

情報 I

令和 6 年度 学年末考查

2025 年 2 月 3 日(月) 2 校時

100 点満点 / 試験時間 60 分

年	組	番号	名前
1			

第 1 問

コンピュータは、さまざまなハードウェアとソフトウェアによって構成されている。中心的な処理を担うのが **ア**（中央処理装置）であり、これはプログラムの命令を解釈し、計算や処理を行う役割を持つ。**ア** が処理を行う際、一時的にデータを保持するのが **イ** 装置である。この装置は、電源を切ると内容が消えてしまい、一時的なデータの保持に過ぎない。電源を切ってもデータを保持できる記憶装置として **ウ** 装置がある。

コンピュータとユーザーがやり取りを行うための装置が必要である。たとえば、キーボードやマウスは **エ** 装置 に分類され、ディスプレイやプリンタは **オ** 装置に分類される。このようにコンピュータは、さまざまな装置で構成されている。

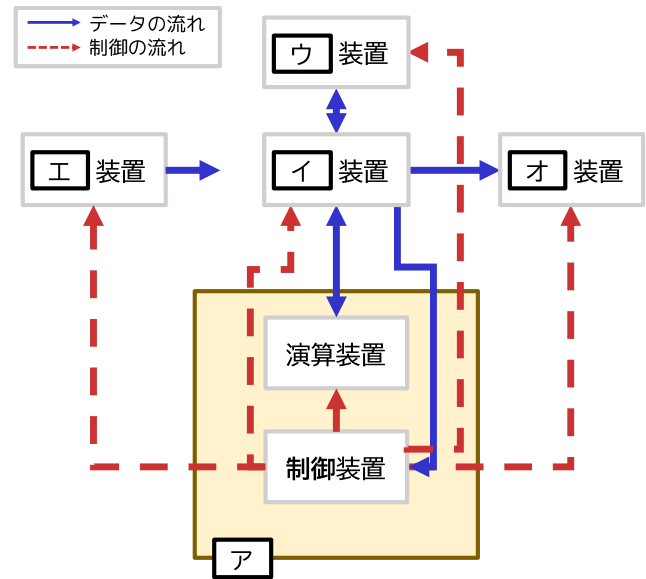


図 1 コンピュータを構成するさまざまな装置

問 1: **ア** ～ **オ** に入る適当なものを、以下の選択肢から一つずつ選べ。

⑩仮想メモリ	①補助記憶	②クロック周波数	③主記憶
④入力	⑤キャッシュメモリ	⑥出力	⑦CPU
			⑧GPU

問 2: **ウ** 装置についての説明について、以下の特徴に当てはまるのはハードディスク・SSD のどちらかの組み合わせを、以下の選択肢から記号で答えよ。 **カ**

- ・アクセス速度が速い…………… **い**
- ・より安価である…………… **ろ**
- ・衝撃や振動に強い…………… **は**

カ の選択肢

	い	ろ	は
⑩	HDD	HDD	HDD
①	HDD	HDD	SSD
②	HDD	SSD	HDD
③	HDD	SSD	SSD
④	SSD	HDD	HDD
⑤	SSD	HDD	SSD
⑥	SSD	SSD	HDD
⑦	SSD	SSD	SSD

第 2 問 以下の図は、情報処理室のネットワーク構成図である。

問 1 図中 **ア** ～ **ウ** に当てはまる機器の名称を、説明文中の **エ** ～ **オ** に当てはまる語句を以下の選択肢から選べ。

図 2 のように、情報処理室にもネットワークがあることがわかる。このように限られた範囲で利用されるネットワークのことを、**エ** という。

例えば、PC(コンピュータ)と機器**ア**の間は **オ** で接続されていることがわかる。

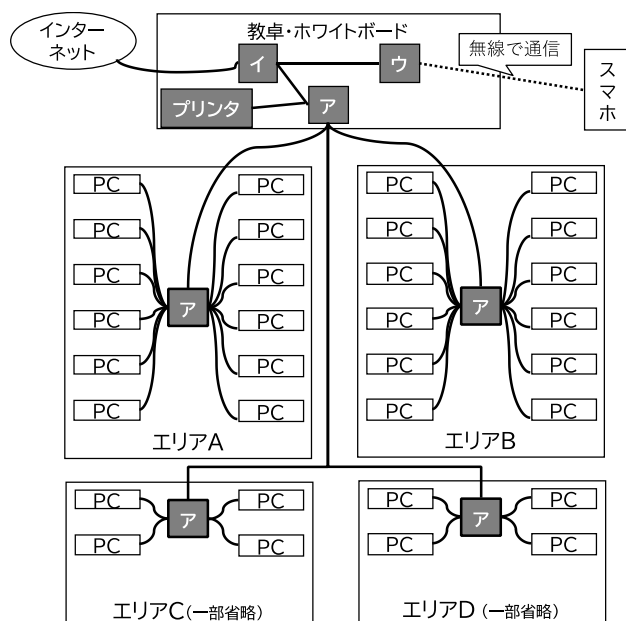


図 2 情報処理室のネットワーク構成図

ア ～ **オ** の選択肢

- | | | | | |
|------|---------|---------|---------|-----------|
| ①LAN | ①ルータ | ②Wi-Fi | ③無線 LAN | ④有線 LAN |
| ⑤WAN | ⑥無線 WAN | ⑦有線 WAN | ⑧ハブ | ⑨アクセスポイント |

問 2 情報処理室での授業中、以下のネットワークトラブルが発生した。

ケース A: エリア B の PC だけがすべてインターネットに接続できなくなった。

ケース B: 全ての PC がインターネットに接続できなくなったが、プリンタは動作する。

このとき、故障している可能性があると考えられるのはどれか。ケース A を **カ** に、ケース B を **キ** にそれぞれ選べ。

カ ～ **キ** の選択肢

- | | | |
|-----------------------|-----------------------|-----------------------|
| ①教卓にある機器 ア | ①エリア A にある機器 ア | ②エリア B にある機器 ア |
| ③エリア C にある機器 ア | ④エリア D にある機器 ア | ④機器 イ |

第 3 問

以下の S 先生と生徒 Q さんの会話文の、ア ～ ク、について、以下の選択肢から選べ。
また、ケ・コについては当てはまる数値を入れよ。

Q さん: 先生、インターネット上ではどのような処理がされているのでしょうか。

S 先生: 例えば「岩手県」というデータを送るとしましょう。そのデータをそのまま送るのではなく、データのある一定の小さい単位、つまり ア に分割して送ります。その後 ア は、接続されているルータを目指して進みます。ア はルータから、次目指すべきルータを教えてくれるのです。

Q: なるほど。ルータはなぜ次目指すべきルータの案内ができるのですか。

S: 自分がどのルータとつながっているか、最終的に辿り着く機器はどれか、などのルート情報を保持しているからです。この情報のことを イ といいますよ。

Q: それでは、そのパケットが最終的に目指している場所について、ルータはなぜ把握できているのでしょうか。

S: 実は ア には ウ と呼ばれるものがデータとは別に追加されていて、その情報の一部を読み取って案内をしているのです。宛先には エ とよばれる固有の番号を割り当てています。

Q: あれ、普段使っているインターネット上の宛先って、「https://www.pref.iwate.jp/」のようなものですね。

S: それは オ ですね。厳密に言うと、これは Web ページの場所を表していて、人間にわかりやすいようになっています。ただ、インターネット上では エ を使っているので、それを変換する必要があります。それを行う仕組みのことを、カ と呼びます。

その後、S さんは自分でインターネットの仕組みを調べているうちに、新たな疑問が湧いた。

Q さん: 先生、キ とはなんなのでしょう。

S 先生: インターネット上で通信するときに必要な手順や、形式などに関する決まりのことです。実は情報をやり取りするときにはたくさんの決まりがあり、それを 4 つの階層にして処理を行っているのですが、それを ク と呼びます。

表 1 ク の階層構造

名称	階層	
アプリケーション層	第 4 層	インターネットの通信の内容に応じた情報を追加する
トランスポート層	第 3 層	正しいデータを通信・受信するための情報を追加する
インターネット層	第 2 層	送信先の宛先に関する情報を追加する
ネットワーク インタフェース層	第 1 層	通信機器に関する情報や信号に関する情報を追加する

S: 先ほど、宛先の情報を [ウ] で追加する、という話をしましたが、それを行っているのが第 [ケ] 層になります。

Q: 他にはどんな情報を付与しているのですか？

S: 例えばデータを分割したうちの何番目のデータかなどを記録しています。

Q: なぜそのようなことをするのですか。

S: 実は [ア] は、同じ宛先を目指しているにも関わらず、違うルートを辿ることがあるのです。Qさんも学校に行くとき、いろいろなルートがあると思いますが、インターネットの世界も同じです。それで、ルートによっては遠回りになることもある。なので、最終的な宛先に辿り着いても、データがバラバラに辿り着くこともあるのです。

Q: つまり、届いたバラバラのデータを元に戻すためにそのような情報があるのですね。ちなみに、それは第 [コ] 層で行われていますか。

S: その通りです！

[ア] ~ [ク] の選択肢

- | | | | | |
|-----------|-----------|--------|--------------------|---------|
| ① URL | ① 通信プロトコル | ② パケット | ③ DNS | ④ 経路制御表 |
| ⑤ IP アドレス | ⑥ ヘッダ | ⑦ TCP | ⑧ インターネットプロトコルスイート | |

第 4 問

各文字をそれぞれ同じ文字数だけシフト(ずらして)して暗号を作成する暗号手法をとる。今回は、2 文字後ろにずらして暗号化を行うことを考える。なお、末尾の Z のあとは A に戻るものとする。(X を 3 文字ずらすときは、 $X \rightarrow Y \rightarrow Z \rightarrow A$ となる。)この時、以下の問いに答えよ。

問 1 以下の文章を暗号化してください。

(a) EASY

ア

(b) TOHOKU

イ

問 2 以下の文章を受け取ったとき、そのデータを元に戻してください。

(a) JCTF

ウ

(b) KYCVG

エ

問 3 以下の文章に当てはまる空欄を、下の選択肢から選べ。

暗号化したデータを元に戻す(2)の作業を、**オ** という。また、元のデータのことを **カ** と呼ぶ。今回「2 文字ずらす」という行為を行ったが、このように暗号化と **オ** のための具体的な手順のことを **キ** と呼ぶ。また、今回のように暗号化と **オ** では「2 文字ずらす」という同じ **キ** を使っているが、このようなことを **ク** と呼ぶ。

また、異なる **キ** を使う暗号方式として、公開鍵暗号方式というものがある。以下の手順で行われる。

- ① **ケ** をデータの送信者が受け取る
- ② データの送信者が、受け取った **ケ** を使って暗号化し、受信者にデータを送信する
- ③ 受信者は受け取った暗号化されたデータを、**コ** を使って **オ** する。

ア ～ **エ** の選択肢

- | | | | | |
|---------|----------|----------|--------|---------|
| ① GCUA | ① UPIPLV | ② FBTZ | ③ HARD | ④ IWAKI |
| ⑤ IWATE | ⑥ HAND | ⑦ VQJQMW | | |

オ ～ **ク** の選択肢

- | | | | | |
|------|------|-----------|-----------|----------|
| ① 鍵 | ① 平文 | ② 同一鍵暗号方式 | ③ 共通鍵暗号方式 | ④ セッション鍵 |
| ⑤ 復号 | ⑥ 復元 | | | |

ケ ～ **コ** の選択肢

- | | | | |
|-----------|-----------|-----------|-----------|
| ① 送信者の公開鍵 | ① 受信者の公開鍵 | ② 送信者の秘密鍵 | ③ 受信者の秘密鍵 |
|-----------|-----------|-----------|-----------|

第 5 問

以下はプログラミングの問題である。なお、プログラム言語は以下の通りになっている。

1. 変数

通常の変数例: `kosu kingaku_kei` (変数名は英語で始まる英数字と「_」の並び)

配列変数の例: `Tokuten[3]` (配列名は先頭文字が大文字)

配列の要素を指定する添字は 0 から始まる

2. 文字列

文字列はダブルクォーテーション(“)で囲む。+で連結できる。

`moji = "I'll be back."` `message = "祇園精舎の" + "鐘の声"`

3. 代入文

`a ← 10` `a` に 10 を代入

`Hairetsu ← {10,20,30,40,50}` `Hairetsu` に配列を代入

4. 算術演算

加減乗除の四則演算は、+、-、*、/ で表す

整数の除算では、商(整数切り捨て部分)を `÷` で、余りを `%` で表す。

`a ← 3, b ← 11` のとき、`b ÷ a` は 3 (`b ÷ a` の整数切り捨て部分)

`b % a` は 2 (`b ÷ a` の余り) `b / a` は 3.6666… (`b ÷ a` の整数切り捨て部分)

5. 比較演算

`a > b` (`a` は `b` より大きい), `a < b` (`a` は `b` より小さい)

`a ≥ b` (`a` は `b` 以上) `a ≤ b` (`a` は `b` 以下)

`a = b` (`a` と `b` は等しい) `a != b` (`a` は `b` と等しくない)

6. 論理演算

『かつ』(論理積) 『または』(論理和)

7. 条件分岐

もし `x ≥ 3` ならば

`x ← x + 1`

を実行し、そうでなければ

`y ← y * 2`

を実行する

8. 繰り返し(ループ)

`i` を 0 から 2 まで 1 ずつ増やしながら,

`i` を表示する

を繰り返す

`i ← 0`

`i ≤ 2` の間,

`i` を表示する

`i ← i + 1`

を繰り返す

問 1: 以下のプログラムについて、どのような数値が出力されるかを答えよ。 ア

1	$a \leftarrow 2$
2	$b \leftarrow 3$
3	a を表示する

問 2: 以下のプログラムについて、どのような数値が出力されるかを答えよ。 イ

1	$a \leftarrow 2$
2	$b \leftarrow 3$
3	$c \leftarrow a + b$
4	c を表示する

問 3: 以下のプログラムについて、どのような数値が出力されるかを答えよ。 ウ

1	$a \leftarrow 2$
2	$b \leftarrow 3$
3	$c \leftarrow 12$
4	$d \leftarrow a * b$
5	d を表示する

問 4: 以下のプログラムについて、どのような数値が出力されるかを答えよ。 エ

1	$\text{sum} \leftarrow 0$
2	i を 1 から 3 まで 1 ずつ増やしながら,
3	$\text{sum} \leftarrow \text{sum} + i$
4	を繰り返す
5	sum を表示する

問 5: 以下のプログラムについて、どのような数値が出力されるかを答えよ。 オ

1	$i \leftarrow 1$
2	$i \leq 6$ の間,
3	$i \% 2 \neq 0$ ならば
4	$\text{sum} \leftarrow \text{sum} + i$
5	を実行する
6	を繰り返す
7	sum を表示する

第 6 問

問 1: 以下のプログラムについて、円の面積を変数 `menseki` に代入したい。空欄に当てはまるプログラムを以下の選択肢から選べ。

1	<code>hankei ← 4</code>
2	<code>pai ← 3.14</code>
3	<code>menseki ← </code> <input type="text" value="ア"/>
4	“円の面積は”と <code>menseki</code> と “ cm^2 です” を表示する

の選択肢

① <code>hankei + pai</code>	① <code>hankei + hankei + pai</code>
② <code>hankei * pai</code>	③ <code>hankei * hankei * pai</code>

問 2: 以下のプログラムについて、3 つの数の平均を計算する変数 `heikin` に代入したい。空欄に当てはまるプログラムを以下の選択肢から選べ。

1	<code>kazu1 ← 4</code>
2	<code>kazu2 ← 7</code>
3	<code>kazu3 ← 12</code>
4	<code>heikin ← </code> <input type="text" value="イ"/>
5	“3 つの数の平均は”と <code>heikin</code> と “です” を表示する

の選択肢

① <code>kazu1 + kazu2 + kazu3 / 3</code>	① <code>(kazu1 + kazu2 + kazu3) / 3</code>
② <code>kazu1 + kazu2 + kazu3 * 3</code>	③ <code>(kazu1 + kazu2 + kazu3) * 3</code>

問 3: 以下のプログラムについて、繰り返しや条件分岐を使って「ダンダダン」という文字を表示したい。空欄に当てはまるプログラムを以下の選択肢から選べ。なお、 ・ ・ は数値を、 は選択肢から選べ。

1	<code>i</code> を 0 から <input type="text" value="ウ"/> まで 1 ずつ増やしながら、
2	もし <code>i = </code> <input type="text" value="エ"/> <code> </code> <input type="text" value="オ"/> <code> </code> <input type="text" value="カ"/> ならば
3	“ン” を改行なしで表示する
4	を実行し、そうでなければ
5	“ダ” を改行なしで表示する
6	を実行する
7	を繰り返す

の選択肢

① かつ	① または
------	-------

問 4: 以下のプログラムについて、配列に入っている要素全てを過不足なく表示したい。空欄に当てはまるプログラムを以下の選択肢から選べ。ただし、以下の関数を使うこと。

- 要素数(配列) … 引数に 配列 をとる。配列の要素数を戻り値とする関数。

例: 要素数({10,20,30,40}) の戻り値は 4。

1	Hairetsu ← {59,86,70,93,98,38,65,29,36,89,58,95,68,20,34}
2	i を 0 から <input type="text" value="キ"/> まで 1 ずつ増やしながら,
3	<input type="text" value="ク"/> を表示する
4	を繰り返す

の選択肢

① 要素数(i)	① 要素数(Hairetsu)	② 要素数(Hairetsu)-1
③ 要素数(Hairetsu[i])		

の選択肢

① i	① Hairetsu	② Hairetsu - i	③ Hairetsu[i]
-----	------------	----------------	---------------

問 5: 以下のプログラムについて、50%の確率で”あたり”が、残りの 50%の確率で”はずれ”をランダムに表示したい。空欄に当てはまるプログラムを以下の選択肢から選べ。ただし、以下の関数を使う。

- 実数乱数() … 0 以上 1 未満のランダムな数字を戻り値とする。

例: 実数乱数()の戻り値が 0.791 など

1	ransu ← 実数乱数()
2	もし <input type="text" value="ケ"/> ならば
3	”あたり” を表示する
4	を実行し、そうでなければ
5	”はずれ” を表示する
6	を実行する

の選択肢

① ransu < 0.49	① ransu < 0.50	② ransu ≤ 0.50
③ ransu < 49	④ ransu < 50	⑤ ransu ≤ 50

第 7 問

次の A さんと B さんの会話を読んで、設問に答えなさい。

A さん: コンピュータって、指示に対して絶対にミスなく処理できるのだよね

B さん: 実際はミスがあっても、修正できる能力を持っているみたいだよ。例えばインターネットでデータのやり取りをするときは、「パリティビット」と呼ばれる仕組みを利用してミスを検知している。

A: それはこういった仕組み？

コンピュータ内部では、データは 0 と 1 の羅列にしているというのは習ったよね。例えば「10101」というデータがあると思うけど、その中に 1 の個数は何個入っているかな？

B: 個ですね。

A: そう。このデータが送られるとき、パリティビットを使って、1 の個数が「奇数」か「偶数」かをチェックするんだ。今回は偶数パリティを使っていると仮定しよう。データを送る前に、「通信は偶数パリティを使いましょう」と受信側と送信側で決めておく。実際にデータを送る際は、元のデータ「10101」の末尾にパリティを追加して、全体のデータの 1 の個数が偶数個になるようにしておくんだ。

B: ということは、元のデータの 1 の個数が奇数だから、パリティビットは「」になるんですね。

A: その通り！ インターネットでは時々、データ送信中にビットが反転してしまうことがあるんだ。例えば元のデータにパリティビットを追加したデータが 1 つでも間違っ、例えば「101001」に変わったら、1 の個数は 3 個になって奇数になるよね。

B: あ、そうになると、偶数パリティには合わなくなりますね。

A: そう。このように、この仕組みはデータの誤りを ということになります。

B: パリティビットは簡単な仕組みだけど、データ通信においては重要なんだね。

問 1 ・ に当てはまる数字を答えよ。

問 2 に当てはまる言葉について、適切なものを選択肢から選べ。

の選択肢

- ㊦ 奇数個検出する仕組み
- ① 偶数個検出する仕組み
- ② 場所を特定する仕組み
- ③ 場所を特定し、修正する仕組み

問 3 Aさんは実際に、偶数パリティビットを付与するプログラムを作成した。以下の空欄に当てはまるプログラムを選択肢から選べ。

関数 count1(data)は

kosu ← 0

// データの中にある 1 の個数を数える

i を 0 から 要素数(data) - 1 まで 1 ずつ増やしながら,

もし ならば

kosu ← kosu + 1

を実行する

を繰り返す

kosu を返す

を実行する

関数 addParity(data)は

// 引数で指定した data に、偶数パリティビットを付与する

pari ← 0

kosu ← count1(data)

もし kosu % 2 != 0 ならば

を実行し, そうでなければ

を実行する

data を返す

を実行する

// 元データ

data ← {0,0,1,0,1}

”元データ→” + data を表示する

// パリティビットを付与する

data ← addParity(data)

”パリティビットを付与した結果→” + data を表示する

表示結果

元データ→{0,0,1,0,1}

パリティビットを付与した結果→

使う関数

要素数(配列名)...

引数に 配列 をとる。その配列の中にある要素数を戻り値とする関数。

count1(data)...

引数として渡された配列(data)の中に含まれる 1 の個数を数える関数。配列の各要素をチェックし、戻り値として data の中にある 1 の個数を戻り値とする。

例: data ← {1,0,1,1,1} のとき、count1(data) の戻り値は 4 となる。

末尾に追加(Data, value)

引数である data (配列) の末尾に 2 つ目の引数 value を追加する。戻り値はなし。

例: Data← {1,0,1,1,1}, value←1 のとき、配列 Data の値は {1,0,1,1,1,1} となる。

addParity(data)

引数で渡された配列 data の末尾に偶数パリティビットを追加する関数。偶数パリティビットを付与した新しい配列を戻り値とする。

工 の選択肢

- ① kosu % 2 = 0
- ① kosu % 2 != 0
- ② data[i] = 0
- ③ data[i] = 1

オ の選択肢

- ① 末尾に追加(data, 0)
- ① 末尾に追加(data, 1)
- ② 末尾に追加(0,data)
- ③ 末尾に追加(1,data)
- ④ data ← 末尾に追加(0)
- ⑤ data ← 末尾に追加(1)

カ の選択肢

- ① {0,0,1,0,1,0}
- ① {0,0,1,0,1,1}
- ② {0,0,0,1,0,1}
- ③ {1,0,0,1,0,1}