2024年度
ＡＯ入試１次選考問題

情報理工学部

情報科目

注 意 事 項

1. 試験開始の合図があるまで、問題用紙を開いてはいけません。
2. 解答はすべて、所定の解答用紙に記入してください。
3. 解答用紙に受験番号と氏名（フリガナ）を記入してください。
4. 解答時間は６０分です。問題は２２ページあります。
5. 問題Ⅰから Ⅳ までは必答、問題 Ⅴ は(A)，(B)のいずれかを選択し、解答してください。
6. 問題用紙・解答用紙および計算用紙はすべて回収します。一切持ち帰ってはいけません。

〔 I 〕 以下の文章を読んで、設問に答えなさい。

次の文章は、あるユーザがスマートフォンで写真を撮影して SNS に投稿する過程で行われたデータの処理や操作の説明について段階的に述べたものである。

空欄 [ア] ～ [オ] に入れるのに最も適切な語をそれぞれの解答群のうちから一つずつ選べ。

1. あるユーザがスマートフォンを操作してカメラアプリを起動し、シャッターボタンを押して撮影操作をした。
2. このときカメラアプリはカメラから縦 2048、横 1536 [ア] のカラー画像データを得た。
3. この画像を SSD（ストレージ）へ保存する際は、[イ] に、JPEG フォーマットに変換する。
4. このときの JPEG フォーマットへの変換では一般に [ウ] が行われる。
5. 次に SNS アプリを起動して、SNS サービスに画像データを [エ] するが、このとき、画像データは小さな [オ] と呼ばれる形に分割されてインターネットに接続されたサーバに送信される。

[ア] の選択肢：

【 1. ビット　2. バイト　3. 画素　4. フレーム 】

[イ] の選択肢：

【 1. 秘密保持のため　2. データ量を削減するため
3. 画質を劣化させないため　4. 画像処理を可能にするため 】

[ウ] の選択肢：

【 1. 可逆圧縮　2. 非可逆圧縮
3. 公開鍵を用いた暗号化　4. 秘密鍵を用いた暗号化 】

[エ] の選択肢：

【 1. クラウド化　2. 符号化　3. サンプリング
4. アップロード　5. ダウンロード 】

[オ] の選択肢：

【 1. プロトコル　2. ルータ　3. IP アドレス　4. パケット 】

〔Ⅱ〕 以下の文章を読んで、設問（A）～設問（D）に答えなさい。

3つの数字X, Y, Zに対して、チェックディジット（検査数字）となる数字Cを計算し、それらで構成される数字列 [X, Y, Z, C]を考える。X, Y, Zはそれぞれ0～5の整数で、今回、Cは以下の計算式で求められる。ここで、a mod b は aをbで割った余りを表すものとする。

$C=(X\times5+Y\times2+Z\times1) \bmod 7$

例えば、X=1, Y=2, Z=3のとき、Cは $(1\times5+2\times2+3\times1) \bmod 7 = 12 \bmod 7 = 5$ となる。

設問（A）

Cのチェックディジットの役割や性質として適切なものを、以下から全て選べ。

(a) Cは数字列の最後に置く必要がある。
(b) Cがあることにより、X, Y, Zの数字の入力間違いを検出できる場合がある。
(c) Cがあることにより、X, Y, Zの数字の入力間違いを誤り訂正できる。
(d) Cの値が同じになるX, Y, Zの値の組合せは複数存在する。
(e) X, Y, Zの値を単純に加算したチェックサムを使う方式と違い、桁の入れ替わりを検出できる場合がある。

設問（B）

数字列 [2, 5, 3, C] のとき，Cに当てはまる数は何か。

設問（C）

数字列 [3, 4, Z, 5] がこの条件を満たすとき，Zに当てはまる数は何か。

設問（D）

数字列 [1, Y, Z, 5] がこの条件を満たすとき，Y, Zの組合せは何通りあるか。

〔Ⅲ〕以下の文章を読んで、設問（A）～ 設問（D）に答えなさい。ただし、この問〔Ⅲ〕では、問題冊子の最後に示す記法を使ってプログラムを記述する。

次のような操作が行えるリストを考える。

- push 操作は、リストの最後に文字を追加できる。

  `[ a b c ] -- push d → [ a b c d ]`
- pop 操作はリストの最後の位置にある文字をリストから削除した上で取り出せる。取り出した文字はそのまま出力される。

  `[ a b c d ] -- pop → [ a b c ]` （d が出力される）
- swap 操作を行うと、リストの最後の文字と最後から２つ目の文字が入れ替わる。

  `[ a b c ] -- swap → [ a c b ]`
- dup 操作はリストの最後の文字を重複させる。

  `[ a b c ] -- dup → [ a b c c ]`

リストには文字が１つも含まれていなくてもよく、そのようなリストを空リストと呼ぶ。また、含まれる文字の個数に制限はないものとする。

このとき、空リストに対して順に push r, push a, push t, push s と 4 回 push 操作をすると `[ r a t s ]` というリストが得られる。その後、4 回 pop すると、`star` が出力される。

ここまでを踏まえて、以下の問に答えよ。

設問（A）

`[ o n f i ]` というリストが与えられたとき、以下の操作を行って、`info` という出力を得た。(1)～(4)で行う適切な操作を答えよ。

`[ pop ], [ (1) ] , [ (2) ] , [ (3) ] , [ (4) ]`

設問（B）

空リストに対して、以下の操作を順に行なって `[ e y e ]` というリストを得た。(1), (2) で行う適切な操作を答えよ。ただし、(1), (2) は push 操作ではないものとする。

`[ push e ] , [ (1) ] , [ push y ] , [ (2) ]`

設問（C）

[n a p j] というリストが与えられたとき、japan と出力するための最短の操作手順を答えよ。ただし push 操作は行わないものとする。

設問（D）

使用できる操作を pop と swap に限定したとき、与えられたリスト str から目的の文字列 target が出力できるかを判定する関数（プログラム）を作成したい。ただし、str と target の長さは同じであるものとする。

図 1 に示すフローチャートを参考にして、図 2 に示した関数 check()を完成させよ。関数 check()の二つの引数 str、target にはともに配列変数として文字列の情報が渡される。戻り値には 1（出力可能）あるいは 0（出力不可能）が判定結果として設定される。

なお、関数 check()で使われている len 関数は引数で与えられた配列の長さを返すものとし、swap 関数は第 1 引数で与えられた配列の中の第 2 引数、第 3 引数のインデックスの要素を入れ替える関数である。

例えば関数 check()の引数として
str に ["D", "B", "C", "A"]
target に ["A", "B", "C", "D"]
を与えて呼び出せば、
その戻り値は 1 となる。
また操作の確認のために
POP
SWAP
POP
POP
POP
と画面に出力する。

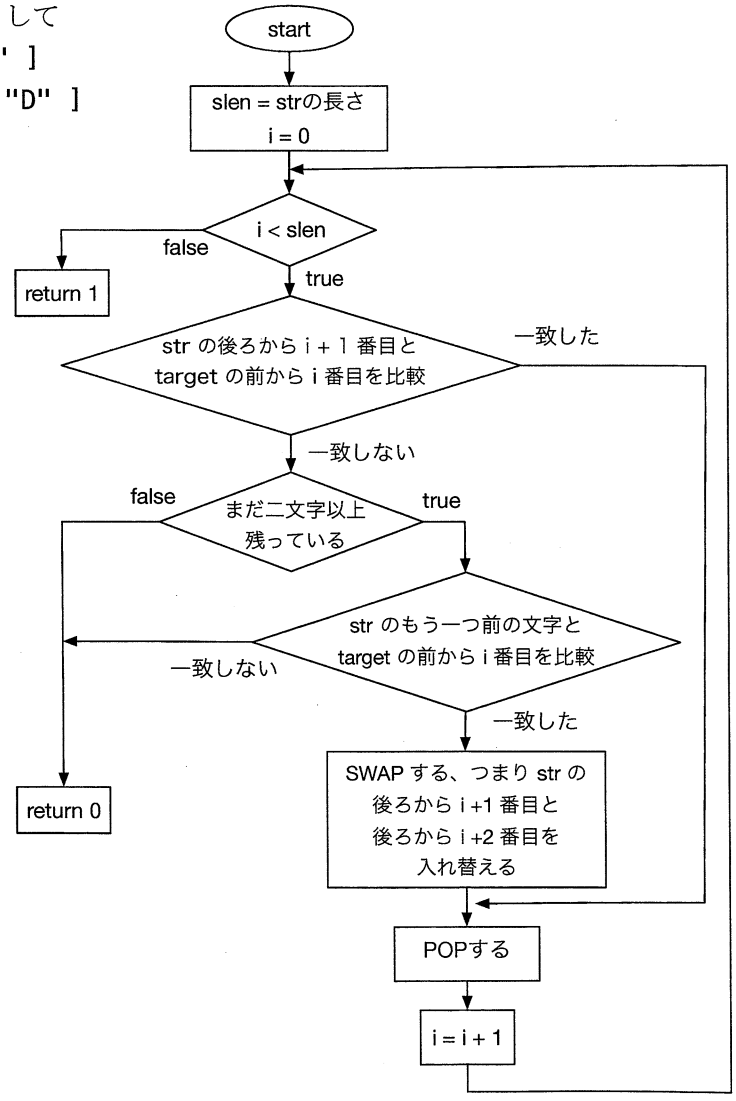

図 1. フローチャート

```
func check(str, target)
    var i, slen, ws1, wt1, ws2
    slen = len(str)

    for (i = 0; i < slen; i = i + 1)
        ws1 = [          ]
        wt1 = [          ]
        if ws1 == wt1
            [          ]
        else if (slen - i - 2) >= 0
           ws2 = [          ]
           if ws2 == wt1
               swap(str, slen - i - 2, slen - i - 1)
               print("SWAP")
               print("POP")
           else
               return 0
           end
        else
           return 0
        end
    end
    return 1
end
```

図 2. プログラム

(解答用紙にも同じ図がある)

〔Ⅳ〕 以下の文章を読んで、設問（A）～ 設問（C）に答えなさい。

学生Xが「情報通信機器の保有率と、インターネットへ接続するときに利用する情報通信機器に関する近年の変化」について調べることとした。そこで学生Xは、総務省の実施した「通信利用動向調査」から2014年から2022年（2016年は除く）までの3種類（A、B、C）の情報通信機器に関する統計情報を収集した。この統計情報は、20歳以上の世帯主を対象としたものである。また、年齢区分を20歳代（20 ～ 29歳）、30歳代（30 ～ 39歳）、40歳代（40 ～ 49歳）、50歳代（50 ～ 59歳）、60歳代（60 ～ 69歳）、70歳代（70 ～ 79歳）、80歳以上に設定し、各年齢区分での情報通信機器A、B、Cの保有率を求めている。ここで保有率とは、各年齢区分の回答者全員に対する百分率のことである。

まず学生Xは、情報通信機器Aに関して、世帯主の年齢区分ごとの保有率に関して統計情報を表として整理することにした。表 1は、情報通信機器Aの保有率を世帯主の年齢区分ごとに整理した表である。

表 1 情報通信機器Aに関する年齢区分ごとの保有状況 ［単位 %］

	2022年	2021年	2020年	2019年	2018年	2017年	2015年	2014年
20歳代	97.8	98.2	97.7	99.0	95.9	97.1	98.4	94.5
30歳代	98.6	99.4	99.3	97.9	97.2	97.4	94.2	92.4
40歳代	98.1	98.5	97.4	97.2	95.0	94.1	88.8	83.8
50歳代	97.3	96.8	96.0	94.5	89.7	88.3	83.9	75.0
60歳代	93.0	91.0	87.6	83.8	78.3	69.5	64.2	47.7
70歳代	81.6	78.3	71.1	60.4	55.4	44.9	36.7	28.0
80歳以上	62.4	56.9	50.0	44.6	35.5	31.5	31.9	20.8

設問（A）

情報通信機器Aの保有率に関して、表 1を見て読み取れることを、次の解答群からすべて選べ。ただし、表に記載されていない年についての保有率は考慮しないものとする。

設問（A）の解答群

（ 1 ） 20歳代の保有率は、2014年以後2022年（2016年は除く）まで90%以上となっている。

（ 2 ） 2014年から2022年（2016年は除く）までの、全ての年齢区分の保有率は、年々増加している。

（ 3 ） 70歳代に関して、2022年の保有率は2014年の2倍以上になっている。

（ 4 ） 50歳代の保有率が、2014年の30歳代の保有率以上に、初めてなったのは2019年である。

（ 5 ） 半数の年齢区分で90%以上の保有率となったのは、2017年からである。

次に学生 X は、各年齢区分における情報通信機器 A と B の保有率に関して、統計情報を整理することにした。図 1 は、2014 年から 2022 年（2016 年は除く）までの情報通信機器 A と B の保有率を、年齢区分ごとにそれぞれ 1 つにまとめて箱ひげ図として表した図である。したがって、各箱ひげ図において、保有率の最も高い年が最大値をとり、最も低い年が最小値をとる。この箱ひげ図により、情報通信機器 A、B の保有率が最も少なかった年から、最も保有率の多かった年への変化（保有率に関する最小値と最大値の幅）を視覚的に知ることができる。また平均値、中央値はそれぞれ、8 年間の各情報通信機器の保有率の平均値と中央値である。

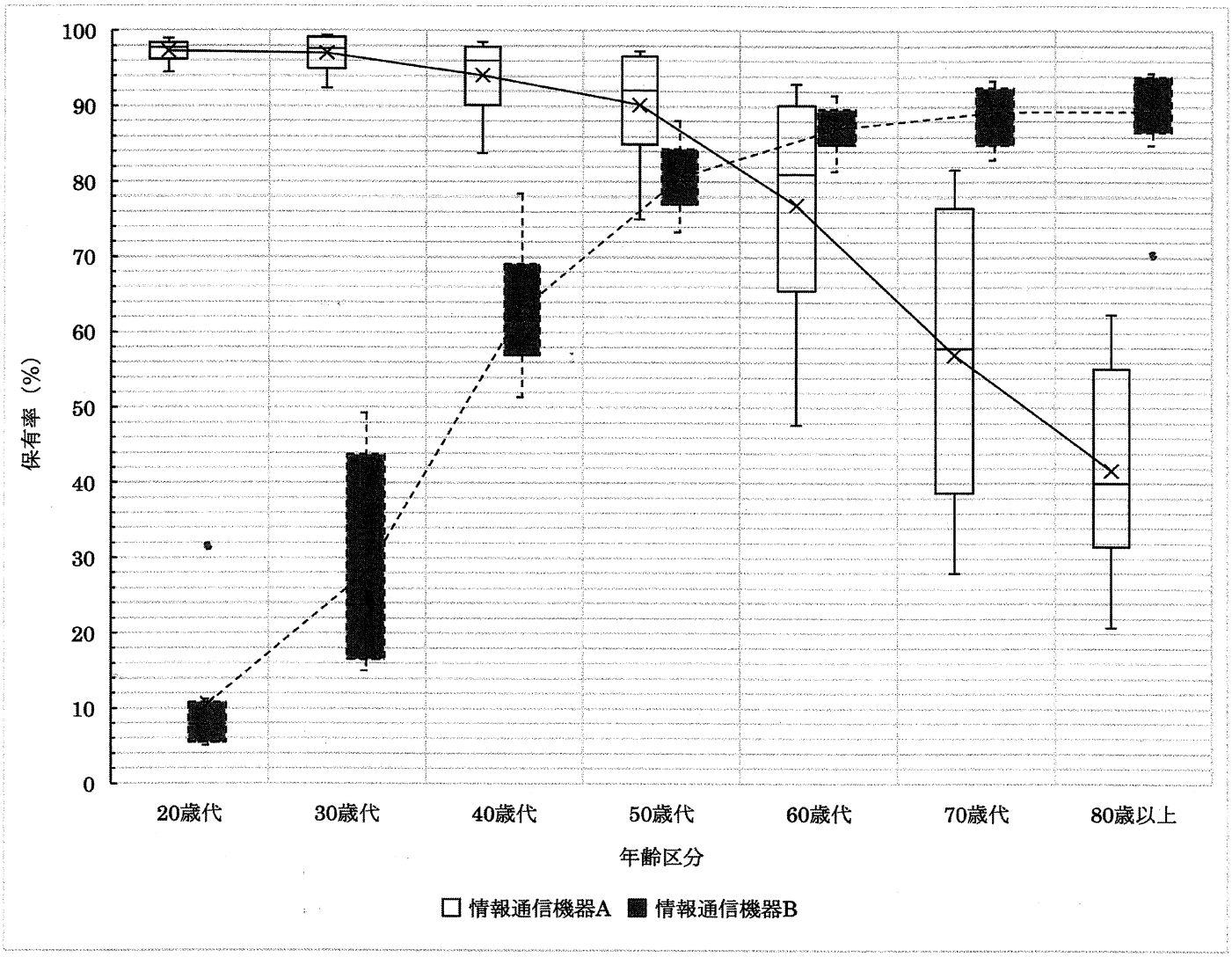

図 1 2014 年から 2022 年（2016 年は除く）までの 8 年間の各年齢区分における情報通信機器 A と B の保有率

設問（B）

情報通信機器 A と B の保有率に関する図 1 を見て読み取れることを、次の解答群からすべて選べ。ただし、グラフに記載されていない年についての保有率は考慮しないものとする。

設問（B）の解答群

（1） 8 年間の中で保有率が最も大きく変化したのは、70 歳代の情報通信機器 A の保有率である。

（2） 情報通信機器 A の 60 歳代の保有率の最大値は、情報通信機器 A の 40 歳代の 8 年間の中央値より低い。

（3） 情報通信機器 A と B の 20 歳代の 8 年間における保有率の変化は、どちらも 20%以下である。

（4） 情報通信機器 A と B の両方の 8 年間の平均保有率が 80%以上であるのは、50 歳代だけである。

（5） 年齢区分が低くなるほど情報通信機器 A の 8 年間の保有率の平均値は高くなり、年齢区分が高くなるほど情報通信機器 B の 8 年間の保有率の平均値は高くなる。

最後に学生Xは、インターネットを利用するときにどの情報通信機器を用いているのかを調査した統計情報を分析した。情報通信機器A、B、Cの中で、情報通信機器Bは、インターネットに接続する機能を保有していないため、インターネットを利用する情報通信機器の統計調査から除外した。この分析では、インターネットを利用するときに情報通信機器Aを用いると回答した世帯主の率（各年齢区分の回答者全員に対する百分率）を、「情報通信機器Aを用いたインターネット利用率」と定義した。「情報通信機器Cを用いたインターネット利用率」についても、同様に定義した。2014年から2022年（ただし、2016年は除く）に関して、情報通信機器Aを用いたインターネット利用率と、情報通信機器Cを用いたインターネット利用率の相関係数rは -0.64であり、その関係性を散布図として図 2に表した。

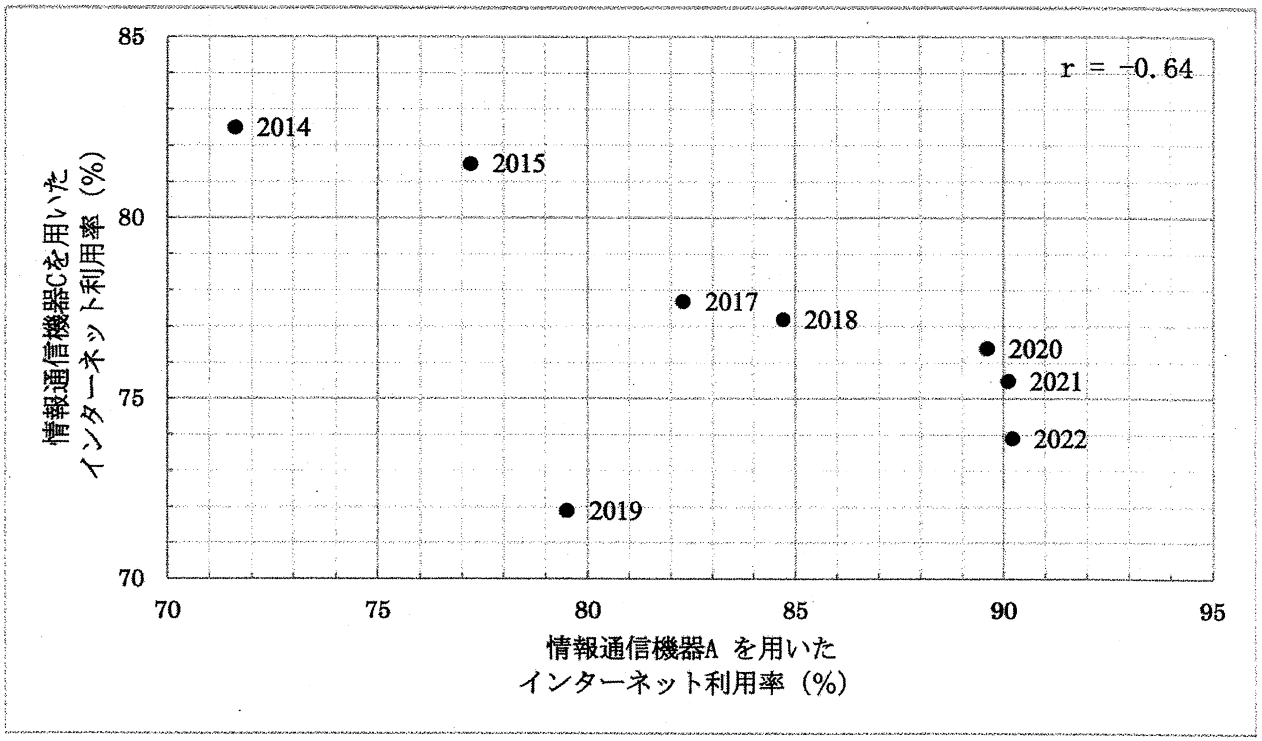

図 2　情報通信機器Aを用いたインターネット利用率と、情報通信機器Cを用いたインターネット利用率の関係性を表した散布図

設問（C）

情報通信機器Aを用いたインターネット利用率と情報通信機器Cを用いたインターネット利用率の関係性ついて、図2から読み取れることを次の解答群からすべて選べ。ただし、グラフに記載されていない年についての保有率は考慮しないものとする。

設問（C）の解答群

（1）　情報通信機器Aを用いたインターネット利用率が増えるにつれて、情報通信機器Cを用いたインターネット利用率も増えている。

（2）　2017年では、情報通信機器Aを用いたインターネット利用率は約82%であった。一方、情報通信機器Cを用いたインターネット利用率は約78%であった。

（3）　情報通信機器Aを用いたインターネット利用率と情報通信機器Cを用いたインターネット利用率の相関係数rの値から、これら2つの情報通信機器を用いたインターネット利用率の間には相関関係はない。

（4）　情報通信機器Aを用いたインターネット利用率と情報通信機器Cを用いたインターネット利用率の相関係数rの値から、これら2つの情報通信機器を用いたインターネット利用率の間には負の相関関係がある。

（5）　情報通信機器Aを用いたインターネット利用率は、年々増加している。

問題Vは(A),(B)のうち一つを選択して解答してください。問題V(B)はp.15にあります。

〔V(A)〕 以下の文章を読んで、設問（A）～ 設問（D）に答えなさい。ただし、この問〔V(A)〕では、問題冊子の最後に示す記法を使ってプログラムを記述する。

あるコンピュータにセンサが接続されている。センサからは 0 ～ 100 （0以上100以下）の整数値を読み取ることができ、コンピュータではこれを1バイト（8ビット）のデータとして扱う。このセンサから1秒ごとにデータを1回読み取り、1時間かけて得たすべての値をひとつのファイルに保存した。

このファイルからデータを読み込んで表示や計算を行うプログラムを作成する。

プログラム1は、ファイルに保存されたデータを1バイトずつ読み込んで、コンピュータの画面に整数値として表示するプログラムである。

ここで関数 eofData()は、ファイルから読み込めるデータがあるかどうかをチェックするものである。もう読めるデータがない（ファイルの終わりに到達した）場合に値 1 を、まだ読み込むことができるデータがある場合は 0 を返す。

関数 getData()はファイルからデータを読み込む。呼び出されるたびにファイルから1バイトずつデータを読み込んで 0 ～ 255 の整数値として返す。

```
var data # 変数の宣言
while eofData() == 0    # 読み込めるデータがある限り繰り返す
    data = getData()    # ファイルから1バイト読んで変数に代入
    print(data)         # 整数値を表示
end
```

プログラム1

設問（A）

プログラム2は、上で述べたファイルから10個のデータを読み込むたびに、それらの平均値を計算して出力するプログラムである。

意図した通りに正しく動作するように、空欄を埋めなさい。

```
var sum, count, data
sum = 0
count = 0
while eofData() == 0
    data  = getData()
    [          ]
    [          ]
    if count == 10
        print(sum / 10)
        sum = 0
        count = 0
    end
end
```

プログラム2

より長い時間、センサからデータを取得することにしたが、データを保存するファイルのサイズが巨大になってしまう。センサからは繰り返し 0 が返される傾向が高いことが分かっているため、これを利用してデータを 1 日の終わりに圧縮して保存することにした。

センサから得られるのは 1 バイトの整数データだが、100 よりも大きな値は出現しない。そこで、101 から 255 の範囲の整数値をデータの圧縮表現のために利用する。

圧縮方法を以下に示す。

データの圧縮方法

センサから読み取ったデータに値 0 が N 個連続して出現した場合、(257 − N) を計算し、その値を N 個の 0 の代わりに 1 バイトのデータとしてファイルに保存する。ただし、N は 2 以上、156 以下でなければならない。それ以外のデータと、連続しない 1 個だけの 0 はそのままファイルに保存する。N が 156 より大きい場合は、101 をファイルに保存して N の値を 156 減らすという手順を繰り返す。

たとえば、センサから取得した 0 5 0 0 0 0 0 1 というデータ列に上記の圧縮方法を適用することを考える。先頭の 0 はそのまま表すことになるが、途中に 5 個の 0 が連続しているため、 0 5 252 1 という 4 バイトで表現できる。

逆に、圧縮されたデータが 66 0 7 7 7 254 29 というバイトの列としてファイルに保存されていたとする。復元すると 66 0 7 7 7 0 0 0 29 というデータ列がセンサから得られていたことが分かる。

設問 (B)

まず上のアルゴリズムで圧縮されたデータを復元する方法について考える。

プログラム 3 は圧縮されたデータをファイルから読み込み、復元してセンサから得られた元のデータ列を表示するプログラムである。

意図した通りに正しく動作するように、空欄をすべて埋めなさい。

```
var data, n, i
while eofData() == 0
    # 圧縮されたファイルからデータを読む
    data = getData()
    if data <= 100
        print(data)
    else
        n = [          ]
        for (i = 0; i < n; [          ])
            [          ]
        end
    end
end
```

プログラム 3

設問（C）

センサからのデータは圧縮せずにファイルに保存し、1日の終わりに得られたデータを圧縮形式に直して、別のファイルに保存する。

プログラム4の関数 saveCompressedData() は、センサのデータを保存しているファイルからデータを読み取り、圧縮形式に直して別のファイルに書き込む。この関数では、処理の見通しを良くするために関数 putZeros() を用いている。これらの関数は戻り値を返さない。なお、関数 putData() は新しく作成する保存用ファイルに1バイトのデータを書き出す。

関数 saveCompressedData() が意図通りに動作するように、空欄をすべて埋めなさい。

データの圧縮方法（再掲）

センサから読み取ったデータに値 0 がN個連続して出現した場合、（257 − N）を計算し、その値をN個の 0 の代わりに1バイトのデータとしてファイルに保存する。ただし、Nは2以上、156 以下でなければならない。それ以外のデータと1個だけの 0 はそのままファイルに保存する。Nが 156 より大きい場合は、101 をファイルに保存してNの値を 156 減らすという手順を繰り返す。

```
func putZeros(n)
    # n個の連続する 0 を処理する
    var count, data
    count = n
    while count > 156
        putData(101)
        count = count - 156
    end
    if count > 0
        if count == 1
            data = [          ]
        else
            data = [          ]
        end
        putData(data)
    end
end
```

```
func saveCompressedData()
    var length, data
    length = 0
    while eofData() == 0
        # 圧縮されていないデータを読む
        data = getData()
        if data == 0
            [                  ]
        else
            if length > 0
                putZeros(length)
                [                  ]
            end
            putData(data)
        end
    end
    if length > 0
        putZeros(length)
    end
end
```

プログラム4

（設問（D）は次ページに続く）

既存の分析プログラムはすべて圧縮されていないデータ列をファイルから読み込むように設計されている（図 1)。これを図 2 に示すように単に圧縮データに対応した関数を呼ぶように修正するだけで、同じように機能するようにしたい。

具体的には、圧縮されたファイルからデータを復元しながら元のデータを１バイトずつ読み込む関数 getExData() と、それ以上読み込めるデータがないことを判定する関数 endExData() を新たに作成した。プログラムの中の getData() を getExData() に、eofData() を endExData() に置き換えることで、圧縮形式のファイルを入力としてデータ分析が行えるようになる。

```
func Analyze()
  while eofData() == 0
    v = getData()
    if v > 10
        data = getData()
        ...
    v = getData()
    while v > 50
        if eofData() == 1
        ...
```

図１　元の分析プログラム例

```
func Analyze()
  while endExData() == 0
    v = getExData()
    if v > 10
        data = getExData()
        ...
    v = getExData()
    while v > 50
        if endExData() == 1
        ...
```

図２　修正した分析プログラム例

たとえば以下のプログラム５は、プログラム１の getData() を getExData() に、eofData() を endExData() に置き換えたものである。このプログラムは圧縮されたデータを入力として、すべてのデータを表示できる。

```
var data # 変数の宣言
while endExData() == 0  # 読み込めるデータがある限り繰り返す
   data = getExData()   # データを１バイト読んで変数に代入
   print(data)          # 整数値を表示
end
```

プログラム５

設問（D）

プログラム６は関数 getExData() と関数 endExData() の定義である。変数 count はグローバル変数で、プログラムの実行開始時の初期値は 0 である。関数 endExData() は、圧縮形式のファイルからデータを復元して読み込める間は 0、読み込めなくなった場合に 1 を返す。

これら２つの関数が意図通りに正しく動作するように、空欄をすべて埋めなさい。

データの圧縮方法（再掲）

センサから読み取ったデータに値 0 が N 個連続して出現した場合、（257 − N）を計算し、その値を N 個の 0 の代わりに１バイトのデータとしてファイルに保存する。ただし、N は２以上、156 以下でなければならない。それ以外のデータと１個だけの 0 はそのままファイルに保存する。N が 156 より大きい場合は、101 をファイルに保存して N の値を 156 減らすという手順を繰り返す。

```
# グローバル変数
var count = 0

func endExData()
    var eod
    if count > 0
        [          ]
    else
        eod = eofData()
        # 圧縮形式ファイルの末尾？
    end
    return eod
end
```

```
func getExData()
    var data
    if count > 0
        count = count - 1
        [                    ]
    else
        data = getData()
        # 圧縮形式のファイルから読み込む
        if data > 100
            [                    ]
            data = 0
        end
    end
    return data
end
```

プログラム６

問題Vは(A),(B)のうち一つを選択して解答してください。問題V(A)はp.9にあります。

〔V(B)〕 以下の文章を読んで、設問（A）～ 設問（C）に答えなさい。

あるロボット工場では、製造したロボットの出荷前の評価を2段階に分けて行っている。1つは、パーツごとの基本性能を数値的に評価する性能試験で、2つ目は、ロボットが全体として不良品でないかを最終的に判定する動作チェックである。工場で製造したロボットには全て、これら 2 種類の評価をそれぞれ実施し、そのデータを製造月ごとにまとめている。

下の表 1 は、この工場で製造したロボットの性能試験の結果を表している。この表は、工場で製造した全ロボットのうち、性能試験で基準値未満であったロボットの個数と、基準値以上であった個数を、4月から7月までの月ごとにまとめたものである。

一方、表 2 は、工場で製造したロボットのうち最終動作チェックにおいて不良品と判定されたロボットだけに絞って、表 1 と同様に、性能試験で基準値未満であった個数と基準値以上であった個数を月ごとにまとめたものである。例えば、4月に製造したロボットで不良品と判定されたもののうち性能試験で基準値未満であったロボットは24個、基準値以上であったロボットは6個である。これらを合わせて、4月の不良品の総数は30個である。

以下の設問では、この性能試験が不良品を見分けることにどの程度役立つかを考える。

表 1. ロボットの性能試験における基準値未満の個数と基準値以上の個数（個）

		月				
		4月	5月	6月	7月	計
性能試験	基準値未満	96	142	160	242	640
	基準値以上	654	858	1090	1758	4360
	計	750	1000	1250	2000	5000

表 2. 不良品のロボットが性能試験で基準値未満であった個数と基準値以上であった個数（個）

		月				
		4月	5月	6月	7月	計
性能試験	基準値未満	24	48	40	48	160
	基準値以上	6	12	10	12	40
	計	30	60	50	60	200

設問（A）

（i）

まず初めに、4月から7月までの月を合算して、性能試験の結果（基準値未満、基準値以上）と最終動作チェックの結果（不良品、不良品でない）とを掛け合わせたロボットの個数の表を作成した。これを示す下の表 3 において、例えば、性能試験が基準値未満で最終動作チェックで不良品と判定された個数は、表中の①の箇所に記載され、その数は 160 個である。同様にして、解答用紙の表内の②から⑨までの箇所全てに、適切な数値（該当する個数）を記入せよ。

表3. 4月から7月までを合算した、性能試験と最終動作チェックの結果を掛け合わせた表(個)

		性能試験		
		基準値未満	基準値以上	計
最終動作チェック	不良品	① 160	②	③
	不良品でない	④	⑤	⑥
	計	⑦	⑧	⑨

（ii）

（i）で作成した表 3 を用いてまず、4 月から 7 月までの月を合算して、不良品のロボットのうち性能試験で基準値未満であった割合を調べることを考える。この割合を百分率（%）で求め、途中の計算式とともに解答用紙の式の＝の右に続けて記載せよ。最終的な計算結果の数値には単位も記載すること。

設問（B）

(i)

設問（A）で作成した表 3 を用いて次に、ロボットが性能試験で基準値未満であった場合に不良品である割合を見積もることを考える。4 月から 7 月までの月を合算して、この割合を百分率（%）で求め、途中の計算式とともに解答用紙の式の＝の右に続けて記載せよ。最終的な計算結果の数値には単位も記載すること。

(ii)

性能試験が基準値未満であったか基準値以上であったかの結果をもとにロボットが不良品であるか否かを高い割合で当てることができるようにするためには、（ｉ）で解答した割合を高めることに加えて、性能試験に関してまた別の改善が必要である。この改善内容を説明した次の文章の［ ア ］、［ イ ］、［ ウ ］内に、下に示した単語群の中から適切なものをそれぞれ 1 つずつ選んで a ～ f の記号で解答せよ。

改善内容：「［ ア ］に対して性能試験が［ イ ］の結果を示す割合を［ ウ ］できるような改善」

単語群：

a. 不良品のロボット	b. 不良品でないロボット	c. 基準値以上
d. 基準値未満	e. 50%に	f. 低く

設問（C）

（i）

最後に、4月から7月にかけてこの工場で製造したロボットに占める不良品の割合がどのように変化したかを考える。そこで、冒頭に示した表1と表2のデータから、各月に工場で製造したロボットの中の不良品の割合を百分率（%）でそれぞれ求め、その値の4月から7月までの変化を表すグラフを作成した。このグラフを、解答用紙の図内に折れ線グラフで描き入れよ。その際、折れ線グラフは、4月から7月の各月の不良品の割合を表す各点をグラフ描画領域内に○のマークで記載し、それらを実線でつないだグラフとして記入すること。

（ii）

（i）で作成したグラフと、冒頭に示した表1および表2から分かることとして適切なものを下の選択肢 a ～ e の中から全て選んで、解答用紙の枠内に a ～ e の記号で記載せよ。

選択肢：

a. この工場で製造したロボットの不良品の割合は、4月から7月まで毎月増え続けている

b. この工場で製造したロボットの不良品の割合は、4月から5月にかけて増加したが、それ以降6月から7月にかけて減少した

c. 不良品のロボットのうち性能試験で基準値未満であった割合は、4月から7月まで変化せず一定である

d. 不良品のロボットの総数が6月から7月にかけて増えていることは、この工場で製造したロボットに占める不良品の割合が6月から7月にかけて増えたことを示している

e. 不良品のロボットの総数が6月から7月にかけて増えていることは、この工場で製造したロボットに占める不良品の割合が増えたためではなく、製造したロボットの個数が6月から7月にかけて増えたことと関係している

プログラムの記法の説明

本試験の問、および解答においてプログラムを記述するために用いる記法について説明する。

文

変数 = 式	変数に式の値を代入する。以下の説明ではこの文を「代入文」と呼ぶ。
for（代入文１; 条件式; 代入文２） ... end	まず、代入文１を実行し、条件式を評価する。条件式が偽であれば何もしない。条件式が真のとき、end までの命令を実行し、次に代入文２を実行する。その後、条件式が真である間、end までの命令と代入文２を実行する。
while 条件式 ... end	条件式が真である（条件が成立する）間、end までの命令（文の並び）を実行する。
if 条件式 ... end	条件式が真（条件が成立）のとき、end までの命令（文の並び）を実行し、偽（条件が不成立）のときは何もしない。

if 条件式
  ...(命令 1)...
else
  ...(命令 2)...
end
	条件式が真のとき、else までの部分（命令 1）を実行し、偽のときは else から end までの部分（命令 2）を実行する。

if 条件式１
  ...(命令 1)...
else if 条件式２
  ...(命令 2)...
else if 条件式３
  ...(命令 3)
else
  ...(命令 n)...
end
	複数の条件式を順番に調べ、最初に真になった条件式に対応する命令だけを実行したい場合、else と if を組み合わせて左のように記述する（左は条件式が３つある場合）。どの条件式も偽の場合、最後に else があればその部分の命令（命令 n）を実行するが、なければ何もしない。
	曖昧さを避けるために「else if 条件式」の部分は１行に記述しなければならない。また、else は最後の部分として記述しなければならない。

break	for 文、または while 文（これらをループ文と呼ぶ）の内部でのみ利用できる。実行すると繰り返しの処理を打ち切り、ループ文の次の文の実行に移る。ループ文の中でループ文が使われている（ネストしている）時は、一番内側のループ文の処理だけが打ち切られる。
return 式 return	関数の内部でのみ利用できる。実行すると関数の実行を打ち切り、式の値を関数の評価値として呼び出し側に渡す。 戻り値の必要ない関数では、return の後の式を記述しない。
print(式)	式の値を表示する。整数値の場合、10 進数で表示する。

関数

```
func F( x )
  var a, b
  ...
  return c
end
```

名前*F*を持つ関数（サブルーチン）を定義する。*x*は引数で、実行の際に呼び出し側から値が渡される。引数を指定しない場合は ( )だけを記述し、複数の引数を指定する場合は「,」で区切る。
var の後に変数名を記述して、関数の内部でのみ利用可能な変数（ローカル変数）を宣言できる。
戻り値のない関数を定義することもでき、その場合、return 文の式を省略できる。さらに、関数の末尾の end の直前が戻り値のない return 文の場合、return 文自体を省略することができる。

F(式)

名前*F*を持つ関数（サブルーチン）を実行する。式の値は引数として関数に渡される。式は、関数の定義で示された引数の数だけ記述する。引数が必要ない場合は ( ) だけを記述し、複数の引数がある場合は、foo(3, a+1) のように「,」で区切る。
戻り値のある関数は a = foo(3, a+1) - 1 のように式の中で呼び出すことができる。関数の値は、return で返される式の値である。

定数

整数

プログラムでは10進数の整数値を記述できる（例: 120, -1, +5)。

文字列

対になった " " で囲って文字列を表現できる（例: "ABC", "Level 5"）。
文字を1つも含まない文字列（空文字列）は "" のように表す。

変数と配列

var *a*, *b*[整数]
（*a*, *b* は名前の例）

関数の内部で宣言した場合、その関数でのみ利用可能な変数（ローカル変数）となり、関数の外部で宣言するとどこからでも利用可能な変数（グローバル変数）となる。
名前の直後に [整数] を続けて記述すると、整数値が示す要素数を格納できる配列を宣言できる。ただし、値は正整数でなければならない。
変数名、配列名を「,」で区切って複数個が宣言できるが、その場合の変数、配列の初期値は不明な値になる。

var *a* = 定数

変数の宣言と同時に初期値を指定できる。グローバル変数の場合、プログラムの実行開始に初期値が与えられる。ローカル変数には、関数が実行される際に値が代入される。

var *b*[ ] = { 定数, ... }

変数の宣言と同時に初期値を指定できる。グローバル変数の場合、プログラムの実行開始に初期値が与えられる。ローカル変数には、関数が実行される際に値が代入される。
なお、この形式では配列の要素数を省略できる。

配列名[式]

宣言された配列の要素を代入先として指定したり、式の中で参照したりできる。配列 a の要素数が N (N は正の整数) のとき、配列 a は 0 番目から (N−1) 番目までの要素を持ち、i 番目の要素は a[i] と表現する。

算術演算子

整数値に対して算術演算を行うことができる。

+	加算（足し算）を行う。
−	減算（引き算）を行う。
*	乗算（掛け算）を行う。
/	除算（割り算）を行う。ただし、結果は実数で表される。
%	剰余算を行う。剰余算は割り算の余りを求める。例えば 7%3 は 1 となる。

また、− は式の先頭(左)に付けて −1 を乗じるのと同じ演算を表すことができる(例: −x)。

複数の算術演算子が混在した式では * と / の計算が + や − よりも先に行われる。
算術演算子と比較演算子が混在した式では、算術演算子が先に計算される。
式の中で( ) を使い、計算の順序を示すことができる。

比較演算子

整数値同士、文字列同士を比較できる。結果は真か偽のいずれかである。

A == B	AとBの値が等しい。
A != B	AとBの値が等しくない。

２つの整数値の大小を比較できる。結果は真か偽のいずれかである。

A <= B	Aの値がBの値以下である。
A < B	Aの値がBの値より小さい。
A >= B	Aの値がBの値以上である。
A > B	Aの値がBの値より大きい。

条件式

if 文やループ文の実行は条件式の評価結果（真または偽）で制御される。比較演算子を用いた値の比較は条件式である。また、条件式同士を AND、OR、または ! で組み合わせた式も条件式となる。

A AND B	２つの条件式AとBが両方とも真の場合のみ、結果が真になる。
A OR B	AとBのどちらか一方、または両方が真の場合、結果が真になる。
!A	条件式Aの否定を表す。つまり、Aが真の場合、!A は偽であり、Aが偽の場合、!A は真になる。

AND と OR が混在した条件式では AND の評価が OR よりも先に行われる。
AND、OR、または ! よりも算術演算子、比較演算子が先に計算される。
式の中で( ) を使い、計算の順序を示すことができる。

コメント

プログラムの記述中に # が現れた場合、そこから行末までの文字列はコメントとみなし、実行されない。ただし、文字列の中に現れた # はコメントとしては扱わない。

プログラムの例

(1) 右のプログラムで関数 main を実行すると、1＋2＋3＋… と加算を繰り返し、その値を表示する。合計が 100 を超えたら関数の実行は終わる。

関数 add() は値を返す関数、関数 accumulate() は値を返さない関数の例になっている。関数 accumulate() の最後に return 文はあってもなくてもよいことに注意。

```
var sum # グローバル変数の宣言

func add(n)
    sum = sum + n
    return sum
end

func accumulate()
    var i, s # ローカル変数
    sum = 0  # グローバル変数を初期化
    for (i = 1; i < 100; i = i+1)
        s = add(i)
        print(s)
        if s > 100
            return
        end
    end
end
```

(2) 1から9までの整数に対応するローマ数字を表示するプログラム。ローマ数字の表記で用いられる3種類の文字を、関数の第2引数として配列で渡している。

複雑な if 文の例となっている。

```
func printRoman(arabic, figures)
    var d
    if arabic == 9          # 9の場合, "IX"
        print(figures[0])
        print(figures[2])
    else if arabic == 4     # 4の場合, "IV"
        print(figures[0])
        print(figures[1])
    else
        if arabic >= 5      # 5, 6, 7, 8 の場合, "V"
            print(figures[1])
            d = arabic - 5
        else
            d = arabic
        end
        while d > 0         # 必要なだけ "I" を表示
            print(figures[0])
            d = d - 1
        end
    end
end

var figs[] = [ "I", "V", "X" ]
printRoman(8, figs)         # "VIII" が表示される
```