〔C1 C2〕−2024− デジタル

専門（記述式）試験問題

注　意　事　項

1. 問題は **6 題**あります。このうち**任意の 2 題**を選んで解答してください。なお、この問題集の**裏表紙**に科目別構成の詳細が記載されていますので、解答開始前によく読んでおいてください。

2. 解答時間は **3 時間**です。

3. 答案用紙の記入について
   ㋐　答案は濃くはっきり書き、書き損じた場合は、解答の内容がはっきり分かるように訂正してください。
   ㋑　問題 **1 題に 1 枚**（両面）を使用してください。
   ㋒　表側の各欄にそれぞれ必要事項を記入してください。
   問題番号欄には、解答した問題の番号をそれぞれ記入してください。
   ㋓　試験の公正を害するおそれがありますので、答案用紙の氏名欄以外に氏名その他解答と関係のない事項を記載しないでください。

4. 下書き用紙はこの問題集の**中央部**にとじ込んであります。**試験官の指示**に従って、**試験開始後に**問題集から下書き用紙だけを慎重に**引きはがして**使用してください。なお、誤って問題集を破損しても、問題集の交換はできませんので注意してください。

5. この問題集で単位の明示されていない量については、全て国際単位系（SI）を用いることとします。

6. この問題集は、本試験種目終了後に持ち帰りができます。

7. 本試験種目の途中で退室する場合は、退室時の問題集及び下書き用紙の持ち帰りはできませんが、希望する方には後ほど渡します。別途試験官の指示に従ってください。なお、試験時間中に、この問題集から**下書き用紙以外**を切り取ったり、問題を転記したりしないでください。

8. 下欄に受験番号等を記入してください。

第 1 次試験地	試験の区分	受験番号	氏　　名
	デジタル		

指示があるまで中を開いてはいけません。

途中で退室する場合……本試験種目終了後の問題集の持ち帰りを　　**希望しない**

【No. **1**】 本問を通して、＋は言語の和、* はスター演算(クリーネ閉包)、ϵ は空列を意味する記号とし、$\Sigma = \{0, 1\}$ とする。以下の設問に答えよ。

(1) 次の正規表現 E_1, E_2, E_3 により定義される Σ 上の言語 $L(E_1)$, $L(E_2)$, $L(E_3)$ はどのようなものか。それぞれについて、20 字程度で説明せよ。

(a) $E_1 = (0+1)^*(01+10)$

(b) $E_2 = (\epsilon+1)(01)^*(\epsilon+0)$

(c) $E_3 = (1(0+01)^*)+((0+01)^*)$

(2) $L(E_1)$, $L(E_2)$, $L(E_3)$ を(1)で定義されている言語とする。次に述べる値を求めよ。

(a) $L(E_1)$ に含まれる長さ 4 以下の語の個数

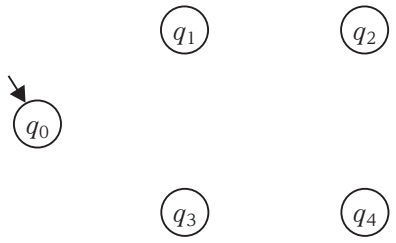
(b) $L(E_1) \cap L(E_2)$ に含まれる長さ 5 以下の語の個数

(c) $L(E_3)$ に含まれる、長さ 10 で記号 1 をちょうど三つ含むような語の個数

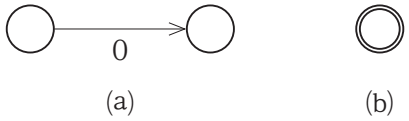
(3) (1)で定義した言語 $L(E_1)$ を認識する、状態数 5 の決定性有限オートマトンを図示せよ。解答は、答案用紙中に次の図Ⅰを転記し、次の二つのルールに従って記述すること。

・状態遷移は、図Ⅱ(a)で示すような、当該遷移に対応する入力シンボルを付記した矢印で表すこと。

・始状態は必ず q_0 として、終状態は図Ⅱ(b)で示すような二重丸で表すこと。



図Ⅰ



図Ⅱ

(4) L を Σ 上の言語とする。二つの語 x, $y \in \Sigma^*$ が、どのような $z \in \Sigma^*$ に対しても、$xz \in L$ かつ $yz \in L$、あるいは、$xz \notin L$ かつ $yz \notin L$ のいずれかを満たすとき、二項関係を表す記号 $\equiv_L$ を用いて $x \equiv_L y$ と表すこととする($x \equiv_L y$ であるためには、$z = \epsilon$ に対しても上の条件が成立する必要があることに注意せよ。)。このとき、以下の問いに答えよ。

(a) Σ^* 上の二項関係 $\equiv_L$ が同値関係であることを示せ。ただし、反射性及び対称性は定義より自明に成立するとしてよい。

(b) $L(E_1)$を(1)で定義した言語として、$L' \subseteq \Sigma^*$ を以下のように定義する。

$$L' = \{\epsilon, 00, 01, 10, 11, 001, 010\}$$

L' を、$\equiv_{L(E_1)}$ に関して互いに同値であるような値の集合(同値類)に分割せよ。

(c) L を正規言語、M を L を認識する任意の決定性有限オートマトンとする。語 $w \in \Sigma^*$ を M に入力として与えた場合における M の最終状態を $\Delta_M(w)$ で表すものとする。任意の $x,\ y \in \Sigma^*$ について以下の式が成立することを証明せよ。

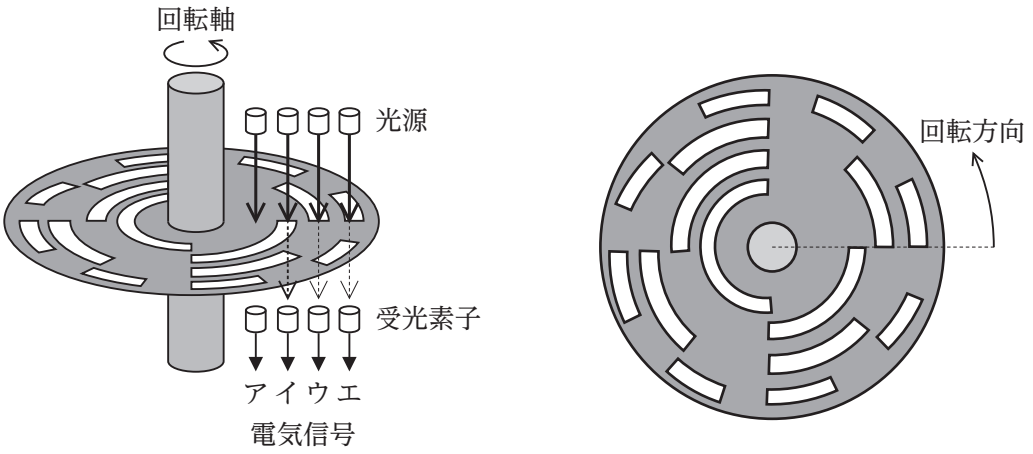
$$\Delta_M(x) = \Delta_M(y) \Rightarrow x \equiv_L y$$

(5) (4)で示した事実を用いて、(1)で定義した言語 $L(E_1)$ を認識する状態数 4 以下の決定性有限オートマトンは存在しないことを証明せよ。

【No.　**2**】 デジタル回路に関する以下の設問に答えよ。

ロボットのアームの角度など、刻々と変化する角度の情報をデジタルデータとして読み取るために、「ロータリーエンコーダ」と呼ばれる部品が用いられる。ロータリーエンコーダには、パルス数を数えて角度を求めるインクリメンタル型のほかに、基準位置からの絶対角度に比例した数値を、直接デジタルデータとして出力するアブソリュート型がある。

アブソリュート型ロータリーエンコーダは、図Ⅰ(a)に示すように、出力データの各ビット値に対応した同心円状のパターンの、光を通過させるスリットが刻まれた図Ⅰ(b)のようなディスクが回転軸に取り付けられている。そして、1列に並んだ受光素子が、連続的に変化するディスクの回転角度により刻々と変化するスリットのパターンを読み取る。光源のビームが、あるビットでは透過し、別のビットでは遮られ、透過したビームが受光素子で電気信号に変換され、波形整形されて各ビットのデジタル信号が作られる。



(a)　ロータリーエンコーダ　　(b)　回転ディスク

図Ⅰ　アブソリュート型ロータリーエンコーダと4ビットバイナリ符号のパターンをもつ回転ディスク

(1)　各光源の中心が図Ⅰ(b)点線上に位置する状態から矢印の方向に回転ディスクが一回転するとき、四つの受光素子ア～エが出力する電気信号が回転角とともにどのように変化するかをそれぞれ表したグラフを、できるだけ正確に描け。

ただし、光源のビームは一定の太さをもち、受光素子は強い光が当たるほど強い電気信号を出力するものとする。

(2)　図Ⅰ(b)の回転ディスクは、2進数(バイナリ符号)のデジタルデータを発生させるディスクであるが、実際にはこれを用いても、回転角度を反映した正しい値が常に出力されるとは限らない。図Ⅰ(a)に示された状況を参考に、どのような問題が生じるのかを200字以内で説明せよ。

ただし、次の語句を全て用い、用いた語句に下線を引くこと。

[語句：信号が連続的に変化、受光素子の感度のばらつき]

(3) (2)で述べたような問題を解決するために、図Ⅰ(b)のような通常の2進数(バイナリ符号)とは異なる符号化法を定めて、回転ディスクのパターンとして用いることを考える。このとき、以下の問いに答えよ。

(a) 2ビット符号列00，01，1㋐，1㋑で、隣り合う符号の間で(末尾の符号と先頭の符号も隣り合っていると考えるものとする。)、変化するビットが必ず1ビットだけに限られるようにするには、㋐と㋑に0と1のいずれが入ればよいか答えよ。

(b) 3ビット符号列000，001，01㋐，01㋑，1㋒，1㋓，1㋔，1㋕で、隣り合う符号の間で(末尾の符号と先頭の符号も隣り合っていると考えるものとする。)変化するビットが1ビットしかないようにするには、㋒～㋕に00～11のいずれが入ればよいか答えよ。ただし、㋐と㋑には、(a)と同じ値が入るものとする。

(c) 隣り合う符号の間で変化するビットが1ビットしかないNビット符号列が与えられたとき、これを基に同様なN＋1ビット符号列を構成する方法を示せ。

(d) このような符号化法を使うことで、なぜ、(2)で述べたような問題が解決できるのかを、200字以内で説明せよ。

(4) 2進数(バイナリ符号)を、(3)で求めたような、隣り合う符号間で変化するビットが1ビットしかないような符号へと変換するエンコーダ回路を設計することを考える。このとき、以下の問いに答えよ。ただし、設計した回路は、NOT、AND、OR、XORのうち必要なゲートだけを使用して、できるだけ簡約化された回路図で示すものとする。

(a) 2ビットの2進数(バイナリ符号)を、(3)(a)で求めた、隣り合う符号間で変化するビットが1ビットしかないような2ビットの符号へと符号化する、2ビット幅のエンコーダ回路を設計せよ。ただし、入力00に対して00を、入力01に対しては01を出力するものとする。

(b) 同じく、3ビット幅のエンコーダ回路を設計せよ。

(c) Nビット幅のエンコーダ回路の構成法を示せ。

(5) 今度は、隣り合う符号間で変化するビットが1ビットしかないような符号を、対応する2進数(バイナリ符号)へと逆変換する、デコーダ回路を設計することを考える。ディスクから読み出したデータをこのデコーダ回路へと通すことにより、より使い勝手の良い、2進数(バイナリ符号)を直接出力するロータリーエンコーダを構成することができる。このとき、以下の問いに答えよ。

(a) (3)(a)で求めた、隣り合う符号間で変化するビットが1ビットしかないような2ビットの符号を、対応する2ビットの2進数(バイナリ符号)へと逆変換する、2ビット幅のデコーダ回路を設計せよ。

(b) 同じく、3ビット幅のデコーダ回路を設計せよ。

(c) Nビット幅のデコーダ回路の構成法を示し、なぜそれが正しいかを説明せよ。

【No. **3**】 以下の設問に答えよ。

図Ⅰは、n(≧ 2)要素をもつ二つの配列変数 a, b に対し、同じインデックス i をもつ要素どうしを比較し、a[i]が b[i]より小さい要素の数を調査する C 言語コードである。

```
for( c = 0, i = 0; i != n; i++ )
    if( a[i] < b[i] ) c++;
```

図Ⅰ

このコードを含むプログラムを 32-bit RISC-V(RV32I)アーキテクチャ向けコンパイラで処理した結果、図Ⅰの C 言語コードに対応する部分として図Ⅱのアセンブリコードが得られたとする。なお、x から始まるオペランドはレジスタを表し、x0 は常に値 0 が格納されている特殊なレジスタである。

さらに、次の条件・前提を設ける。

- 配列変数の各要素及びスカラ変数は、特に断りがない限り 1 ワード(32 ビット)のサイズをもつとする。また、変数宣言は別途なされているものと仮定する。
- 配列 a 及び b それぞれの先頭アドレスは、x8 及び x9 に格納済みであるとする。
- n の値も x19 に格納済みであるとする。

行	ラベル	opcode	オペランド	意味
1		add	x6, x0, x0	
2		add	x18, x0, x0	
3	LOOP:	slli	x7, x6, 2	(x6 を 2 ビット左シフトした値を x7 に格納)
4		add	x28, x8, x7	(x8 と x7 の和を x28 に格納)
5		lw	x30, 0(x28)	(x28 の値が示すアドレスに格納されている値を x30 に転送)
6		add	x29, x9, x7	
7		lw	x31, 0(x29)	
8		slt	x5, x30, x31	(x30 < x31 ならば x5 に 1 を、そうでなければ 0 を格納)
9		beq	x5, x0, SKIP	(x5 の値が 0 と等しいなら SKIP に分岐)
10		addi	x18, x18, 1	(x18 と即値 1 の加算結果を x18 に格納)
11	SKIP:	addi	x6, x6, 1	
12		bne	x6, x19, LOOP	(x6 の値が x19 と異なるなら LOOP に分岐)

図Ⅱ

(1) 図Ⅱのアセンブリコードについて、以下の問いに答えよ。

(a) 変数 i，c，a[i]，b[i]の値を格納する用途に用いられているレジスタはどれか、それぞれ答えよ。

(b) 図Ⅱのアセンブリコードを実行した際の総実行命令数はいくつか、n の式で答えよ。ただし、全配列要素のちょうど半分にあたる $\frac{n}{2}$ 個の要素に対して a[i] < b[i] が成り立つ、すなわち、n は偶数とし、コード実行後の変数 c の値は $\frac{n}{2}$ となるものと仮定する。

(2) 図Ⅱのアセンブリコードを、5 ステージから成る 1-way(1 本)の命令パイプラインで実行することを考える。なお、各ステージを $s1, s2, \ldots, s5$ と呼ぶものとし、最初のステージを $s1$、最後のステージを $s5$ とし、1 ステージが 1 クロックサイクルで実行されるものとする。また、これ以降、以下の(ア)～(エ)に示す理想的な条件が成り立つと仮定する。

(ア) キャッシュミスは発生しないものとし、lw 命令におけるメモリアクセスレイテンシは考慮しなくてよいものとする。

(イ) 命令間のデータ依存は、フォワーディングにより適切に解決され、データハザードによるパイプラインバブルは発生しないものとする。

(ウ) 図Ⅱのアセンブリコードを実行する範囲では、12 行目の bne 命令において、分岐予測ミスは発生しないものとする。

(エ) 分岐予測ヒット時には、パイプラインバブルは発生しないものとする。

このとき、以下の問いに答えよ。

(a) 条件分岐命令とその後続命令の間には、制御依存が存在するため、分岐予測ミス時には制御ハザード(パイプラインハザード)が発生する。条件分岐命令の $s3$ ステージでの処理完了により、後続命令が確定するとした場合、この制御ハザード 1 回当たりに後続命令の実行は何クロックサイクル遅れるか答えよ。

(b) 分岐予測器が、9 行目の beq 命令に対して「常に成立する」と予測する場合、(1)(b)及び(2)(a)で示した条件の下で図Ⅱのアセンブリコードを実行した際の総実行サイクル数を答えよ。

(c) (1)(b)の結果と(2)(b)の結果を用いて、この場合の CPI(cycles per instruction)を算出せよ。

ただし、$n = 8$ であるとし、計算においては小数第 3 位を四捨五入して小数第 2 位まで求めること。

(d) 分岐予測器が、9 行目の beq 命令に対して 1 ビット履歴による予測を行う場合、すなわち、「前回の分岐結果と同じとなる」と予測する場合、a，b のどのインデックスにおいて値が一致するかにより、総実行サイクル数が異なる。(1)(b)及び(2)(a)で示した条件下で総実行サイクル数の最も小さい場合と最も大きい場合を、n の式で答えよ。ただし、履歴の初期値、すなわち初回実行時のミス／ヒットは自由に定めてよいものとする。

⑶　条件分岐命令を削減することで、制御ハザードによる性能低下を抑制することを考えるに当たり、以下の問いに答えよ。

⒜　図Ⅱのアセンブリコードの９～10 行目にある２命令を、まとめて一つの命令で置き換えることを考えたい。どのような命令で置き換えることができるか答えよ。

ただし、opcode としては、図Ⅱのアセンブリコード中で使用されているもののうちから適切なものを選んで用いることとし、適切なオペランドとともに示すこと。

⒝　⑶⒜で行った置き換え後のアセンブリコードにおいて、CPI を答えよ。

ただし、$n = 8$ であるとし、計算においては小数第３位を四捨五入して小数第２位まで求めること。

⑷　次に、⑵で示した条件のうち(ア)を破棄し、データキャッシュのヒット／ミスによる性能の変化について考える。以下の問いに答えよ。

ただし、配列変数 a 及び b の先頭は、キャッシュブロック境界にアラインメントされており、また、初期状態において、キャッシュ内の全てのブロックが空であると仮定する。

⒜　データキャッシュは、プログラム中のデータ参照が「局所性」をもつとき有効に働く。この局所性は時間的局所性と空間的局所性に大別されるが、配列変数 a 及び b に対するデータ参照がもつのはそのいずれか、あるいは両方か、その理由とともに答えよ。

⒝　いま $n = 100$ であるとする。キャッシュブロックサイズが４ワード、８ワード、16 ワードのそれぞれの場合において、⑶⒜で行った置き換え後のアセンブリコードの CPI を答えよ。

ただし、簡単のため、lw 以外の命令のレイテンシは１クロックサイクルと仮定し、分岐予測ミスなども発生しないものとする。また、lw 命令のレイテンシはキャッシュヒット時には１クロックサイクル、キャッシュミス時には 10 クロックサイクルであると仮定する。

⑸　次に、図Ⅱのアセンブリコードを 2-way のパイプラインで並列実行するための静的命令スケジューリングについて考える。以下の問いに答えよ。

⒜　例えば、１行目及び２行目にある二つの命令間には依存関係が存在しないため、レジスタファイルの同時読み出し可能数や演算器の数に制限がないとすると、同時実行や逆順実行が可能である。このような粒度の並列性を何と呼ぶか答えよ。

⒝　図Ⅱのアセンブリコードを、2-way パイプラインで最大２命令まで同時実行するに当たり、まずは単純な方法として、アセンブリコード内の命令順に反しない範囲で、同時実行可能命令を最大二つずつ順にグループ化していった場合、本コードの全 12 命令が何グループで構成できるか、理由とともに答えよ。

⒞　次に、性能を向上させることを目的とし、プログラムの実行結果が変わらない範囲で、図Ⅱのアセンブリコード内の命令順を変更してよいとした場合では、何グループで構成できるか、理由とともに答えよ。

⒟　さらに、⑶⒜で行った書き換え後のアセンブリコードを対象にした場合、⑸⒞で示した条件の下、何グループで構成できるか、理由とともに答えよ。

【No. **4**】 以下の設問に答えよ。

C 言語で順列・組合せを生成するプログラムを考える。
ただし、本問においては、<stdio.h> が読み込まれているものとする。

⑴ まず、順列の生成を考える。順列を生成する関数として、例えば、次のようなものが作成できる。この関数を func(r, 0, 4, 1)と呼び出したときの出力を書け。ただし、r[]は大きさ 5 の int の配列である。

```
void func(int r[], int f, int n, int k) {
  int i;

  if (k > n) {
    for (i = 1; i <= n; i++) {
      printf("%d", r[i]);
    }
    printf("¥n");
    return;
  }
  for (r[k] = 1; r[k] <= n; r[k]++) {
    if (!(f & 1 << r[k]) && r[k] != k) func(r, f | 1 << r[k], n, k + 1);
  }
  return;
}
```

⑵ 本問及び次の⑶においては、順列は大きさ n + 1(n は正の整数)の int の配列 r[1]～r[n]に生成されるものとする(r[0]は使用しない)。1 から n までの整数を全て使った順列は、これらを連結することで n 桁の n + 1 進法の正の整数とみなすこともできる。本問では、n 桁の n + 1 進法の正の整数とみなしたときの最小のもの(例えば 6 桁であれば 1 2 3 4 5 6)から始めて、その次に大きいもの(同じ例では 1 2 3 4 6 5)、その次に大きいもの(同様に 1 2 3 5 4 6), ……, と、二つ以上の桁の数字を並べ替えて生成していくことにする。

ここで、r[1]～r[n]を、r[1]～r[k − 1]と r[k]～r[n]に分けて考える(k は 1 から n − 1 の整数)。すなわち、r[k]～r[n]の値を並べ替えて、r[1]～r[k − 1]の値が固定された条件での順列を生成する(k = 1 のときはどの要素の値も固定されない。)。この条件では、r[k]には n − k + 1 種類の数字が入る可能性がある。この可能性を、r[k]～r[k], r[k]～r[k + 1],

r[k]～r[k + 2]，……，r[k]～r[n]を回転させて(例えばr[k]～r[n]を回転させるならば、r[k]の値をr[k + 1]に、r[k + 1]の値をr[k + 2]に、……、r[n − 1]の値をr[n]に移し、r[n]の値はr[k]に移して)それぞれ生成する。これによって生成されたそれぞれのr[k]～r[n]から成る順列は、r[k]がその数字であるとき、r[k]～r[n]に含まれる数字を使った順列のうち、最小のものになる。これらそれぞれについて、さらにr[k]の値を固定し、r[k + 1]～r[n]に対して再帰的に順列を生成させればよい。例えば、6個の数全てを使った順列を生成する際、k = 4のとき、図Iのように三つの場合のr[k]～r[n]が準備され(破線部分が回転によって順序を変えられた要素)、それぞれk = 5について再帰呼び出しがなされる。

r[1]		r[k − 1]	r[k]		r[n]
1	2	3	4	5	6
1	2	3	5	4	6
1	2	3	6	4	5

図I

この方針で、n個の数字を全て使った順列を、n桁のn + 1進法の正の整数とみなしたときの昇順に、全て出力させる関数を次のように作成した。例えば、1から6まで全てを使った順列を出力させるには、r[1] = 1，r[2] = 2，……，r[6] = 6を代入して用意した上でprintPerm(r, 6, 1)と呼び出す(初回の呼び出しでは第3引数は1とする。)。㋐～㋓に適切な式を入れよ。

```
void printPerm(int r[], int n, int k) {
  int i, j, t;

  if ( ㋐ ) {
    for (i = 1; i <= n; i++) {
      printf("%d", r[i]);
    }
    printf("¥n");
    return;
  }
  for (i = k; i <= n; i++) {
    t = r[i];
    for (j = i; j > k; j--) ㋑ ;
```

```
    r[k] = t;
    printPerm(r, n, [ ㋒ ]);
    for (j = k; j < i; j++) [ ㋓ ];
    r[i] = t;
  }
  return;
}
```

(3) 一方で、一度で全ての順列を出力させるのではなく、呼び出しごとに「次に大きい順列」を生成する関数を考える。例えば、「3 1 5 4 2」の次に大きい順列である、「3 2 1 4 5」を求めることを考える。ここで、先頭から見て最初に数字が変わる桁である r[2] より後ろは、元の順列では降順になっている。一般の場合にも、次の順列を生成するとき、先頭から見て最初に数字が変わる桁(r[k] とする。)より後ろは降順になっていることが確かめられる。そして、「次に大きい順列」は、この部分を昇順に並べ替えた後、r[k] の値より大きく r[k] の値に最も近い値をもつ要素 r[i] を見付け、r[k] と r[i] の値を入れ替えればよい。先ほどの例では、「3 1 5 4 2」を「3 1 2 4 5」に並べ直した後、r[2] である 1 と、それより大きく最も近い値の r[3] の 2 を入れ替え、「3 2 1 4 5」と生成できる。

この方針で、呼び出す度に r[1]～r[n] の順列の「次に大きい順列」を r[1]～r[n] を変更して作る関数を次のように作成した。㋐～㋖に適切な式又は式文を入れよ。

ただし、㋐、㋑には単一の式が、㋕、㋖には一時的な変数 t を用いた複数の式文が補充される。ここで、r[1]～r[n] には順列が用意されていなければならないため、順列を昇順に全て生成させていくときは r[1] = 1, r[2] = 2, ……, r[n] = n と初期設定して呼び出す。

```
void nextPerm(int r[], int n) {
  int i, k, t, s, e;

  for (k = n - 1; k > 0 && [ ㋐ ]; k--)
    ;
  for (s = k + 1, e = n; s < e; s++, e--) {
    [ ㋕ ]
  }
  if (k == 0) return;
  for (i = k + 1; [ ㋑ ]; i++)
    ;
```

```
    [ ㋖ ]
    return;
  }
```

⑷ 次に、組合せを生成することを考える。本問及び次の⑸においては、1 ～n から m 個(m は 1 ～n の整数)を選ぶ組合せを大きさ m + 1 の int の配列 r[]の r[1]～r[m]に対して生成する。このとき、r[1] < r[2] < … < r[m]が成り立つように生成する。これも、r[1]～r[m]を連結することで m 桁の n + 1 進法の正の整数とみなせる。

1 ～n から m 個を選ぶ組合せを全て列挙するには、ある桁 r[k](k は 1 から m − 1 の整数)より r[k + 1]が大きくなるような r[1]～r[m]を生成するように、for ループを含む関数を再帰的に呼び出せばよい。

この、1 ～n から m 個を選ぶ組合せを全て出力する関数を次のように作成した。この関数は、例えば 1 ～10 から 4 個を選ぶ組合せを出力したい場合、r[0] = 0 として printComb(r, 10, 4, 1)と呼び出す(初回の呼び出しでは第 4 引数は 1 とする。)。㋕～㋷に適切な式を入れよ。

```
void printComb(int r[], int n, int m, int k) {
  int i;

  if ( [ ㋕ ] ) {
    for (i = 1; i <= m; i++) {
      printf("%d", r[i]);
    }
    printf("¥n");
    return;
  }
  for (r[k] = [ ㋖ ] ; r[k] <= [ ㋗ ] ; i++) {
    r[k] = i;
    printComb(r, n, m, [ ㋷ ] );
  }
  return;
}
```

（下書き用紙）

（下書き用紙）

（下書き用紙）

（下書き用紙）

⑸ 組合せについても、⑶と同様に、呼び出すごとにr[1]～r[m]の「次に大きい組合せ」をr[1]～r[m]を変更して生成することを考える。

例えば、1～10から4個選ぶ場合、「1 4 6 7」の次は「1 4 6 8」であり、また、「1 4 9 10」の次は「1 5 6 7」である。これは、r[]を末尾から一つずつ見ていき、最初に見付けたところの値を増やせる余地のある要素(r[i]とする。)が1増やされることを示している。また、r[i]より後ろの要素は、r[i]の値が増やされた後で取り得る最も小さい値になる。

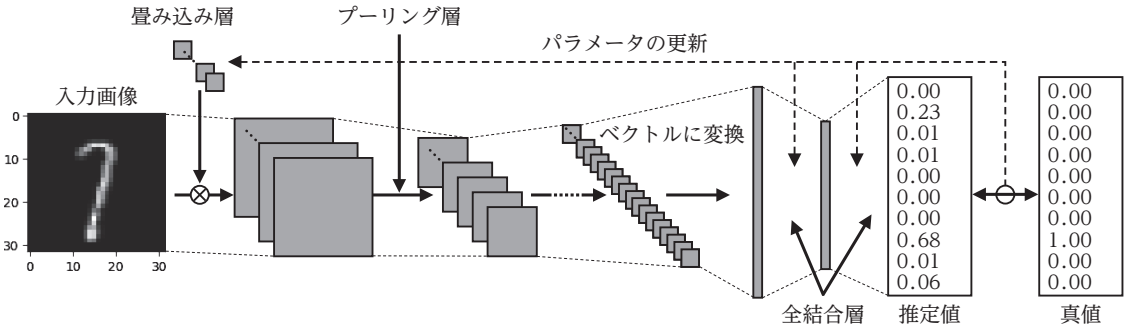
この方針で、呼び出す度にr[1]～r[m]の組合せの「次に大きい組合せ」をr[1]～r[m]を変更して作る関数を次のように作成した。㋐、㋑、㋒に適切な式を入れよ。ここで、r[1]～r[m]には組合せが用意されていなければならないため、組合せを昇順に全て生成させていくときはr[1] = 1, r[2] = 2, ……, r[m] = mと初期設定して呼び出す。また、この関数は成功した場合は非ゼロを、失敗した場合はゼロを返す。

```
int nextComb(int r[], int n, int m) {
  int i;

  for (i = m; i > 0 && r[i] == [ ㋐ ] ; i--)
    ;
  if (i == 0) return 0;
  r[i] = [ ㋑ ] ;
  for (i++; i <= m; i++) {
    r[i] = [ ㋒ ] ;
  }
  return !0;
}
```

【No. **5**】 以下の設問に答えよ。

0 ～ 9 までの 10 種の数字の画像を認識するニューラルネットワーク(図Ⅰ)について考える。データセットとして、32 × 32 画素の手描き数字の画像と、真値のラベルとしてその画像に描かれている数字のラベル 0 ～ 9 との対が複数与えられているとする。ネットワークに画像を入力すると、畳み込み層とプーリング層の処理を数回繰り返すことにより特徴が抽出され、それをベクトルとして全結合層を数回経て推定値のベクトルを得る。10 種の数字の場合、10 次元のベクトルが出力される。入力画像に対応した真値のラベルに対応した成分だけ 1 で他の成分は 0 となるような真値のベクトル(one-hot ベクトル)と比較し、その差が小さくなるように畳み込みや線型結合のパラメータを更新することにより学習が行われる。



図Ⅰ

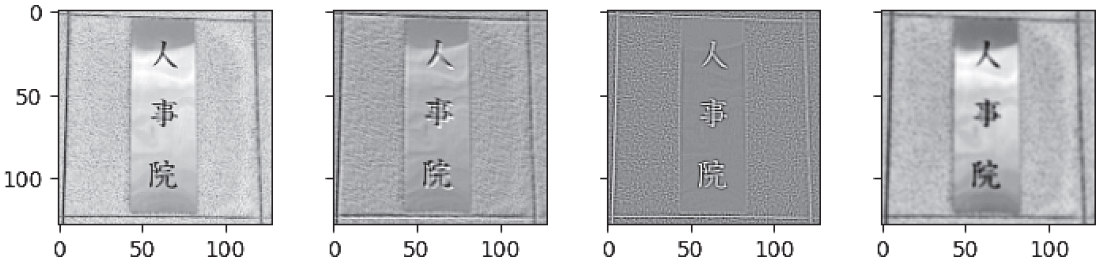
(1) 画像サイズを $W \times H$ 画素(横×縦)とし、画素をインデックス $(i, j)(i = 0, \ldots, W-1, j = 0, \ldots, H-1)$ で表す。画素 (i, j) の画素値を x_{ij} と表し、負の値を含む実数値をとるとする。フィルタと呼ぶサイズの小さい画像を考え、そのサイズを $W_f \times H_f$ 画素(横 × 縦)とする。フィルタの画素はインデックス (p, q) $(p = 0, \ldots, W_f - 1, q = 0, \ldots, H_f - 1)$ で表し、画素値を h_{pq} と書く。h_{pq} は x_{ij} と同様に任意の実数値をとる。画像の畳み込みは、画像とフィルタ間で定義される次の積和演算である。以下の問いに答えよ。

$$u_{ij} = \sum_{p=0}^{W_f-1} \sum_{q=0}^{H_f-1} x_{i+p,\, j+q} h_{pq}$$

(a) このとき、次に示す元画像(128 × 128)に次のフィルタ 1 , 2 , 3 (3 × 3)を畳み込んで得られる画像は画像Ⅰ, Ⅱ, Ⅲのそれぞれどれか、3 行程度の理由とともに述べよ。

ただし、画素の値は画像中の最大値の画素が白、最小値の画素を黒になるようなグレースケールで表示している。

フィルタ 1 : $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$、フィルタ 2 : $\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$、フィルタ 3 : $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



元画像　画像Ⅰ　画像Ⅱ　画像Ⅲ

(b)　畳み込みは画像にフィルタを重ねたとき、画像とフィルタの重なり合う画素どうしの積を求めて、それらの和を求める計算であるが、画像からフィルタがはみ出すような位置に重ねることはできない。画像サイズを $W \times H$ 画素とし、フィルタのサイズを $W_f \times H_f$ 画素としたときに、画像内にフィルタ全体が収まる範囲でフィルタを動かすと、出力画像のサイズはいくつになるか。

(c)　入力画像の外側に画素を加えることにより、畳み込み結果の画像が入力画像と同じサイズになるようにすることができる。入力画像の左側に $\left\lfloor \frac{W_f - 1}{2} \right\rfloor$、右側に $\left\lfloor \frac{W_f}{2} \right\rfloor$、上側に $\left\lfloor \frac{H_f - 1}{2} \right\rfloor$、下側に $\left\lfloor \frac{H_f}{2} \right\rfloor$ の幅の値が 0 の画素を加えて畳み込みを行うと、入力画像と同サイズの出力画像が得られる(ゼロパディング)。

ただし、$\lfloor x \rfloor$ は実数 x を超えない最大の整数を表す。ゼロパディングを行い次の入力画像にフィルタを畳み込んで得られる出力画像を示せ。

$$\text{入力画像：} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}\text{、フィルタ：} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(d)　同じサイズ($W \times H$)の画像 C 枚を層状に重ねた 3 次元配列 z_{ijc} を考える($c = 0, \ldots, C - 1$)。この層のことをチャンネルと呼び、その 3 次元配列のサイズを $W \times H \times C$ と書く(横 × 縦 × チャンネル数)。サイズが $W_f \times H_f \times C$ の 3 次元配列のフィルタを C_{out} 個用意し h_{pqck} とする($k = 0, 1, \ldots, C_{\text{out}} - 1$)。$W \times H \times C$ のサイズの入力画像に k 番目のフィルタを畳み込む演算は次のように表され、$W \times H \times 1$ の出力が得られる。

$$u_{ijk} = \sum_{c=0}^{C-1} \sum_{p=0}^{W_f-1} \sum_{q=0}^{H_f-1} z_{i+p,\, j+q,\, c}\, h_{pqck} + b_k$$

このフィルタ h_{pqck} の要素数とバイアス b_k の個数の和を求めよ。

(2)　サイズ $W \times H \times C$ の入力画像について、s を正の整数としたときに、座標(si, sj)を左上端とする $W_p \times H_p$ の領域から、チャンネル毎に独立に一つの画素値を次のように求める操作を最大プーリングと呼び、s をストライドと呼ぶ。以下の問いに答えよ。

$$u_{ijc} = \max_{0 \le p < W_p,\, 0 \le q < H_p} z_{si+p,\, sj+q,\, c}$$

(a) 次の入力画像に対して、$s = 2$, $W_p = 2$, $H_p = 2$ の最大プーリングを適用したとき、出力画像を示せ。

$$\text{入力画像：} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

(b) 最大プーリングはどのような性質を有し、画像のどのような変動に対して効果的に働くか３行以内で述べよ。

(3) 全結合層において、P 個の値 $z_1, \ldots, z_P$ から次の Q 個の値 $u_1, \ldots, u_Q$ への変換は次のように書ける。

$$u_j = w_{0j} + \sum_{i=1}^{P} w_{ij} z_i$$

全結合層の重み係数とバイアスの個数の和を求めよ。

(4) 活性化関数 f により(2)の u_{ijc} と(3)の u_j はそれぞれ $z'_{ijc} = f(u_{ijc})$, $z'_j = f(u_j)$ のように変換される。活性化関数 f としては例えば次のような関数が用いられる。以下の問いに答えよ。

$$\text{ReLU：} \quad f(x) = \max(0, x)$$

$$\text{SoftPlus：} f(x) = \log(1 + e^x)$$

$$\text{Sigmoid：} f(x) = \frac{1}{1 + e^{-x}}$$

$$\text{Tanh：} \quad f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(a) これらの関数のグラフの概形をそれぞれ描け。
ただし、漸近線が存在する場合は、その漸近線も示せ。

(b) 上記のような活性化関数がなぜ必要になるのか、３行以内で述べよ。

(5) ネットワークの最終段階において、その前段階の出力を $\{u_k\}$ $(1 \leqq k \leqq K)$ とするとき、次式で表される Softmax 関数を適用して得られた値 $\{t_k\}$ $(1 \leqq k \leqq K)$ 中で最大値をとるラベル k を、入力された画像について推定されたラベルとする。以下の問いに答えよ。

$$t_k = \frac{e^{u_k}}{\sum_{j=1}^{K} e^{u_j}}$$

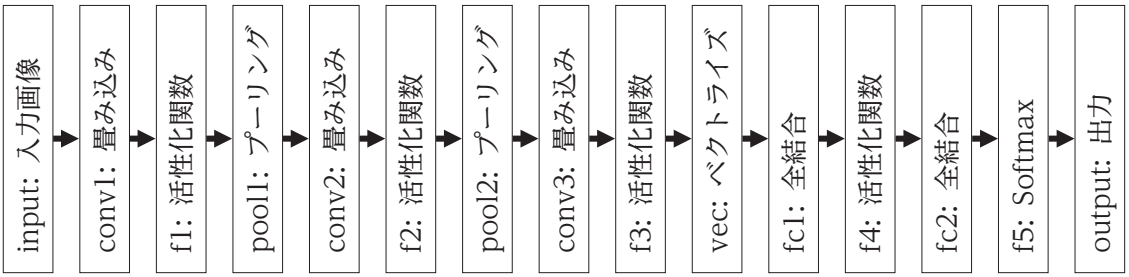
(a) Softmax 関数はどのような性質を満たし、その出力はどのように解釈することができるか３行以内で述べよ。

(b) Softmax 関数で出力された推定値を評価するために損失関数として交差エントロピー

$$E = -\sum_{k=1}^{K} y_k \log t_k$$

が用いられる。ここで、$\{y_k\}$ $(1 \leqq k \leqq K)$ は k が真値のラベルである場合の one-hot ベクトルとする。このとき、$\{t_k\}$ がどのような場合に E の値は最小になるか。理由を含めて３行以内で述べよ。

(6) 図Ⅰで示された処理の実装の一例は、次のようになる。空欄を埋めよ。㋐～㋔には正の整数の積(横 × 縦 × チャンネル数)、Ⓐ～Ⓗには正の整数が入り、同じ記号には同じ値が入る。ただし、前の処理の出力を入力として処理するものとする。



処理の名称と内容	出力サイズ	パラメータ数
input: 入力画像	$(32 \times 32 \times 1)$	0
conv1: 畳み込み($W_f = 5$, $H_f = 5$, $C_{\text{out}} = 6$, パディング無し)	㋐	Ⓓ
f1: 活性化関数 f	㋐	0
pool1: 最大プーリング($s = 2$, $W_p = 2$, $H_p = 2$)	㋑	0
conv2: 畳み込み($W_f = 5$, $H_f = 5$, $C_{\text{out}} = 16$, パディング無し)	㋒	Ⓔ
f2: 活性化関数 f	㋒	0
pool2: 最大プーリング($s = 2$, $W_p = 2$, $H_p = 2$)	㋓	0
conv3: 畳み込み($W_f = 5$, $H_f = 5$, $C_{\text{out}} = 120$, パディング無し)	㋔	Ⓕ
f3: 活性化関数 f	㋔	0
vec: ベクトライズ(多次元配列を 1 次元のベクトルに変換)	Ⓐ	0
fc1: 全結合($P =$ Ⓐ , $Q = 84$)	Ⓑ	Ⓖ
f4: 活性化関数 f	Ⓑ	0
fc2: 全結合($P =$ Ⓑ , $Q = 10$)	Ⓒ	Ⓗ
f5: Softmax 関数	Ⓒ	0
output: 出力	Ⓒ	0

(7) データセットを、50000 対の学習用のデータセットと 10000 対の検証用のデータセットに分割し、前者で学習して後者で評価を行った。10000 対の検証用のデータセットでの混同行列を次の表に示す。この結果について定量的評価を含めて 5 行以内で述べよ。

		推定結果ラベル									
		0	1	2	3	4	5	6	7	8	9
真値ラベル	0	991	0	2	0	0	1	2	1	3	0
	1	0	991	1	0	1	2	4	0	1	0
	2	1	1	994	1	1	0	0	1	1	0
	3	0	0	2	992	0	2	0	1	2	1
	4	0	0	0	1	987	0	4	1	0	7
	5	1	0	0	18	1	973	4	1	0	2
	6	4	2	1	0	1	1	989	0	2	0
	7	0	2	7	3	0	0	0	985	1	2
	8	1	0	3	4	0	0	0	3	988	1
	9	0	1	0	0	6	4	0	3	0	986

【No. **6**】 セキュリティを守るために様々なシステムで利用される暗号学的ハッシュ関数に関する以下の設問に答えよ。

(1) 次のハッシュ関数のもつべき性質に関して述べた文章について、以下の問いに答えよ。

「ハッシュ関数は、任意長のビット列を固定長のビット列に変換する関数である。ハッシュ関数の出力のビット長をハッシュ長と呼び、以下では ℓ で表す。

ハッシュ関数がもつべき基本的な性質として一方向性(原像計算困難性)がある。ハッシュ関数は、全ての取り得るハッシュ値が同じ頻度で出力されるように設計されるが、その場合、［ ① ］個の入力に対してハッシュ値を計算すれば、50 % より高い確率で特定のハッシュ値をもつ入力を得ることができる。したがって、［ ① ］が十分大きいことが、一方向性をもつための必要条件となる。

利用方法によっては、ハッシュ関数が衝突耐性をもつことも必要である。衝突耐性とは、同じハッシュ値をもつ入力のペア(衝突)を見付けることが事実上不可能であることであり、衝突耐性に対する攻撃として、バースデーパラドックスを利用したバースデー攻撃がある。

教室に d 人の生徒がいる場合、同一の誕生日をもつ生徒がいる確率は、$d = 3$ であれば［ ㋐ ］、$d = 4$ であれば［ ㋑ ］であり、一般の d に対して、この確率は $1 - e^{-\frac{d^2}{2\times 365}}$ で近似できることが知られている(e は自然対数の底である。)。したがって、$d = 19 \fallingdotseq \sqrt{365}$ の場合には同一の誕生日をもつ生徒がいる確率は $1 - e^{-\frac{1}{2}} \fallingdotseq 39$ % となり、$d = 59 \fallingdotseq 3.1 \times \sqrt{365}$ の場合には確率は $1 - e^{-\frac{3.1^2}{2}} > 99$ % と、直感に反した高い確率となる。

同様に考えることで、任意のハッシュ関数に対して、$d = O($［ ② ］$)$個の異なる入力に対してハッシュ値を計算することにより、高い確率で衝突を見付け出すことができる。このような戦略で衝突を見付ける方法を、バースデー攻撃と呼ぶ。約 39 % の確率でこの攻撃を成功させるには、［ ③ ］回のハッシュ関数の計算が必要であり、計算したハッシュ値を全て保存しておく必要があるため、［ ④ ］個の〈入力、ハッシュ値〉のペアを保存するメモリ領域が必要である。これに対し、バースデー攻撃における必要なメモリ領域を $O($［ ⑤ ］$)$に削減する方法も知られている。この方法では、ハッシュ関数 H に対して、まず、適当な $a_0 = b_0$ から $a_i = H(a_{i-1})$, $b_i = H(H(b_{i-1}))$ を $i = 1, 2, 3, \ldots$ に対して計算していき、$a_j = b_j$ となる j を探索する。次に、a_0 と $c_0 = a_j$ から $a_i = H(a_{i-1})$, $c_i = H(c_{i-1})$ を $i = 1, 2, 3, \ldots$ に対して計算していき、$a_k = b_k$ となる k を探索する。

そのほかのハッシュ関数の性質として、第二原像困難性がある。この性質をもつハッシュ関数は、一つの入力が与えられたときに、それと同じハッシュ値をもつ別の入力を求めることが事実上不可能である。」

(a) ①～⑤に当てはまる最も妥当なものを[1，$\log \ell$，$\sqrt{\ell}$，ℓ，ℓ^2，$2^{\frac{\ell}{2}}$，2^ℓ]から選択せよ。

ただし、同じ記号には同じ式が入り、異なる記号にも同じ式が入る可能性がある。また、当てはまるものが複数あるときは、最も小さいオーダを示すこと。

(b) $N = 365$ とする。㋐と㋑に入る値を N を用いて表せ。ただし、うるう年は無視する。

(2) ハッシュ関数の利用方法の一つに、パスワードファイルにおけるパスワードの保護がある。パスワードファイルの漏洩に備え、パスワードファイルには《ユーザ名、パスワードのハッシュ値》を保存することが考えられる(これを方法 1 とする。)が、一般には、《ユーザ名、ソルト、ソルトとパスワードの連結のハッシュ値》を保存する(これを方法 2 とする。)。

ただし、ソルトとは、ユーザごとに独立に選ばれたランダムな値である。

以下では、ソルトは十分長いビット列とする。また、各ユーザは既知のパスワード集合 D から独立一様ランダムにパスワードを選択して利用しているものとし、ハッシュ長 ℓ に対して $2^\ell \gg |D| \gg 1$ を仮定して概算してよい。以下の問いに答えよ。

(a) パスワードファイルの漏洩時に他者によるなりすましを防ぐために、ハッシュ関数に必要とされる性質は、一方向性、衝突耐性、第二原像困難性のうちのどれか。また、その理由を４行程度で説明せよ。複数の性質が必要な場合は、最も重要な性質について答えよ。

(b) ハッシュ関数が(a)で答えた性質をもつとし、パスワードファイルが方法 1 で作成されているとする。このとき、パスワードファイルを手に入れた攻撃者が、ある特定のユーザのパスワードを求めるために必要なハッシュ関数の計算回数の期待値を、理由とともに答えよ。また、パスワードファイルに M 人のユーザの情報が登録されているときに、攻撃者がユーザのうちの誰か一人のパスワードを求めるために必要なハッシュ値の計算回数の期待値を、理由とともに答えよ。

(c) (b)において、パスワードファイルが方法 2 で作成されているとする。このとき、(b)のそれぞれの期待値を、理由とともに答えよ。

(3) ハッシュ関数の利用方法の一つに、データ検証への応用がある。データ F を外部ストレージに保存したいデータ保持者がいたとする。最も単純な方法では、データ保持者は F を外部ストレージに保存する前に、ハッシュ値 $h = H(F)$ を計算してローカルストレージに保存する。

ただし、H はハッシュ関数である。データ保持者が F を利用したい場合には、外部ストレージから F をダウンロードし、ハッシュ値 $h' = H(F)$ を再計算し、ローカルストレージに保存してある h と比較することで、データが改変されていないことを検証できる。

データ F が大きく、データ保持者がデータを部分ごとにしか利用しない場合、データを小さなブロックに分割しハッシュ木(マークル木)を作成することによって、データ利用時の通信効率を上げることができる。アルゴリズム 1 は、データを 4 分割($N = 4$)した場合のハッシュ木の作成方法を説明している。一般の N の場合は、N 個の葉ノードをもつ高さ最小の二分木を一つ

定め、各ブロック F_i のハッシュ値 $h_i = H(F_i)$ を各葉ノードに割り当てる。各内部ノードについては、二つの子ノードのもつハッシュ値の連結に対するハッシュ値を割り当てる。ルートノードに割り当てられたハッシュ値 h_{root} をルートハッシュと呼ぶ。

ハッシュ木を使ったデータ検証では、データ保持者はデータ F を外部ストレージに保存する前に N 個のブロックに分割してハッシュ木を作成し、ルートハッシュ h_{root} のみをローカルストレージに保存する。全ブロックをもつ外部ストレージは、ハッシュ関数を用いてハッシュ木を再作成できることに注意する。データ保持者がブロック F_i を利用するとき、外部ストレージは F_i をデータ保持者へ送ると同時に、F_i を用いてルートハッシュを再計算するために必要な最低限のハッシュ値もデータ保持者へ送る。データ保持者は、受け取った情報からルートハッシュを再計算し、ローカルストレージ内のルートハッシュと比較することで、受け取ったブロック F_i が改変されていないことを検証できる。

アルゴリズム 1　ハッシュ木の作成方法（$N = 4$ の場合）

データ F を四つに分割したブロックを F_0，F_1，F_2，F_3 とする。

Step 1：4 個の葉ノードをもつ完全二分木を考え、各葉ノードに各データ F_i のハッシュ値 $h_i = H(F_i)$ を割り当てる（$i = 0, 1, 2, 3$）。

Step 2：葉ノードを子ノードとしてもつノードに $h'_i = H(h_{2i} \mid h_{2i+1})$ を割り当てる（$i = 0, 1$）。ただし、$\mid$ はビット列の連結を表す。

Step 3：ルートノードに $h_{\mathrm{root}} = H(h'_0 \mid h'_1)$ を割り当てる。

以下の問いに答えよ。

(a)　外部ストレージによるデータの改変を検出するために、ハッシュ関数に必要とされる性質は、一方向性、衝突耐性、第二原像困難性のうちのどれか。また、その理由を 4 行程度で説明せよ。複数の性質が必要な場合は、最も重要な性質について答えよ。

(b)　データを 4 個のブロックに分割する場合を考える（$N = 4$）。データブロック F_1 を利用するときに、外部ストレージが F_1 と共にデータ保持者に送るハッシュ値はどれか。アルゴリズム 1 の記号を用いて答えよ。

(c)　データを N 個のブロックに分割する場合を考える。データ保持者が一つのブロックを利用するときに、外部ストレージがデータ保持者に送るハッシュ値の個数を N で表せ。ただし、ハッシュ木は完全二分木と仮定してよい。

(d)　データ F を 1 GB、ハッシュ長を $\ell = 128$ ビット（16 バイト）とし、データを N 個のブロックに分割するとする（全てのブロックは同じ長さとする。）。データ保持者が F の中のいずれかの 1 ビットのみを利用する場合、外部ストレージがデータ保持者に送る総通信量［バイト数］の最大値を N で表せ。

⑷　ハッシュ関数の応用として、デジタル署名がある。代表的な署名方式としては、次のように記述される RSA 署名がある。以下の問いに答えよ。

ここで、H はハッシュ関数、$a \bmod b$ は整数 a を b で割った余りを表す。

秘密鍵：(N, d)。ただし、N は二つの素数 p，q の積とする。

公開鍵：(N, e)。ただし、e，d は $ed \bmod (p-1)(q-1) = 1$ を満たす正の整数とする。

署名生成：メッセージ m に対して $\sigma = H(m)^d \bmod N$ を計算し、σ を署名とする。

署名検証：$\sigma^e \bmod N$ と［　Ⓐ　］が等しいことを確認する。

⑷　空欄Ⓐに入る適切な式を答えよ。

⑹　デジタル署名で利用するハッシュ関数に必要とされる性質は一方向性、衝突耐性、第二原像困難性のうちのどれか。また、その理由を 4 行程度で説明せよ。複数の性質が必要な場合は、最も重要な性質について答えよ。

⑹　デジタル署名に関する次の記述は誤りである。どのような誤りがあるかを 3 行程度で説明せよ。

「大規模な汎用量子コンピュータが実用化されると、RSA 署名は安全ではなくなる。これは、量子コンピュータによってハッシュ関数の安全性が破られるためである。」

科目別構成の詳細

科　　目	出題数	問題番号	ページ
計算機科学	1題	No. **1**	1～2
情報工学(ハードウェア)	2題	No. **2**, **3**	3～7
情報工学(ソフトウェア)	2題	No. **4**, **5**	8～17
情報技術	1題	No. **6**	18～21

○　**6題**のうちから**任意の2題**を選んで解答してください。