインターフェース#1とインターフェース#2に対し, IPアドレス192.168.130.1宛のパケットが同時刻に到着する場合を考える。インターフェース#1に到着するパケットXと,インターフェース#2に到着するパケットYの二つのパケットが同時刻に到着してから出力先のインターフェースから送り出されるまでに, ルーターが到着パケットに対して行う処理を簡潔に説明せよ。ただし, 「経路表」, 「バッファ」(buffer), 「競合回避」(contention resolution)の語句を用いて説明すること。

配点：(1-1) 20 点, (1-2) 20 点, (1-3) 20 点, (1-4) 15 点, (2-1) 25 点, (2-2) 25 点

(1) 次の性質を持つ同期式カウンタ(synchronous counter)について、以下の各小間に答えよ。

1 個の入力(input) x と 6 個の状態(state)をもち、入力 $x = 0$ のときはクロックが入力される毎に状態が $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_0$ と遷移(transition)し、 $x = 1$ のときはクロックが入力される毎に状態が $S_0 \rightarrow S_2 \rightarrow S_4 \rightarrow S_0$ と遷移するカウンタを考える。ただし、初期状態(initial state)は S_0 とし、入力 $x = 1$ で状態が S_1, S_3, S_5 のいずれかのときは、それぞれ S_2, S_4, S_0 に遷移するものとする。本カウンタを図 1 に示すように 3 個の D フリップフロップ(D flip-flop)と論理ゲート(logic gate)を使って順序回路(sequential circuit)として実現する。D フリップフロップの出力を Q_i ($i = 0, 1, 2$) と表す。各状態は表 1 のように割り当てるものとする。また、入力 x はクロック CLK に同期して遅れなく入力されるものとし、初期化時にリセット信号 RST が入力されるとすべての D フリップフロップの出力 Q_i ($i = 0, 1, 2$) は 0 になるものとする。

(1-1) 本カウンタの状態遷移図(state transition diagram)を作成せよ。

(1-2) 状態 (Q_2, Q_1, Q_0) の次状態(next state)を (Q_2^+, Q_1^+, Q_0^+) で表す。状態割当付きの状態遷移表を作成し、 x, Q_2, Q_1, Q_0 を変数とする Q_2^+, Q_1^+, Q_0^+ の最簡積和形(最小積和形, minimal sum-of-products expression)の論理式(logic expression)を導出せよ。

(1-3) (1-2)で求めた最簡積和形の論理式を用いて、図 1 の順序回路のうち破線内の回路を最小のゲート数で実現せよ。解答用紙には、図 1 全体を記入すること。ただし、論理ゲートは、2 入力 NAND ゲートのみが使用できるものとする。最小であることの証明は不要である。

(1-4) (1-3)で設計した順序回路の最大動作周波数(maximum operating frequency)を求めよ。設計に用いた 2 入力 NAND ゲートの 1 段分の遅延時間(delay time)を T_v [s]、D フリップフロップのセットアップ時間(setup time)を T_s [s] とせよ。ただし、それ以外の時間は考慮しなくてもよいものとする。

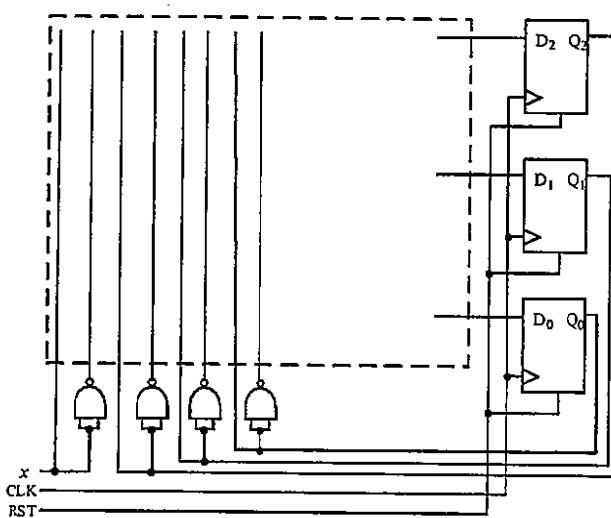


図 1

表 1

状態	状態割り当て		
	Q_2	Q_1	Q_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	1
S_3	1	1	1
S_4	1	1	0
S_5	1	0	0

(2) MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) に関する以下の各小間に答えよ。

(2-1) 図 2 は n-MOSFET の構造を示している。また、図 3 は n-MOSFET を用いた回路の例である。以下の文中の空欄(a)～(f)を適切な語句で埋めよ。

図中の(A)の部分は (a) 型半導体 (semiconductor) で、(B)の部分は (b) 型半導体でできている。ゲート (gate) 電極は酸化膜 (oxide) によって基盤 (substrate/bulk) とは絶縁されている。 V_g はゲート電圧、 V_d はドレイン (drain) 電圧である。 $V_g = 0$ のとき、ソース (source)・ドレイン間は絶縁され、電流が流れない。しかし、 V_g を (c) よりも大きくすると、基盤と酸化膜の境界に (d) が誘起され、(e) が形成されて電流が流れれるようになる。したがって、 V_g を制御することによって、オンオフ・スイッチとして動作させることができる。さらに、極性 (polarity) が逆の p-MOSFET と組み合わせて図 4 のような回路を構成すると、(f) の機能を持つ論理ゲートを実現できる。

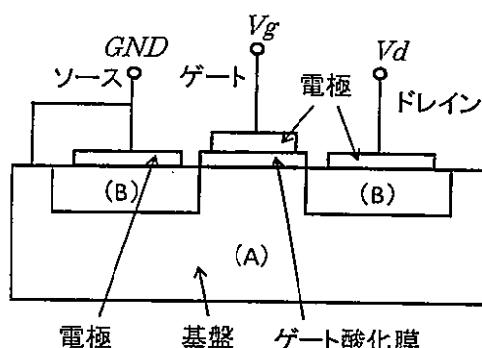


図 2

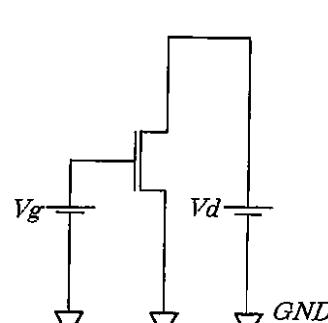


図 3

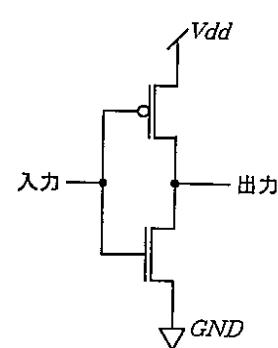


図 4

(2-2) 以下の図 5 および図 6 に示す回路それぞれについて、入力 A 及び B に電位 V_{dd} あるいは電位 0 が印加されたときの出力 X の電位を求めよ。解答は表 2 の形式で記せ。ただし、オン状態の MOSFET における電圧降下はないものとする。

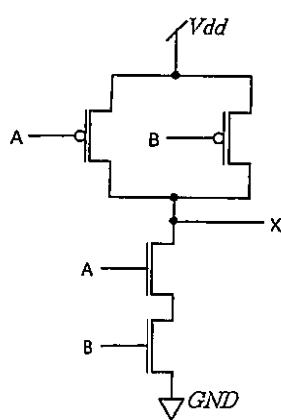


図 5

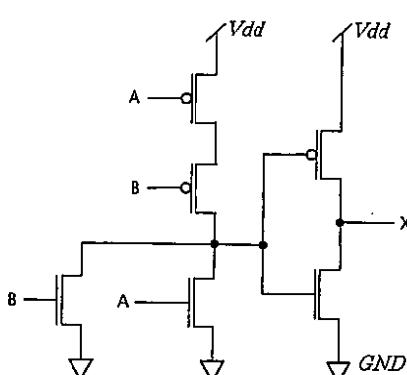


図 6

表 2

A	B	X
0	0	
0	V_{dd}	
V_{dd}	0	
V_{dd}	V_{dd}	

配点: (1-1)30 点, (1-2)35 点, (2)35 点, (3)25 点

以下の各間に答えよ。

(1)

(1-1) 以下のベルヌーイ型の微分方程式 (Bernoulli differential equation) に対して適当な変数変換 (variable transformation) を適用し、線形微分方程式 (linear differential equation) に変換せよ。ただし n は実数であり、 $n \neq 0, 1$ とする。

$$\frac{dx}{dt} + P(t)x = Q(t)x^n$$

(1-2) (1-1) の式において、 $P(t) = \frac{2}{t}$, $Q(t) = -t^2 \cos(t)$, $n = 2$ であるときの $x(t)$ を求めよ。

(2)

留数定理 (residue theorem) を用いて、以下の定積分 (definite integral) I を求めよ。

$$I = \int_{-\infty}^{\infty} \frac{dx}{x^2 + x + 1}$$

(次ページへ続く)

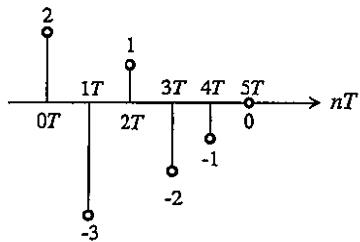
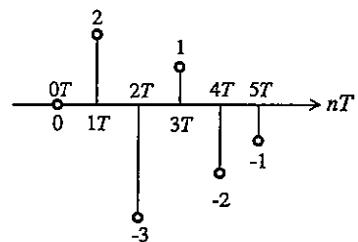
(3)

図1と図2に示すサンプリング周波数が $1/T$ である以下の離散時間信号(discrete-time signal) $x(nT)$ と $y(nT)$ を考える。

$$x(nT) = \{x(0T) = 2, x(1T) = -3, x(2T) = 1, x(3T) = -2, x(4T) = -1, x(5T) = 0\}$$

$$y(nT) = \{y(0T) = 0, y(1T) = 2, y(2T) = -3, y(3T) = 1, y(4T) = -2, y(5T) = -1\}$$

ただし、 n は整数であり、 $n < 0$ または $n > 5$ のとき $x(nT) = 0$, $y(nT) = 0$ とする。

図1: 離散時間信号 $x(nT)$ 図2: 離散時間信号 $y(nT)$

以下の文章の空欄 (a) から (e) を適切な語句または数式で埋めよ。ただし、解答用紙には (a) から (e) とそれらに対応する解の組を列挙すること。

図1と図2から、 $y(nT)$ は $x(nT)$ に対して (a) サンプル分だけ遅れている (delay) ことが分かる。一方、 $x(nT)$ と $y(nT)$ の Z 変換 (Z-transform) はそれぞれ、 $X(z) = (b)$, $Y(z) = (c)$ となる。この結果から、 $X(z)$ と $Y(z)$ には

$$Y(z) = (d) X(z)$$

の関係があることが分かる。 (a) サンプル分の遅延は、上式の (d) に対応しており、 k サンプル分の遅延は (e) の乗算に相当する。これは Z 変換の時間シフト (time shifting) に関する性質によるものである。

【必須問題】アルゴリズムとプログラミング

(情報工学 1/12)

配点：(1) 30 点, (2) 20 点, (3) 15 点, (4) 25 点, (5) 20 点, (6) 15 点

図 1 は ANSI-C 準拠である C 言語のプログラム (program) である。このプログラムにおいて、`insert` 関数はある特定の規則に従ってデータ列 (data sequence) に新しいデータ (data) を挿入 (insert) する関数 (function) であり、データ列を格納する配列 (array) A, 配列の大きさ SIZE, 挿入するデータ d を引数 (arguments) とする。`delete` 関数はデータ列からある特定の規則に従ってデータを一つ取り出し (retrieve), 削除 (delete) する関数であり、データ列を格納する配列 A を引数とし、取り出したデータを戻り値 (return value) とする。`main` 関数は、それらの二つの関数を実行する一例を示している。以下の各間に答えよ。

- (1) 図 1 のプログラムにおいて、43 行目および 46 行目の処理を実行した結果をそれぞれ示せ。
- (2) 図 1 のプログラムにおいて、43 行目の処理を実行した直後の変数 `front` および変数 `rear` の値をそれぞれ示せ。
- (3) 図 1 の `insert` 関数内の 11～21 行目では、新たに挿入するデータの挿入位置を決定している。この際、データ列を最初から一つずつ調べることなく、処理を効率化している。具体的にどのようなことをしているかを簡潔に説明せよ。
- (4) データ列のデータ数を n としたとき、`insert` 関数および `delete` 関数を実行する際の時間計算量 (time complexity) のオーダー (order) を示せ。その理由も簡潔に説明せよ。
- (5) 図 1 の `main` 関数において、47 行目以降に `insert` 関数および `delete` 関数を多数回実行した場合、データ列のデータ数に関わらず、実行途中でデータの挿入に失敗してしまう。その原因、および、データの挿入が失敗する条件を、データ挿入回数の観点から簡潔に説明せよ。
- (6) 上記(5)の問題を解決するために、配列を十分大きくする、および、データの挿入回数に制限を設ける以外に、どのような方法があるか、簡潔に説明せよ。ただし、その方法が `insert` 関数および `delete` 関数の時間計算量のオーダーを変えることがあってはならない。また、できるだけ時間計算量の増加が少ない方法を示すこと。なお、その方法を実現するために、関数に新たな引数を追加することがあってもよいが、新たな関数は追加してはならない。

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int front = 0;
5 int rear = 0;
6
7 void insert(int A[], int SIZE, int d) {
8     int p, left, right, m, i;
9     if (rear > SIZE-1) { printf ("Overflow !!\n"); exit(1); }
10
11    p = -1;
12    if (front == rear) p = front;
13    else {
14        left = front; right = rear-1;
15        while (left < right) {
16            m = (left+right)/2;
17            if (A[m] == d) { p = m+1; break; }
18            if (d < A[m]) right = m-1; else left = m+1;
19        }
20        if (p == -1) { if (A[left] > d) p = left; else p = left+1; }
21    }
22
23    i = rear;
24    while (i > p) { A[i] = A[i-1]; i--; }
25    A[p] = d; rear++;
26}
27
28 int delete(int A[]) {
29     int x;
30     if (front == rear) { printf ("Underflow !!\n"); exit(1); }
31     x = A[front]; front++;
32     return x;
33 }
34
35 int main () {
36     int i;
37     int A[20]; int SIZE = 20;
38
39     int a1[5] = {10, 5, 20, 6, 13};
40     int a2[5] = {2, 5, 18, 7, 5};
41
42     for (i = 0; i < 5; i++) insert(A, SIZE, a1[i]);
43     for (i = 0; i < 3; i++) printf("%d ", delete(A)); printf ("\n");
44
45     for (i = 0; i < 5; i++) insert(A, SIZE, a2[i]);
46     for (i = 0; i < 7; i++) printf("%d ", delete(A)); printf ("\n");
47
48     return 0;
49 }
```

図1：プログラム

配点：(1-1) 36 点, (1-2) 8 点, (1-3) 8 点, (1-4) 18 点, (2-1) 20 点, (2-2) 35 点

(1) 計算機の演算器に関する以下の小間に答えよ。解答は全て解答欄に書くこと。

(1-1) 以下に示す仕様の計算機および C 言語コンパイラを考える。

- 演算器 (ALU: arithmetic logic unit), レジスタ, メモリの語 (word) 長は全て 8 ビット。
- int 型は 8 ビット符号付き (signed) 整数 (integer) で, 2 の補数系 (2's complement system) 表現。
- 整数演算時にはオーバーフローは無視され, 演算結果がそのまま格納される。
- float 型は 8 ビット浮動小数点数 (floating point number). 表現したい数を $(-1)^s \times (1+m) \times 2^{(e-3)}$ の形式で表し, 最上位ビット (MSB: most significant bit) から s, e, m の順にメモリに格納する。 s は 1 ビット, e は 3 ビット, m は 4 ビットである。仮数部 (significand) は正規化 (normalize) ($1 \leq 1+m < 2$) されており, 整数部 (integer part) の 1 はメモリには格納されない (隠しひつ (hidden bit))。仮数部の 2 進数表現の小数部 (fractional part) 上位 4 衔をメモリに格納する。5 衔目以降は切り捨て (round down) により丸める。

この計算機および C 言語コンパイラを用いて, 図 1 に示すプログラムをコンパイル, 実行した。return 文の時点での各変数 (variable) の値を 10 進数で示せ。また, それらを格納するメモリの内容をビット列 (bit string) で示せ。なお, float 型変数の値は, 指数表記 (scientific notation) を用いずに表すこと。また, メモリの内容 (ビット列) は, 左端を最上位ビットとして示すこと。

(1-2) 計算機において, 符号付き整数の表現に 2 の補数系表現を用いることの利点を述べよ。

(1-3) 入力として 8 ビット符号なし (unsigned) 整数 y (ビット列: $y_7 y_6 \dots y_0$) と 1 ビット制御信号 (control signal) w が与えられたとき, 出力 y' (ビット列: $y'_7 y'_6 \dots y'_0$) として, $w = 0$ ならば y のものを, $w = 1$ ならば y の 1 の補数 (1's complement) を出力する回路を作ることを考える。この回路は, 同じ構成の回路を 8 ビット分並べて構成することができる。第 i ビット ($i \in [0, 7]$) 1 ビット分の回路を, 図 2 に示す排他的論理和 (exclusive OR) を用いて構成し, その回路図を示せ。なお, 排他的論理和の論理式は $f = g \bar{h} + \bar{g} h$ である。必要ならば NOT 回路も用いてよい。

(1-4) 図 3 に示す 8 ビット符号なし整数加算器を用いて, 2 の補数系表現による 8 ビット符号付き整数の加算ならびに減算を行う演算器を作ることを考える。被演算数 (operand) x (ビット列: $x_7 x_6 \dots x_0$), y (ビット列: $y_7 y_6 \dots y_0$), 1 ビットの演算制御信号 w が与えられ, 演算結果 z (ビット列: $z_7 z_6 \dots z_0$) が outputされる。 $w = 0$ ならば $z = x + y$, $w = 1$ ならば $z = x - y$ となる。この回路を, 排他的論理和と 8 ビット符号無し整数加算器を用いて構成し, その回路図を書け。なお, 必要ならば NOT 回路も用いてよい。

```
int main()
{
    int i, j, k, m, n;
    float d, e;

    i = 90;
    j = -40;
    k = i + j;
    m = i - j;
    n = j - i;
    d = 0.4;
    e = d + 0.8;

    return 0;
}
```

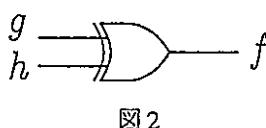


図 2

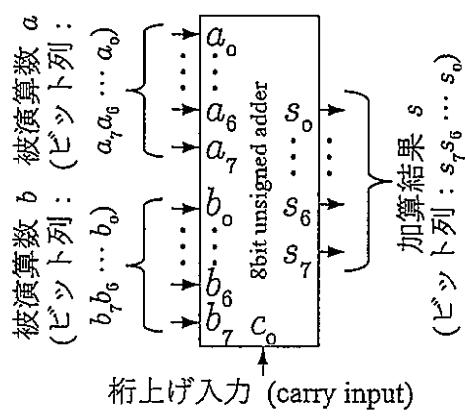


図 3

図 1

(2) 排他制御(mutual exclusion)に関する以下の小間に答えよ.

(2-1) 以下の空欄 ~ を適切な用語で埋めよ.

共有資源(shared resource)の排他制御を実現する方法としてテストアンドセット(test and set)やセマフォ(semaphore)等がある。テストアンドセットは一つしかない共有資源の排他制御を実現する機械語命令(machine instruction)である。これは、「共有資源の 状態と 状態とを監視し制御する」機能を不可分操作(atomic operation)として実行する命令である。不可分操作として実行されるのは以下の二つである。

- 1) テスト：共有資源が 状態か 状態かをチェックする。 状態の場合、終了する。
- 2) セット：共有資源を 状態に変更する。
ここでテストとセットは不可分操作のため、 が発生しない。

(2-2) テストとセットを不可分操作とする必要性を、不可分操作でなかった場合に排他制御が失敗する状況を例示することにより明らかにせよ。なお回答では以下の状況を仮定せよ。

- ・ 二つのプロセス X と Y が一つしかない共有資源 Z を取り合う。
- ・ プロセス X が先にテストを実行する。

配点:(1-1) 20 点 (1-2) 20 点 (1-3) 15 点 (2-1) 15 点 (2-2) 20 点 (2-3-1) 15 点 (2-3-2) 20 点

- (1) 任意の空でない有向グラフ (directed graph) $G = (V, E)$ ($E \subseteq V \times V$) について, $V \times V$ 上の以下の二項関係 (binary relation) R_i ($i = 1, \dots, 5$) を考える. なお, $S = \{(x, x) | x \in V\}$ である. 有向経路 (directed path) および有向閉路 (directed cycle) はいずれも, 異なる頂点を含むものとする. R_3 に関しては, ある $s \in V$ が存在し, s から他のすべての頂点への有向経路が存在するような G のみを対象とする. 以下の各小間に答えよ.

$$R_1 = \{(x, y) | x \text{ から } y \text{ への有向経路が存在する}\} \cup S$$

$$R_2 = \{(x, y) | x \text{ と } y \text{ を共に含む有向閉路が存在する}\} \cup S$$

$$R_3 = \{(x, y) | s \text{ から } y \text{ へのすべての有向経路が } x \text{ を含む}\}$$

$$R_4 = \{(x, y) | (x, y) \in E\} \cup S$$

$$R_5 = \{(x, y) | (\exists z \in V) [(x, z) \in E \wedge (y, z) \in E]\} \cup S$$

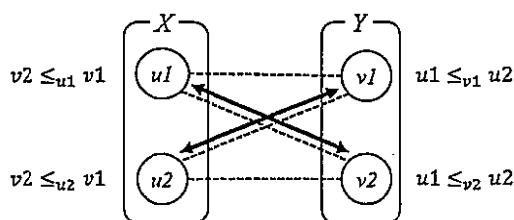
(1-1) 同値関係 (equivalence relation) である R_i をすべて挙げ, それぞれ同値関係であることを証明せよ.

(1-2) 半順序関係 (partial order relation) である R_i をすべて挙げ, それぞれ半順序関係であることを証明せよ.

(1-3) グラフ G を有向閉路のない有向グラフに限定した場合に, 半順序関係である R_i をすべて挙げよ.

- (2) 論理記号 \Leftrightarrow , \rightarrow , \wedge , \vee , \neg はそれぞれ等価 (equivalence), 合意 (implication), 論理積 (conjunction, and), 論理和 (disjunction, or), 否定 (negation, not) を表す. $\wedge_{z \in Z} p(z)$ で集合 Z に属する値 z を引数に持つ論理式 $p(z)$ をすべて論理積で結合した論理式を, $\vee_{z \in Z} p(z)$ でそのような $p(z)$ をすべて論理和で結合した論理式を表す.

完全2部グラフ (complete bipartite graph) $G = (X \cup Y, X \times Y)$ で $|X| = |Y|$, $X \cap Y = \emptyset$ とする. 各頂点 x ($x \in X$) に対して Y の頂点間の全順序関係 (total order relation) \leq_x が与えられ, かつ各頂点 y ($y \in Y$) に対して X の頂点間の全順序関係 \leq_y が与えられるとする. このとき, $\forall (x, y) \in ((X \times Y) \setminus M) (\exists (a, b) \in M (x \leq_y a \vee y \leq_x b))$ であるようなマッチング (一対一対応, matching) M は安定 (stable) であるという. 下図は安定なマッチングの例である (破線 (dashed line) が辺を, 矢印 (arrow) がマッチングをそれぞれ表す).



安定なマッチングを得るあるアルゴリズム A が満たす性質を, 以下の述語 $r(h, x, y)$ および $m(h, x, y)$ を用いて表現することを考える. なお, h , x および y の対象領域をそれぞれ非負整数, X および Y とする. A はラウンドと呼ばれる時間単位で動作し, 0 を初期ラウンドとする.

$$r(h, x, y) \Leftrightarrow \text{ラウンド } h \text{ において, } x \text{ が } y \text{ にマッチング要求をする}$$

$$m(h, x, y) \Leftrightarrow \text{ラウンド } h \text{ において, } (x, y) \text{ はマッチングの要素である}$$

A は以下の 1.~3. にしたがって動作する.

- ラウンド 0 ではどの頂点もマッチング要求をせず, かつどの辺もマッチングの要素でない. これらは以下の論理式 $CZ1$ および $CZ2$ で表せる.

$$CZ1: \forall x \forall y \neg r(0, x, y)$$

$$CZ2: \forall x \forall y \neg m(0, x, y)$$

2. 各ラウンド h ($h \geq 1$) ではある条件を満たす X の頂点がマッチング要求をする。これに関する性質記述として論理式 $CX1 \sim CX5$ を与える。例えば $CX1$ は各ラウンドにおいて x のマッチング要求はただ一つだけであることを表している。なお、ラウンドに対する述語 $<$ ならびに二項演算子 $-$ の解釈は整数上のそれに従い、 $n(x, x')$ は二つの頂点 x, x' が異なるときかつそのときに限り真となる述語である。

- $CX1: \forall h \forall x \forall y \forall w ((r(h, x, y) \wedge n(y, w)) \rightarrow \neg r(h, x, w))$
- $CX2: \forall h \forall k \forall x \forall y ((r(h, x, y) \wedge k < h) \rightarrow \neg r(k, x, y))$
- $CX3: \forall h \forall x ((\bigwedge_{y \in Y} \neg m(h-1, x, y)) \rightarrow \bigvee_{y \in Y} r(h, x, y))$
- $CX4: \forall h \forall k \forall x \forall y \forall w ((r(h, x, y) \wedge r(k, x, w) \wedge n(y, w) \wedge k < h) \rightarrow y \leq_x w)$
- $CX5: \forall h \forall k \forall x \forall y \forall w ((r(h, x, y) \wedge \neg r(k, x, w) \wedge n(y, w) \wedge k < h) \rightarrow w \leq_x y)$

3. ラウンド h のマッチング要求は Y の頂点ごとに処理される。その性質記述として論理式 $CY1 \sim CY5$ を与える。

- $CY1: \forall h \forall x \forall y \forall z ((r(h, x, y) \wedge m(h-1, z, y) \wedge x \leq_y z) \rightarrow \neg m(h, x, y))$
- $CY2: \forall h \forall x \forall y \forall z ((r(h, x, y) \wedge r(h, z, y) \wedge x \leq_y z) \rightarrow \neg m(h, x, y))$
- $CY3: \forall h \forall x \forall y (\boxed{(a)})$
- $CY4: \forall h \forall x \forall y (\boxed{(b)})$
- $CY5: \forall h \forall x \forall y (\boxed{(c)})$

(2-1) $CY3 \sim CY5$ が以下を表すように、空欄 (a)～(c) の論理式を埋めよ。 \exists および \forall は利用しないこと。

$CY3$: 任意の (x, y) とラウンド h について、 h において x が y にマッチング要求するならば、 h において y を含む辺の少なくとも一つはマッチングの要素である

$CY4$: 任意の (x, y) とラウンド h について、 $h-1$ において (x, y) がマッチングの要素ならば、 h において y を含む辺の少なくとも一つはマッチングの要素である

$CY5$: 任意の (x, y) とラウンド h について、 h において (x, y) がマッチングの要素であれば、ラウンド $h-1$ において (x, y) はマッチングの要素であるか、あるいは h において x が y にマッチング要求する

(2-2) 「 A のもとで得られるマッチングが、あるラウンドにおいて安定であること」を、ある論理式 P が充足不能であることを示すことにより示したい。 P を、 $CX1 \sim CX5$, $CY1 \sim CY5$, $CZ1 \sim CZ2$, および、あるラウンド h におけるマッチングが安定であることを表す閉論理式 H を用いて表せ。

(2-3) $X = \{u1, u2\}$, $Y = \{v1, v2\}$ とし、 $v2 \leq_{u1} v1, v2 \leq_{u2} v1, u1 \leq_{v1} u2, u1 \leq_{v2} u2$ とする。

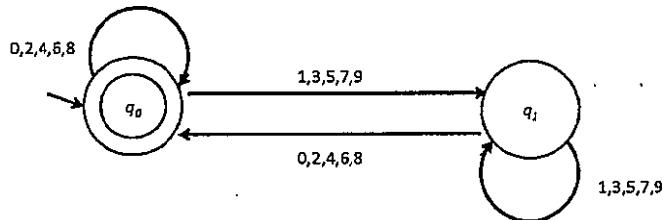
(2-3-1) 以下の空欄 (d)～(h) を、述語の引数に指定された値を代入した命題あるいはその論理和の形の論理式で埋めよ。 $CZ1$ と $CZ2$ 、および $n(x, x')$, \leq_z ($z \in X \cup Y$), $<$, $-$ の解釈のもとで偽である命題は除くこと。

- | | |
|--|---|
| $CX3$ で $(h, x) \leftarrow (1, u1)$ とすると | $A1: r(1, u1, v1) \vee r(1, u1, v2)$ |
| $CX3$ で $(h, x) \leftarrow (1, u2)$ とすると | $A2: r(1, u2, v1) \vee r(1, u2, v2)$ |
| $CX5$ で $(h, k, x, y, w) \leftarrow (1, 0, u1, v2, v1)$ とすると | $A3: \boxed{(d)}$ |
| $CX5$ で $(h, k, x, y, w) \leftarrow (1, 0, u2, v2, v1)$ とすると | $A4: \boxed{(e)}$ |
| $CY2$ で $(h, x, y, z) \leftarrow (1, u1, v1, u2)$ とすると | $A5: \neg r(1, u1, v1) \vee \neg r(1, u2, v1) \vee \neg m(1, u1, v1)$ |
| $CY3$ で $(h, x, y) \leftarrow (1, u1, v1)$ とすると | $A6: \boxed{(f)}$ |
| $CY5$ で $(h, x, y) \leftarrow (1, u1, v2)$ とすると | $A7: \boxed{(g)}$ |
| $CY5$ で $(h, x, y) \leftarrow (1, u2, v2)$ とすると | $A8: \boxed{(h)}$ |

(2-3-2) 上記 (2-3-1) の制約のもとでは、ラウンド 1 の終了時に $(u2, v1)$ のみがマッチングの要素であることを、導出原理 (resolution principle) を用いて証明せよ。論駁過程 (refutation process) も示すこと。

配点:(1-1) 25 点, (1-2) 25 点, (1-3) 10 点, (2-1) 15 点, (2-2) 30 点, (2-3) 20 点

- (1) 有限オートマトン(finite automaton)は5項組 $(Q, \Sigma, \delta, q_0, F)$ で与えられる。ここで $Q, \Sigma, \delta, q_0, F$ は、それぞれ、状態(state)の有限集合、入力記号(input symbol)の有限集合であるアルファベット(alphabet)、遷移関数、開始状態(initial state) ($q_0 \in Q$)、受理状態集合($F \subseteq Q$)である。有限オートマトンに関する以下の各小間に答えよ。アルファベット Σ を $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ とする。以下の各小問では Σ 上の語(word)を先頭を上位桁とする10進数の非負整数とみなす。通常の非負整数表現以外の語(例えば00045)についての動作は考慮しなくて良い。次に与える有限オートマトン A は与えられた語が2で割り切れるときのみ受理(accept)するオートマトンである。



- (1-1) 与えられた語が3で割り切れるときのみ受理する有限オートマトン B の構築の方法を説明せよ。説明にあたってはオートマトンの図を用いててもよい。
- (1-2) 小問(1-1)の構築方法で得られる有限オートマトン B を用いて5で割り切れるときのみ受理する有限オートマトン D を構築したい。まず与えられた語が5で割り切れるときのみ受理する有限オートマトン C を示せ。ついでこの2つの有限オートマトン B と C をどのように用いればオートマトン D を構築できるかを説明せよ。
- (1-3) 小問(1-2)の構築方法で得られる有限オートマトン D を少し変更し、3または5で割り切れるときのみ受理する有限オートマトン E を構築するにはどのようにすればよいか、変更の方法を説明せよ。
- (2) 文脈自由文法(context-free grammar) G は4項組 (N, T, P, S) で与えられる。ここで N, T, P, S は、それぞれ、非終端記号(non-terminal symbols)集合、終端記号(terminal symbols)集合、生成規則(production rules)集合、開始記号(start symbol)である。文脈自由文法 G の生成文全体を表す言語を $L(G)$ とする。この言語は一般に文脈自由言語(context-free language)と呼ばれる。これらに関する以下の各小間に答えよ。
- (2-1) 文脈自由言語 $L_1 = \{c^m \mid m \geq 1\}$ を生成する文脈自由文法 G_1 を示せ。
- (2-2) 言語 $L_2 = \{a^n b^n c^m \mid n \geq 1, m \geq 1\}$ および $L_3 = \{a^m b^n c^n \mid n \geq 1, m \geq 1\}$ が文脈自由言語であることを証明せよ。言語 L_2, L_3 を生成する文法を示して証明する場合は、それらの生成言語がそれぞれ L_2, L_3 と等価になることを証明すること。なお必要であれば小問(2-1)の結果と以下の補題1(lemma)を証明なしに用いてよい。
[補題1]
 文脈自由言語全体の集合は連接演算・について閉じている。すなわち、任意の2つの文脈自由言語 L_α と L_β に対して言語 $L_\alpha \cdot L_\beta = \{xy \mid x \in L_\alpha, y \in L_\beta\}$ も文脈自由言語である。
- (2-3) 文脈自由言語全体の集合が積演算 \cap について閉じていないことを証明せよ。必要であれば小問(2-2)の結果と以下の補題2を証明なしに用いてよい。
[補題2]
 言語 $L_4 = \{a^n b^n c^n \mid n \geq 1\}$ は文脈自由言語ではない。

5 【選択問題】 ネットワーク

(情報工学 8/12)

配点: (1-1)6 点, (1-2)8 点, (1-3-1)12 点, (1-3-2)14 点,
(2-1)20 点, (2-2)10 点, (2-3)10 点,
(3-1)20 点, (3-2)10 点, (3-3)15 点

(1) 記憶がない情報源 (memoryless information source) の符号化について以下の小間に答えよ.

(1-1) 記憶がない情報源とは、どのような情報源か説明せよ.

(1-2) 以下の文中的空欄 (あ) ~ (え) それぞれに適当な語を選択肢から一つ選び、その記号を答えよ.

(あ) が最小となる符号化を最適符号化 (または、コンパクト符号化) (optimum coding, compact coding) という。最適符号化を求める方法として (い) のアルゴリズムが知られている。また、(あ) の上界と下界を (う) を用いて与える (え) の情報源符号化定理が知られている。

選択肢 : (a) クラフト (Kraft) (b) シャノン (Shannon) (c) ハフマン (Huffman) (d) ハミング (Hamming)
(e) エントロピー (entropy) (f) 最大符号語長 (maximum codeword length)
(g) 平均符号語長 (average codeword length) (h) 符号化率 (code rate)

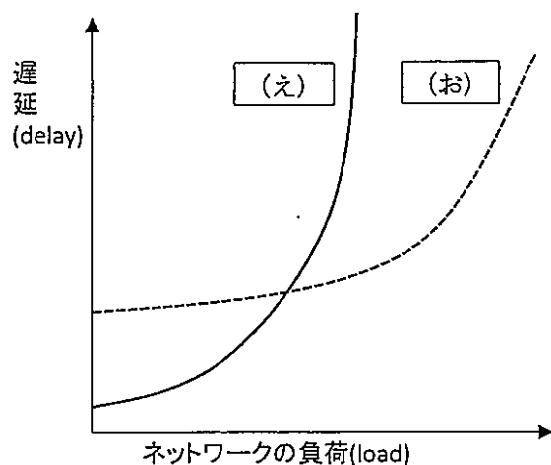
(1-3) 情報源記号が s 個の記憶がない情報源を考える。各記号の生起確率 p_1, p_2, \dots, p_s は、 $p_1 > p_2 > \dots > p_s > 0$ を満たすとする。小問 (1-2) の文中的最適符号化を求めるアルゴリズムによって得られる最適符号化について考える。

(1-3-1) $s = 4$ の場合に、符号語の長さがすべて等しくなる 2 元最適符号化が存在するための必要十分条件 (記号の生起確率についての条件) を書け。

(1-3-2) $s = 5$ の場合に、5 個のうち 4 個の符号語の長さが等しくなる 2 元最適符号化が存在するための必要十分条件 (記号の生起確率についての条件) を書け。また、それが必要十分条件となることを説明せよ。

(2) メディアアクセス制御プロトコル(media access control protocol)に関する説明文を読み、以下の各小間に答えよ。

バス型のネットワーク(network)では、ネットワークを効率良く使用するため、(あ) 方式や(い) 方式のメディアアクセス制御プロトコルが開発されている。 (あ) 方式は、送信ホスト(host)間でフレーム(frame)を送信する権利を順にまわしていくものであり、一方、(い) 方式は、送信ホスト間で調整を行わずにフレームを送信する。(い) 方式の一つである CSMA/CD(Carrier Sense Multiple Access / Collision Detection)では、送信ホストは送信前に(う) を行い、他のホストがフレームを送信していないことを確認する。これらの方では、送信ホスト数が増加してネットワークの負荷(load)が高まるにつれて、送信ホストから受信ホストにフレームが到達するまでの遅延(delay)が増加する。下記のグラフでは、ネットワークの負荷を変化させた時の遅延の変化を模式的に示している。



(2-1) 説明文の (あ) ~ (お) に当てはまる最も適切な語句を選択肢から一つ選び、その記号を答えよ。ここで、(え) 及び(お) にはそれぞれ、(あ)、(い) で選択したいずれかの選択肢が入る。

選択肢：

- | | |
|----------------------------------|-----------------------------|
| (a) ポイント・ツー・ポイント(point-to-point) | (b) コネクション(connection) |
| (c) コネクションレス(connectionless) | (d) ランダムアクセス(random access) |
| (e) トークンパッシング(token passing) | (f) 搬送波検知(carrier sense) |
| (g) バックオフ(back off) | (h) タイマ(timer) 設定 |

(2-2) ネットワークの負荷が高くなった場合、(お) 方式が(え) 方式と比較して、遅延が急激に増加しない理由を説明せよ。

(2-3) CSMA/CD ではフレームが衝突すると、定められた平均値になるよう設定されたランダムな待ち時間だけ、送信ホストはフレームの再送を待つ。ここで、再送フレームが再び衝突した場合、さらなる衝突が発生しないように、CSMA/CD でなされている工夫を述べよ。

(3) TCP Reno が実装する輻輳制御(congestion control)に関する説明文を読み、以下の各小間に答えよ。

送信ホストは、輻輳ウィンドウ(congestion window)と呼ばれる、ACK セグメント(acknowledge segment)を受信せずに送信可能な DATA セグメント(data segment)のデータ量を用いて送信レート(transmission rate)を増減する。通信開始時の輻輳ウィンドウは、MSS (Maximum Segment Size)と呼ばれる最大セグメント長 1 個分である。通信開始後、送信ホストは ACK セグメントを受信するごとに、以下の式に従って輻輳ウィンドウを増加させる。

$$\text{新しい輻輳ウィンドウ} = \text{現在の輻輳ウィンドウ} + \boxed{\text{(あ)}} \times \text{MSS}$$

送信ホストは、
 (い) ことで DATA セグメントの紛失を検出すると、高速リカバリ(fast recovery)を開始し、それが終了した時点で輻輳ウィンドウを
 (う) 倍に減少させる。その後、送信ホストは、ACK セグメント受信毎の増加量を
 (え) ×MSS にする。またこの送信レートの増減のしかたは、
 (お) と呼ばれる。

(3-1) 上記の説明文の (あ) ~ (お) に当てはまる最も適切な語句を選択肢から一つ選び、その記号を答えよ。

選択肢：

- | | | | |
|---|--|---------|-------|
| (a) $\text{MSS} \div \text{現在の輻輳ウィンドウ}$ | (b) $2\text{MSS} \div \text{現在の輻輹ウィンドウ}$ | (c) 1 | (d) 2 |
| (e) 1/2 | (f) 1/4 | (g) 1/8 | |

(h) 重複した ACK セグメントを 3 個受信する

- | | |
|--|---|
| (i) 送信した DATA セグメントを確認応答する ACK セグメントを一定時間受信しない | |
| (j) ベストエフォート(best effort) | (k) Additive Increase/Multiplicative Decrease |
| (l) スロースタート(slow start) | (m) コネクションレス |

(3-2) 送信ホストに送信すべき DATA セグメントが常に存在すると仮定する場合、以下の条件に従つて通信を開始すると、送信ホストが最初の DATA セグメントを送信後、最長で何ミリ秒後までに最初の ACK セグメントを受信するかを求めよ。

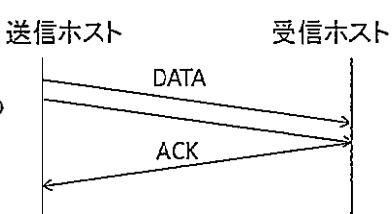
- 受信ホストは Delayed ACK に準じて、以下のいずれかの条件が満たされると、ACK セグメントを 1 個返送する。

条件 1：DATA セグメントを受信した時、当該 DATA セグメントを含めて、ACK セグメントを返送していない DATA セグメントのデータ量が MSS 2 個分以上である。

条件 2：200 ミリ秒周期の Delayed ACK タイマ発火時に、ACK セグメントを返送していない DATA セグメントが存在する。

- 片道の伝送遅延時間 5 ミリ秒。
- MSS 1K バイト。
- DATA 及び ACK セグメントは紛失しない。
- 全ての DATA セグメントのデータ量は MSS。
- 受信ホストが送信ホストに通知する最大のウィンドウサイズ 12K バイト。
- 以上の条件で所要時間が規定されていない処理については、所要時間は無視して良い。

(3-3) 小問(3-2)と同じ条件で通信を開始する時、送信ホストが最初の DATA セグメントを送信してから、最長で何ミリ秒後までに 5 個目の ACK セグメントを受信するかを求めよ。さらに、送信ホストが 5 個目の ACK セグメントを受信するまでの流れを、右の例に倣った図で示せ。



時間 0 で連続して DATA セグメントを送信する場合は、上図に示すように、それらを示す矢印をずらして記載すること。

配点：(1-1) 25 点，(1-2) 10 点，(1-3) 20 点，(1-4) 20 点，(2-1) 20 点，(2-2) 30 点

2 入力マルチプレクサ (multiplexor) について、次の各間に答えよ。

2 入力マルチプレクサは、3 個の入力 select, a, b と 1 個の出力 out をもち、入力 select が 1 のとき out に a の値を出力し、select が 0 のとき out に b の値を出力する。

(1) 2 入力マルチプレクサを次の手順で、論理回路 (logic circuit) で実現する。

(1-1) 2 入力マルチプレクサの真理値表 (truth table) を作成せよ。

(1-2) 2 入力マルチプレクサのカルノー図 (Karnaugh map) を作成せよ。

(1-3) (1-2) で作成したカルノー図を利用して、out の最簡積和形 (最小積和形, minimal sum-of-products expression) の論理式 (logic expression) を導出せよ。

(1-4) (1-3) で求めた最簡積和形の論理式を用いて得られる 2 入力マルチプレクサの論理回路図を示せ。ただし、論理ゲート (logic gate) は、2 入力 NAND ゲートのみが使用できるものとする。

(2) (1-4) で設計した 2 入力マルチプレクサの遅延時間 (delay) について考える。遅延時間とは、入力が変化してから出力が変化するまでに要する時間である。ただし、設計に用いた 2 入力 NAND ゲートの遅延時間は、出力が 0 から 1 に変化するときは T、出力が 1 から 0 に変化するときは 2T であるとする。

(2-1) 入力 (select, a, b) が (0, 0, 0) から (0, 0, 1) に変化するときの、マルチプレクサの遅延時間を T を用いて表せ。

(2-2) select, a, b のいずれか一つの入力が変化するときの、マルチプレクサの遅延時間の最大値を求めよ。また、そのときの入力の変化を下の例にならって答えよ。

例：(select, a, b) が (1, 1, 1) から (0, 1, 1) に変化するとき。

配点: (1-1)35 点, (1-2)40 点, (2)50 点

以下の各間に答えよ.

(1)

(1-1) $f(t) = \cos \omega t$ のラプラス変換 (Laplace transform) が $\frac{s}{s^2 + \omega^2}$ であることを示せ.
ただし、この関数 f は $(0, +\infty)$ で定義され、 ω は定数 (constant number), s は複素変数 (complex variable) である。

(1-2) ラプラス変換を用いて、次の積分方程式 (integral equation) を満たす $x(t)$ を求めよ。ただし、この関数 x は $(0, +\infty)$ で定義される。

$$x(t) = 1 - 4 \int_0^t (t - \tau)x(\tau)d\tau$$

(2)

図 1 に示すような周期 (period) $T = 2$ の矩形波 (square wave) $f(t)$ をフーリエ級数展開 (Fourier series expansion) せよ。ただし、 n は整数 (integer number) である。

$$f(t) = \begin{cases} 0 & (2n - 1 < t \leq 2n - 0.5) \\ 1 & (2n - 0.5 < t \leq 2n + 0.5) \\ 0 & (2n + 0.5 < t \leq 2n + 1) \end{cases}$$

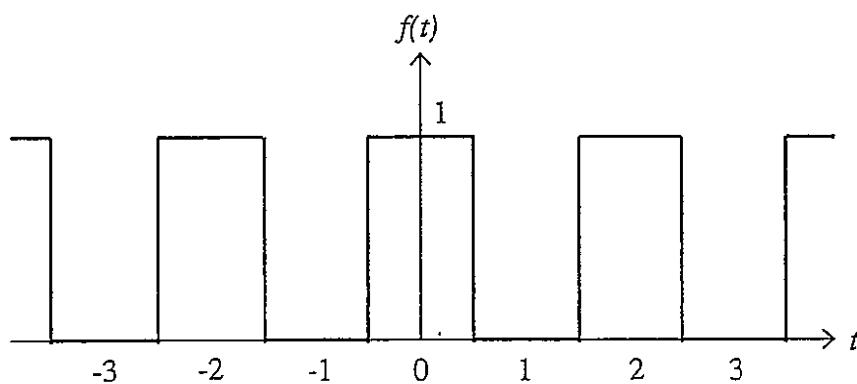


図 1: 周期 $T = 2$ の矩形波 $f(t)$

配点：(1) 26 点, (2-1) 12 点, (2-2) 36 点, (2-3) 26 点, (3) 25 点

図 1 は ANSI-C 準拠である C 言語のプログラム (program) である。このプログラムは、34 行目で宣言 (declare) される struct item_t 型の配列 (array) data 内の要素 (element) を、関数 (function) sort を用いて int 型である key の値に従って整列 (sort) したのち、整列した結果 (result) を関数 show で出力 (output) する。以下の各間に答えよ。

(1) 関数 show は、引数 (argument) である struct item_t 型の配列 data の要素のうち、int 型である key と char 型である val の値 (value) を先頭から順に 28 行目の関数 printf で出力する。そのような動作となるよう、プログラム中の空欄 (ア) および (イ) に適切な式 (expression) を埋めよ。

(2) 関数 sort は、引数である配列 data を入力データ (input data) として整列を実行する。以下の小間に答えよ。

(2-1) 関数 sort の引数 data の内容 (34 行目の (ウ)) が

$\{5, 'a'\}, \{2, 'd'\}, \{6, 'e'\}, \{4, 'h'\}, \{1, 'b'\}, \{8, 'f'\}, \{3, 'c'\}, \{7, 'g'\}$ であったとする。この時、プログラムの標準出力 (standard output) から得られる整列結果を答えよ。

(2-2) 関数 sort のうち、16 行目から 19 行目までの一連の代入は、配列の二つの要素の値を入れ替え操作 (swap operation) によって変更する処理である。この一連の代入が一回起きることを、入れ替え操作が一回起きたとする。34 行目の (ウ) が以下の (a) ~ (d) であったとき、入れ替え操作の回数は (a) ~ (d) それぞれ何回となるか答えよ。

- (a) $\{1, 'a'\}, \{2, 'd'\}, \{4, 'e'\}, \{6, 'h'\}, \{8, 'b'\}, \{3, 'f'\}, \{5, 'c'\}, \{7, 'g'\}$
- (b) $\{8, 'c'\}, \{2, 'f'\}, \{3, 'h'\}, \{4, 'g'\}, \{5, 'd'\}, \{6, 'e'\}, \{7, 'b'\}, \{1, 'c'\}$
- (c) $\{8, 'c'\}, \{1, 'f'\}, \{2, 'h'\}, \{3, 'g'\}, \{4, 'd'\}, \{5, 'e'\}, \{6, 'b'\}, \{7, 'c'\}$
- (d) $\{2, 'x'\}, \{4, 'y'\}, \{6, 'z'\}, \{8, 'w'\}, \{2, 's'\}, \{4, 't'\}, \{6, 'u'\}, \{8, 'v'\}$

(2-3) 関数 sort におけるデータの整列は、入力データに依存して入れ替え操作の回数が大きく変化する。整列が終了するまでに、入れ替え操作が k 回 ($0 \leq k \leq (\text{MAX} \times (\text{MAX} - 1))/2$) 起きるような入力データの条件は次の数式 (formula) で与えられる。

$$|\{<i,j> | 0 \leq i < j \leq \text{MAX} - 1, \boxed{\text{(エ)}}\} | = \boxed{\text{(オ)}}$$

ただし、 $|S|$ は集合 (set) S の要素数、 $<i,j>$ は順序対 (ordered pair) を表す。なお、入力データの key の値を先頭の要素から順に $a_0, a_1, \dots, a_{\text{MAX}-1}$ とする。空欄 (エ) および (オ) に適切な数式を埋めよ。

(3) key の値が同じ要素に対して、整列前の要素の並び順の前後関係が整列後も維持されるとき、安定な (stable) 整列アルゴリズムという。関数 sort が実現しているアルゴリズムが安定であるかどうか、プログラムの箇所を指摘しつつ、理由も含めて説明せよ。

```
1 #include <stdio.h>
2
3 #define MAX 8
4
5 struct item_t {
6     int key; char val;
7 };
8
9 void sort(struct item_t data[])
10 {
11     int i, j, tmp_key;
12     char tmp_val;
13
14     for (i = 1; i < MAX; i++)
15         for (j = i; j > 0 && data[j-1].key >= data[j].key; j--) {
16             tmp_key = data[j].key; tmp_val = data[j].val;
17             data[j].key = data[j-1].key;
18             data[j].val = data[j-1].val;
19             data[j-1].key = tmp_key; data[j-1].val = tmp_val;
20         }
21     }
22
23 void show(struct item_t data[])
24 {
25     int i = MAX;
26
27     while ( [ア] )
28         printf("(%d,%c) ", [イ].key, [イ].val);
29     printf("\n");
30 }
31
32 int main(void)
33 {
34     struct item_t data[MAX] = { [ウ] };
35
36     sort(data);
37
38     show(data);
39
40     return 0;
41 }
```

図1 プログラム

配点: (1-1-1) 7 点, (1-1-2) 7 点, (1-1-3) 7 点, (1-2) 20 点, (1-3) 20 点,
 (2-1) 14 点, (2-2-1) 16 点, (2-2-2) 14 点, (2-2-3) 20 点

(1) 計算機のキャッシュメモリ (cache memory) に関する以下の各小間に答えよ.

(1-1) セット数 2 のセット連想マッピング方式 (2 ウェイ群連想マッピング方式; 2-way set associative mapping) のキャッシュメモリについて考える. キャッシュサイズ (cache size) は 16 [バイト], ブロックサイズ (block size) は 4 [バイト], アドレス (address) はバイト単位, アドレス長は 8 [ビット], ブロック置き換え方法は LRU (Least Recently Used), キャッシュメモリの初期状態は空とする. 以下のアドレス (2 進数表記) に対するメモリアクセス (memory access) が上から順番に発生した場合を考える.

```

01011011
01001010
01001001
00010101
01101001
00010100
01101001
01001010
01011010
00010101
  
```

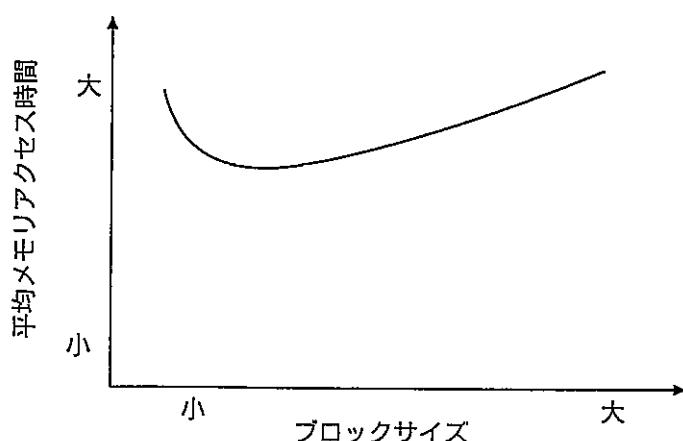
(1-1-1) 初期参照 (ブロックへの最初のアクセス) によるキャッシュミス (cache miss) の回数を求めよ.

(1-1-2) ブロックの置き換えを伴うキャッシュミスの回数を求めよ.

(1-1-3) キャッシュミス率を求めよ.

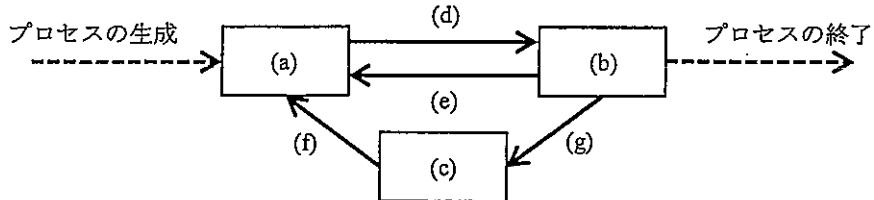
(1-2) キャッシュメモリはプログラムの局所性 (locality) を利用している. プログラムの局所性を二つ挙げよ. また, それがキャッシュの効果にどのように寄与するかを説明せよ.

(1-3) 一般に, ブロックサイズと平均メモリアクセス時間 (mean memory access time) との関係は下図のような傾向を示す. ブロックサイズが小さい時と, ブロックサイズが大きい時に, 平均メモリアクセス時間が大きな値となる理由をそれぞれ説明せよ.



(2) 単一プロセッサ(uniprocessor)をもつマルチプログラミングシステム(multiprogramming system)に関する以下の各小間に答えよ.

(2-1) 単一プロセッサを持つマルチプログラミングシステムにおいて、プロセス(process)は、実行(running)状態、実行可能(ready)状態、待ち(wait または blocked)状態の三つの状態を持ち、プロセスが生成されてから終了するまでその3状態間を遷移する。次に示すプロセスの状態遷移図(state transition diagram)の(a)～(g)にあてはまるもっともふさわしい語句を下の選択肢から一つ選び、記号で答えよ。



選択肢

- (ア) システムコール(system call) (イ) 入出力処理の完了 (ウ) セグメンテーション(segmentation)
- (エ) リモートプロシージャコール(remote procedure call) (オ) 待ち状態 (カ) ページング(paging)
- (キ) ディスパッチ(dispatch) (ク) 実行状態 (ケ) コンパイル(compile) (コ) 横取り(preemption)
- (サ) 実行可能状態 (シ) バッファリング(buffering) (ス) 入出力処理の発生

(2-2) 表1にプロセス P1, P2, P3, P4 が生成される時刻、およびそれぞれのプロセスの処理時間を示す。下記の仮定に留意し、次の(2-2-1)～(2-2-3)に答えよ。

仮定

1. ある時刻に生成あるいは横取りされたプロセスは、その時刻に直ちに実行可能キュー(ready queue)に格納される。生成されたプロセスと横取りされたプロセスが、実行可能キューに同時刻に格納される場合には、生成されたプロセスが先に実行可能キューに格納される。
2. あるプロセスが終了あるいは横取りされた時刻から次のプロセスがプロセッサにディスパッチされる時刻までに要する処理時間（プロセス切り替えに関する処理時間）を0とする。
3. ラウンドロビン(round robin)方式において、タイムスライス(time slice)中にプロセスが終了したときは、直ちに次のプロセスがディスパッチされる。

(2-2-1) プロセスのスケジューリングアルゴリズム(scheduling algorithm)として、次の二つの方式、到着順(first come first service (served))方式、処理時間順(shortest processing time first)方式を用いた場合のプロセス P1, P2, P3, P4 のターンアラウンド時間(turnaround time)を求めよ。

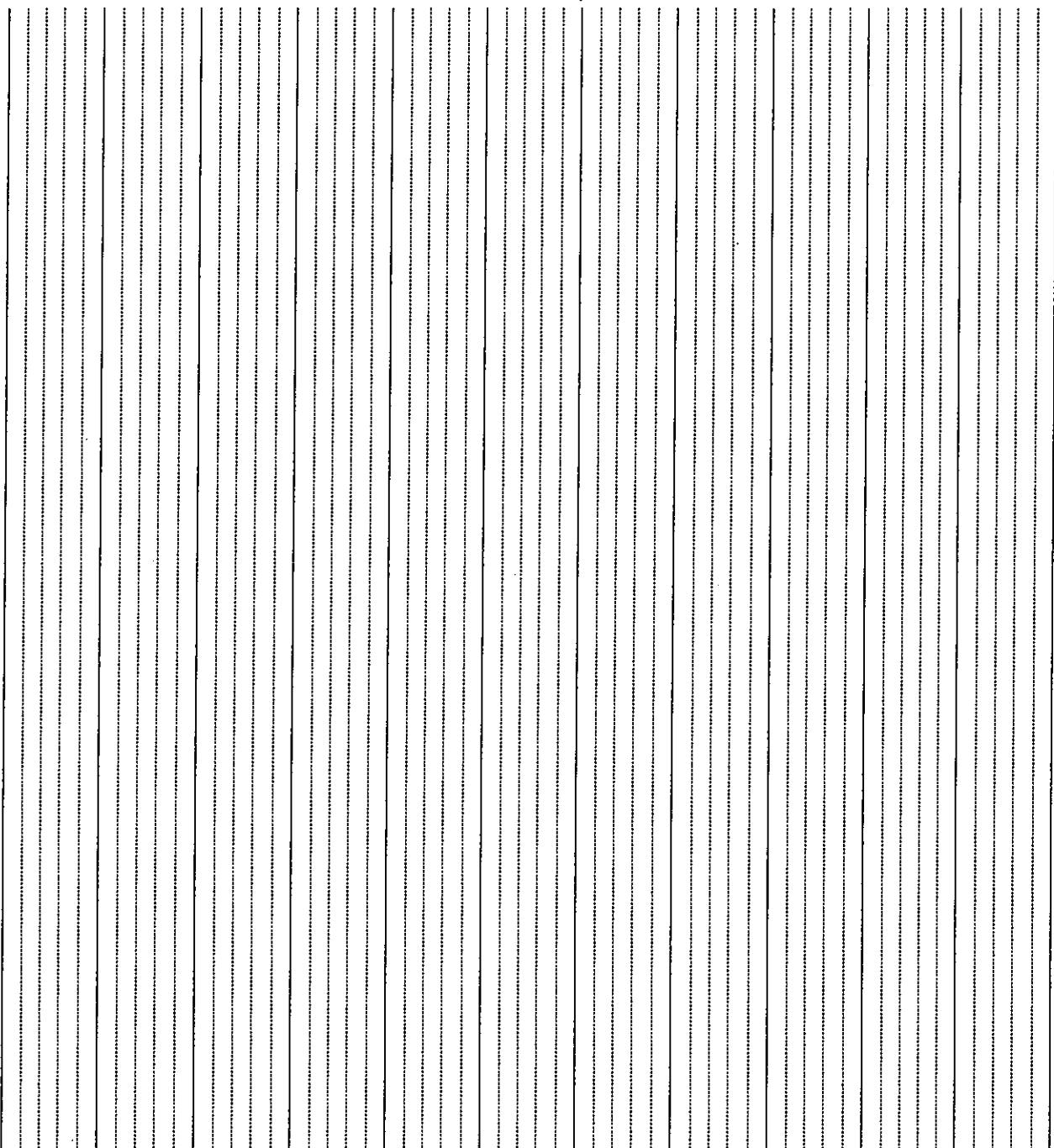
(2-2-2) プロセスのスケジューリングアルゴリズムとして、ラウンドロビン方式を用いる。(a) タイムスライスが4のときのプロセス P1, P2, P3, P4 の平均ターンアラウンド時間、(b) タイムスライスが8のときのプロセス P1, P2, P3, P4 の平均ターンアラウンド時間を求めよ。

(2-2-3) ラウンドロビン方式において、上記の仮定2.を変更し、プロセス切り替えに関する処理時間を0より大きくしたとする。この場合、タイムスライスの値の大小によって平均ターンアラウンド時間がどう変化するか説明せよ。

表1 プロセスの生成時刻と処理時間

プロセス	生成時刻	処理時間
P1	0	20
P2	8	40
P3	18	10
P4	24	30

(計算用紙)



配点:(1) 28 点 (2-1) 17 点 (2-2) 20 点 (3) 30 点 (4-1) 15 点 (4-2) 15 点

論理式 (logic formula) の記号 $\leftrightarrow, \Rightarrow, \wedge, \vee, \neg$ は、それぞれ、等価 (equivalence), 含意 (implication), 連言 (論理積) (conjunction, and), 選言 (論理和) (disjunction, or), 否定 (negation, not) を表す論理演算子とする。また、真 (true), 偽 (false) を表す、true, false の二値からなる集合を B とする。

- (1) 一階述語論理 (first-order logic) における式 (formula) の解釈 (interpretation) I は (D, C, F, P) の 4 項組で与えられる。ここで、 D は値集合、 C は各定数記号への D の要素の割り当て、 F は各 n 引数関数記号への $f : D^n \rightarrow D$ なる関数 f の割り当て、 P は各 n 引数述語記号への $p : D^n \rightarrow B$ なる述語 p の割り当てである。

例えば一階述語論理式 $\forall x p(f(b, x), a)$ に対して、解釈 I_0 として

- D を非負整数 (nonnegative integer) 全体からなる集合とし、
- C として a, b それぞれへ非負整数值 0,1 を割り当て、
- F として 2 引数関数記号 $f(u, w)$ へ非負整数上の加算 $u + w$ を割り当て、
- P として 2 引数述語記号 $p(u, w)$ へ非負整数上の比較演算 $u > w$ を割り当てたとき (例えば $4 > 3$ の値は true である)、

式 $\forall x p(f(b, x), a)$ の解釈 I_0 のもとでの評価値は true となる。

以下の各式が恒真 (valid) か、恒真ではないが充足可能 (satisfiable) か、充足不能 (unsatisfiable) かを答えよ。恒真ではないが充足可能の場合は、値集合 D を $\{a, b\}$ で固定して、真にする解釈と偽にする解釈を 1 つずつ与えよ。

- (a) $\forall x p(x) \Rightarrow \forall y p(y)$
- (b) $\exists x p(x) \Rightarrow \forall x p(x)$
- (c) $\neg \exists y p(y) \Leftrightarrow \forall x p(x)$
- (d) $\forall x \exists y q(x, y) \Rightarrow \exists y \forall x q(x, y)$

- (2) 以下で与えられる論理式 E が恒真であることを、まず、 E の否定のスコーレム化 (Skolemization)，すなわち限量子 (quantifier) \exists に関する変数の消去、を行い、ついで、導出原理 (resolution principle) により、充足不能であることの確認を行うことによって示したい。

$$\begin{aligned} A &= p(f(g(f(g(g(a)))))) \\ B &= \forall x (p(f(g(x))) \Rightarrow p(x)) \\ C &= \forall x (p(g(f(x))) \Rightarrow p(x)) \\ D &= \exists x p(g(x)) \\ E &= (A \wedge B \wedge C) \Rightarrow D \end{aligned}$$

ただし、 a は定数記号、 f, g は関数記号、 p は述語記号である。

以下の各小間に答えよ。

- (2-1) $\neg E$ のスコーレム連言標準形 (Skolem conjunction normal form) E' を求めよ。導出過程は不要。 E' 中では記号 A, B, C, D を用いないこと。
- (2-2) 小問 (2-1) で得た論理式 E' をもとに、導出原理を用いて E' が充足不能であることを示せ。

- (3) $X \cap Y = \emptyset$, $X \neq \emptyset$, $Y \neq \emptyset$ である任意の有限集合 X , Y に対し, $X \cup Y$ 上の以下の関係 R が, 反射律 (reflexive law), 反対称律 (antisymmetric law), 推移律 (transitive law) を満たすかどうかを述べ, その証明も与えよ. また R は順序関係 (order relation) かそうでないかを述べよ.

$$R = X \times Y \cup \{ (x, x) \mid x \in X \cup Y \}$$

- (4) $\Sigma = \{a, b\}$ とする. Σ 上の長さ n ($n \geq 1$) の文字列のうち, bb を部分文字列として含まない文字列の集合を L_n で表し, L_n の要素数を $|L_n|$ で表す. 以下の各小間に答えよ.

- (4-1) 最後尾が w ($w \in \Sigma$) である文字列からなる L_n の最大部分集合を $L_n(w)$ で表す. $n > 2$ のとき, $|L_n(a)|$, $|L_n(b)|$ それぞれを $|L_{n-1}|$, $|L_{n-2}|$ を用いて表せ.
- (4-2) $|L_n|$ を n についての漸化式で表せ.

配点: (1) 15 点, (2) 30 点, (3) 20 点, (4-1) 35 点, (4-2) 25 点

(1) 以下に示す各言語 L_1, \dots, L_6 が文脈自由言語 (context-free language) か否かを, ○ (文脈自由言語である) か× (文脈自由言語でない) で答えよ. 証明は不要である.

- $L_1 = \{ww \mid w \in \{a,b\}^*\}$
- $L_2 = \{ww^R \mid w \in \{a,b\}^*, w^R$ は w の反転 (reversal) $\}$
- $L_3 = \{a^n b^m c^n d^m \mid n \geq 1$ かつ $m \geq 1\}$
- $L_4 = \{a^n b^m c^m d^n \mid n \geq 1$ かつ $m \geq 1\}$
- $L_5 = \{a^n b^n c^m d^m \mid n \geq 1$ かつ $m \geq 1\}$
- $L_6 = \{a^n b^n c^n \mid n \geq 1\}$

(2) 文脈自由文法 (context-free grammar) G_1 を以下の通りとする. この文法には単記号規則 (unit production), すなわち非終端記号 (non-terminal symbols) A と B に対して $A \rightarrow B$ の形の生成規則 (generating rule) が含まれている. このとき新たな非終端記号を追加せずに G_1 から単記号規則を除去した等価 (equivalent) な文法における, 非終端記号 E に対する生成規則すべてを示せ.

$G_1(N_1, T_1, P_1, S_1)$ —————

- 非終端記号 (non-terminal symbol) の集合 $N_1 = \{ E, T, F, I \}$
- 終端記号 (terminal symbol) の集合 $T_1 = \{ a, b, (,), *, + \}$
- 生成規則 (generating rule) の集合 $P_1 = \{ E \rightarrow T, E \rightarrow E + T, T \rightarrow F, T \rightarrow T * F, F \rightarrow I, F \rightarrow (E), I \rightarrow a, I \rightarrow b, I \rightarrow Ia, I \rightarrow Ib \}$
- 始記号 (start symbol) $S_1 = E$

(3) プログラミング言語の文法として, 正規文法 (regular grammar) や文脈依存文法 (context-sensitive grammar) ではなく, 文脈自由文法を用いる事のそれぞれに対する利点を簡潔に述べよ.

(4) オートマトン (automaton) に関する以下の各小間に答えよ。オートマトン M によって認識される言語 (language recognized by M) を $L(M)$ と表記する。すべてのオートマトンについて、入力記号 (input symbols) の集合を $\Sigma = \{0, 1\}$ とする。また、関数 $F : \Sigma^* \rightarrow \mathbb{N}$ を、任意の語 (word) $w = a_1a_2\dots a_n \in \Sigma^*$ に対し、 w を a_1 を最下位ビット (least significant bit) とした 2 進表現とみなして 0 以上の整数值を与える関数とする (\mathbb{N} は 0 以上の整数の集合を表す)。ただし、空語 (empty word) ϵ に対しては 0 を与える。すなわち、 a_i を整数值とみなして F を以下のように定義する。

$$F(w) = \begin{cases} \sum_{i=1}^n 2^{i-1} \cdot a_i & w = a_1a_2\dots a_n, n \geq 1 \\ 0 & w = \epsilon \end{cases}$$

たとえば以下が成り立つ。 $F(\epsilon) = 0, F(0) = 0, F(10) = 1, F(0001) = 8, F(1100) = 3$

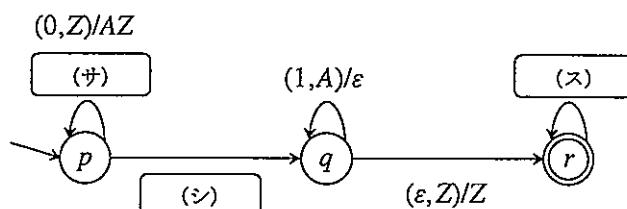
(4-1) $L(M_1) = \{w \mid w \in \Sigma^*, \exists k \in \mathbb{N} : F(w) = 3k\}$ である決定性有限オートマトン (deterministic finite automaton) M_1 について考える。以下は、 M_1 の状態遷移表 (state transition table) である。

	0	1
$\rightarrow @$	b	d
Ⓐ	(ア)	(イ)
c	(ウ)	(エ)
d	c	a
e	(オ)	(カ)
f	(キ)	(ク)

注。 \rightarrow は初期状態、Ⓐは受理状態を示す。

上の状態遷移表の空白部分 (ア)～(ク)を埋めよ。ただし、 M_1 の初期状態 (initial state) は a 、受理状態 (accepting states) は a と b である。

(4-2) 最終状態による受理 (acceptance by final state) を行うプッシュダウンオートマトン (pushdown automaton) M_2 を、 $L(M_2) = \{w \mid w \in \Sigma^*, \exists k \in \mathbb{N} : (F(w) = 2^{2k} - 2^k \text{かつ } F(w) > 0)\}$ となるように構成することを考える。スタック記号 (stack symbols) は A と Z であり、 Z を初期 (底) 記号 (initial stack symbol) とする。以下は、 M_2 の状態遷移図 (state transition diagram) である。図の空白部分 (サ)、(シ)、(ス) を適切な動作で埋めよ。ただし、 M_2 の初期状態 (initial state) は p 、受理状態 (accepting state) は r である。



5 【選択問題】ネットワーク

(情報工学 10/16)

配点: (1-1) 8 点, (1-2) 8 点, (1-3) 8 点, (1-4) 10 点, (1-5) 10 点,
(2-1) 30 点, (2-2) 12 点, (2-3) 24 点, (2-4) 15 点

(1) 通信路符号化に関する以下の文章を読み、その下の各小間に答えよ。

C を 2 元ブロック符号 (binary block code) とし、正整数 d を符号 C の最小ハミング距離 (minimum Hamming distance) とする。 t を、 $2t+1 \leq d$ を満たす任意の非負整数とする。受信語 (received word) v に対する、限界距離 (bounded distance) を t とした復号法 (decoding algorithm) とは、以下のような手続きである。

(i) $C_t(v) = \{w \mid w \in C, \Delta(v, w) \leq t\}$ を求める。ここで $\Delta(v, w)$ は語 v と w の間のハミング距離 (Hamming distance) を表す。このとき、 $C_t(v)$ の要素数は高々 1 となる (あ)。

(ii) $C_t(v)$ が空集合でなければ、その要素を v の復号結果とする。 $C_t(v)$ が空集合ならば、(い)。

以下では、 C は線形符号 (linear code) であるとする。受信語 (を表す行ベクトル) v と、検査行列 (check matrix) H の転置行列 (transposed matrix) H^T との積 vH^T は、(う) と呼ばれる。

(う) は、 v が符号語 (codeword) のときかつそのときのみ零ベクトルとなる。さらに、ハミング距離 $(d-1)/2$ 以内に符号語 w が存在するような v に対しては、(う) は v と w の相違箇所のみに依存して定まる行ベクトルとなる。

(1-1) $C = \{000, 111\}$ の場合の $C_0(010)$ と $C_1(101)$ をそれぞれ求めよ。

(1-2) 限界距離復号法の正しい動作を表す文となるように、空欄 (い) を埋めよ。

(1-3) 空欄 (う) を、適切な用語で埋めよ。

(1-4) 任意の符号 C と任意の受信語 v について下線部 (あ) が成立することを説明せよ。必要であれば、ハミング距離が三角不等式 (triangle inequality) を満たすことを既知の事実として用いてよい。

(1-5) 符号 $C = \{000, 111\}$ は線形符号である。この符号 C の検査行列を一つ答えよ。どのようにして求めたかも記述すること。

- (2) TCP/IP 参照モデル (reference model) におけるトранSPORT層 (transport layer) に関する以下の文章を読み、その下の各小間に答えよ。

TCP/IP 参照モデルにおけるトランSPORT層は、アプリケーション層 (application layer) で動作する二つのプロセス (process) の間に論理的な通信路 (channel) を提供する。論理的な通信路とは、物理的には接続されていないものの、アプリケーション層で動作するプロセスにとってはあたかも物理的に接続されているように見える通信路である。アプリケーション層で動作するプロセスは、トランSPORT層により提供される論理的な通信路を用いてデータを転送する。

TCP/IP 参照モデルにおけるトランSPORT層プロトコル (transport layer protocol) の代表的なものとして、TCP と UDP がある。TCP は、シーケンス番号 (sequence number), 確認応答 (ACK), タイマ等を用いて、データが欠損することなく正しい順序で転送されることを保証するコネクション型 (connection-oriented) のプロトコルである。TCP では、データ転送の開始前にスリーウェイハンドシェイク (three-way handshake) を行い、プロセス間のコネクションを設定する。一方、UDP は、プロセス間のコネクションを設定することなくデータ転送を開始するコネクションレス型 (connectionless) のプロトコルである。

以降では、トランSPORT層のサービスを提供するハードウェア (hardware) もしくはソフトウェア (software) をトランSPORTエンティティ (transport entity) と呼ぶ。

- (2-1) トランSPORT層プロトコルとして UDP を用いる場合、TCP を用いる場合と比較しての利点を二つ述べよ。それぞれについて、その利点を有する理由も併せて述べよ。
- (2-2) アプリケーション層で複数のプロセスが動作する場合、トランSPORT層ではネットワーク層 (network layer) から TPDU (Transport Protocol Data Unit) を受け取り、TPDU に格納されたデータを適切なプロセスに渡さなければならない。
- トランSPORTエンティティは、四つの情報の組み合わせによって TCP のコネクションを識別し、TCP の TPDU であるセグメント (segment) に格納されたデータを適切なプロセスに受け渡す。TCP のコネクションを識別するために必要な四つの情報を列挙せよ。
- (2-3) 図 1 は、アプリケーション層で動作するプロセス A とプロセス B の間で TCP を用いてデータを転送する際、トランSPORTエンティティ間におけるセグメント送受信の例を示している。この例では、プロセス A からプロセス B に対してデータを転送している。

プロセス A 側のトランSPORTエンティティは、プロセス A からコネクション設定を依頼されるとスリーウェイハンドシェイクを開始する。スリーウェイハンドシェイクでは、プロセス A 側のトランSPORTエンティティおよびプロセス B 側のトランSPORTエンティティがそれぞれランダムに初期シーケンス番号 (initial sequence number) を決定し、初期シーケンス番号を互いに通知する。

トランSPORTエンティティが、プロセスからのコネクション設定依頼に対して常に同じ初期シーケンス番号を用いる場合に生じる問題を二つ述べよ。それぞれの問題について、その問題がどのような状況で生じるかを説明し、スリーウェイハンドシェイクにおいて初期シーケンス番号をランダムに決定することにより問題がどのように回避されるかを述べよ。

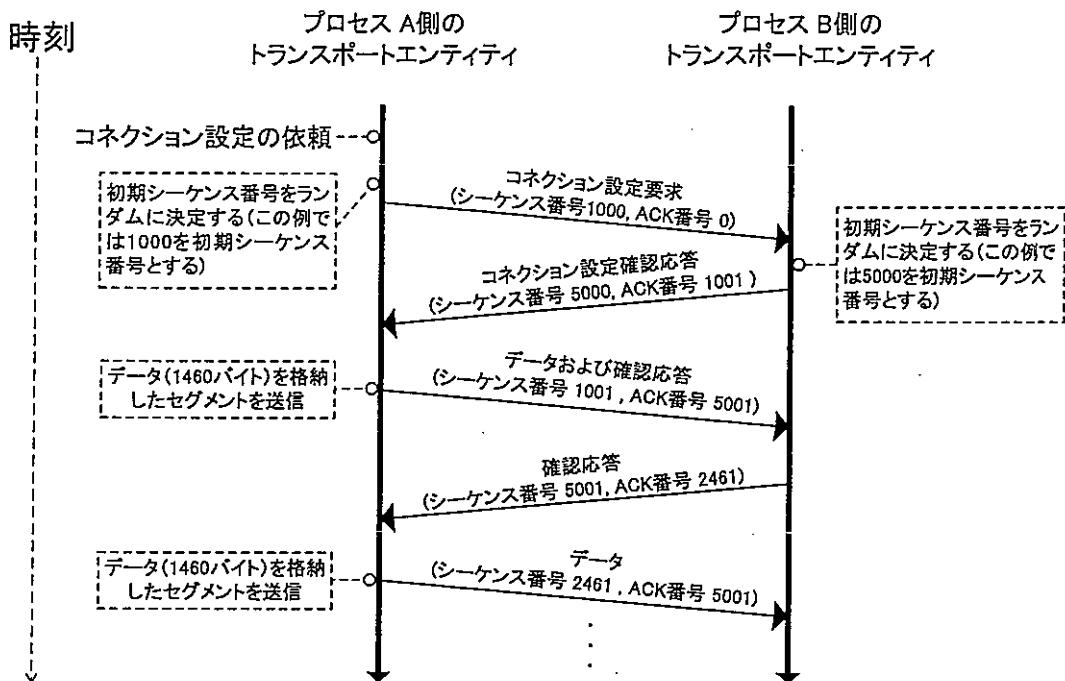


図 1

(2-4) プロセス A が TCP を用いてプロセス B から 100000 [バイト] のファイル (file) を取得する場合を考える。以下に示した条件において、プロセス A 側のトランsportエンティティが送信するセグメントの総数と、プロセス B 側のトランsportエンティティが送信するセグメントの総数を、それぞれ答えよ。計算過程も示せ。なお、コネクション解放時に送信されるセグメントの数は総数に含めない。

- プロセス A とプロセス B の間にコネクションを新たに設定し、そのコネクションを用いてデータを転送する。また、プロセス A 側のトランsportエンティティがスリーウェイハンドシェイクを開始する。
- プロセス A は、プロセス B に対して取得するファイルのファイル名等が記述された 100 [バイト] のデータを転送する。また、プロセス B は、プロセス A に対して取得するファイルのサイズ等が記述された 100 [バイト] のデータと取得するファイルからなる 100100 [バイト] のデータを転送する。
- セグメントのヘッダ (header) のサイズを 20 [バイト]、ヘッダを含めた最大セグメントサイズ (maximum segment size) を 1000 [バイト] とする。
- トランsportエンティティは、データが格納されたセグメントを一つ受信すると確認応答のセグメントを一つ送信する。
- トランsportエンティティは、確認応答のセグメントをピギーバック (piggyback) する。すなわち、トランsportエンティティは、転送するデータがある場合、確認応答のセグメントにデータを格納する。
- トランsportエンティティがセグメントの送受信に用いるバッファは十分に大きく、常に最大セグメントサイズ以上の空き領域がある。
- ファイルやデータの圧縮は考えない。
- セグメントの損失は生じない。

配点: (1-1) 30 点, (1-2) 40 点, (1-3) 30 点, (2) 25 点

連続する 4 個以上の 1 または連続する 3 個以上の 0 を検出する順序回路(sequential circuit)を設計する。この回路は 1 ビットの入力(input) x と 1 ビットの出力(output) z を持つ。各時刻には、 x に 0 または 1 が入力される。 x に 1 が 4 個以上連続入力される、あるいは x に 0 が 3 個以上連続入力されると z は 1 を出力し、それ以外の場合は 0 を出力する。例えば、 x に 0110111110100001 が入力された場合の出力は表 1 のようになる。ただし時刻 0 以前に入力はない。

表 1

時刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
入力	x	0	1	1	0	1	1	1	1	0	1	0	0	0	0	1	...
出力	z	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	...

(1) この回路を 6 状態(state) A, B, C, D, E, F を持つミーリー(Mealy)型順序回路として設計する。以下の小間に答えよ。

(1-1) 各状態を表 2 のように定めるとして、状態遷移図 (state transition diagram) を示せ。表中で現時刻は T とする。出力 z も図中に記載せよ。

表 2

A	初期状態(過去に何も入力されていない状態)
B	時刻 $T-1$ に 0 が入力されたが、時刻 $T-2$ には 0 が入力されていない状態
C	時刻 $T-1$, $T-2$ に共に 0 が入力された状態
D	時刻 $T-1$ に 1 が入力されたが、時刻 $T-2$ には 1 が入力されていない状態
E	時刻 $T-1$, $T-2$ に共に 1 が入力されたが、時刻 $T-3$ には 1 が入力されていない状態
F	時刻 $T-1$, $T-2$, および $T-3$ にすべて 1 が入力された状態

- (1-2) 表3のように3ビットを用いた状態割り当て(state assignment)を行い, 3個のDフリップフロップ(flip flop)を用いて設計する。状態変数(state variable) q_1 , q_2 , q_3 に対応するフリップフロップのD入力を d_1 , d_2 , d_3 とする。 d_1 , d_2 , d_3 および z を q_1 , q_2 , q_3 , x の最小積和形(最簡積和形, minimum sum-of-products expression)で表せ。

表3

状態	q_1	q_2	q_3
A	0	0	0
B	0	0	1
C	0	1	1
D	1	1	1
E	1	1	0
F	1	0	0

- (1-3) (1-2)で求めた最小積和形を利用して、DフリップフロップとNAND, NOTゲートを用いてこの回路を実現し、その回路図を示せ。ただし、NANDゲートの最大入力数は3とする。

- (2) 2入力NANDゲートをPMOS, NMOSからなるCMOS回路として実現したときの回路図を示せ。電源をVdd, グラウンドをVss, 入力信号をそれぞれIN₁, IN₂, 出力をOUTとして記載せよ。

配点：(1) 40 点, (2) 45 点, (3) 40 点

以下の各間に答えよ。

- (1) 常微分方程式 (ordinary differential equation) $f''(t) + 3f'(t) + 2f(t) = 0$ を、ラプラス変換 (Laplace transform) を用いて、初期条件 (initial condition) $f(0) = 2, f'(0) = -3$ のもとで解け。
- (2) 以下の定積分 (definite integral) I を留数定理 (residue theorem) を用いて求めよ。

$$I = \int_0^{2\pi} \frac{1}{5 + 4 \sin \theta} d\theta$$

(次ページへ続く)

- (3) 整数 (integer) k, n に対し、周期 (period) N の周期的 (periodic) な離散信号 (discrete signal) $f(k)$ を考え、 $f(k)$ の離散フーリエ変換 (discrete Fourier transform) $F(n)$ を

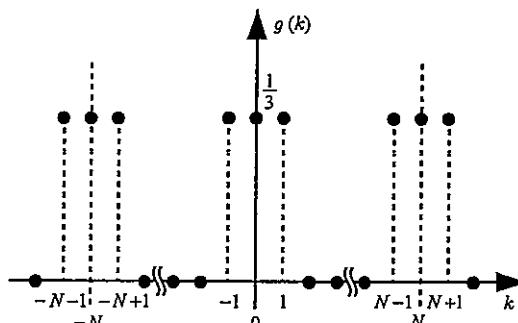
$$F(n) = \sum_{k=0}^{N-1} f(k) e^{-j \frac{2\pi}{N} nk} \quad (1)$$

と定義する。なお、ここで j は虚数単位 (imaginary unit) を表わす。また、入力 (input) $f(k)$ に対し、 $h(k) = \{f(k-1) + f(k) + f(k+1)\}/3$ を出力 (output) するフィルタ (filter) を考える。このとき、以下の文章の空欄 (A), (B) を適切な語句または数式で埋めよ。また、空欄 (c)～(h) に当てはまる最も適切な語句または数式を下記に示す選択肢から選んで記号を解答せよ。

フィルタの出力信号 $h(k)$ は、入力信号 $f(k)$ および下図に示す関数 $g(k)$ を用いて、 $h(k) = \boxed{\text{(c)}}$ と表わされる。 $g(k)$ の離散フーリエ変換 $G(n)$ を求めると、 $G(n) = \boxed{\text{(A)}}$ となる。 $h(k)$ の離散フーリエ変換 $H(n)$ は、 $F(n), G(n)$ を用いて $H(n) = \boxed{\text{(d)}}$ と表わされる。 $|F(n)|^2$ や $|H(n)|^2$ は信号の $\boxed{\text{(B)}}$ と呼ばれ、信号が運ぶ周波数毎のエネルギーを表している。上述の離散フーリエ変換の定義に対応する逆離散フーリエ変換 (inverse discrete Fourier transform) は、 $f(k) = \boxed{\text{(e)}}$ である。

このフィルタは、 $\boxed{\text{(f)}}$ フィルタであり、信号の $\boxed{\text{(g)}}$ する効果がある。

離散フーリエ変換を全ての $n = 0, 1, \dots, N-1$ について計算機で数値計算する場合、式 (1) の通りに計算すると、時間計算量 (time complexity) は $O(\boxed{\text{(h)}})$ となる。



【選択肢】

- | | |
|--|--|
| (ア) アナログ (analog), | (イ) ハイパス (high-pass), |
| (ウ) ローパス (low-pass), | (エ) バンドパス (band-pass), |
| (オ) バンドストップ (band-stop), | (カ) マルチバンド (multi-band), |
| (キ) 振幅 (amplitude) を增幅 (amplify), | (ク) コントラスト (contrast) を強調 (enhance), |
| (ケ) 分解能 (resolution) を改善 (improve), | (コ) 雑音 (noise) を低減 (reduce), |
| (サ) エッジ (edge) を鮮鋭に (sharpen), | (シ) $f(k)g(k)$, |
| (ス) $F(n)G(n)$, | (セ) $\sum_{m=0}^k f(m)g(m)$, |
| (ソ) $\sum_{m=0}^n F(m)G(m)$, | (タ) $\sum_{m=0}^{N-1} f(m)g(k-m)$, |
| (チ) $\sum_{m=0}^n F(m)G(n-m)$, | (ツ) $\sum_{n=0}^N F(n)e^{j \frac{2\pi}{N} nk}$, |
| (テ) $\frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} F(n)e^{j \frac{2\pi}{N} nk}$, | (ト) $\frac{1}{N} \sum_{n=0}^{N-1} F(n)e^{j \frac{2\pi}{N} nk}$, |
| (ナ) $\int_{-\infty}^{\infty} f(x)g(k-x)dx$, | (ニ) $\int_{-\infty}^{\infty} F(\omega)G(n-\omega)d\omega$, |
| (ヌ) $\frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{-j\omega k} d\omega$, | (ネ) N , |
| (ノ) $N \log N$, | (ヌ) $N\sqrt{N}$, |
| (ヒ) N^2 | |