2023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１４ページあります。
5. 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（A）～ 設問（B）に答えなさい。

設問（A）

次の説明を読んで、空欄 ア ～ エ に入れる適切な語を答えよ。

インターネット経由でコンピュータがデータを送るとき、送信者はデータを小さく分割し、これに宛先などの情報を付加したパケットとして送り出す。

この付加した部分を ア と呼ぶ。受信者は受け取ったパケットから ア を取り除いて元のデータを復元して利用する。

図 1. はインターネット越しにコンピュータが通信する場合を模式的に表したものである。

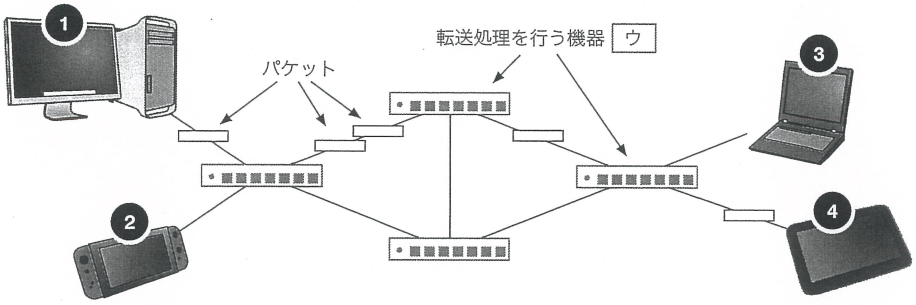

図 1. ネットワークとそこに接続された複数のコンピュータ

インターネットに接続されている各コンピュータには、それぞれを識別するための イ が付けられている。世界中に散在するコンピュータに正しくパケットを届けるために、経路の途中にはパケットを適切な方向に転送する ウ と呼ばれる機器が接続されている。

このようなパケット通信は、従来電話などで用いられてきた エ 方式とは異なり、他の人と同時に経路を共有して通信することになる。

設問（B）

以下の説明について、それぞれの記述の正誤を○と×で示せ。

a [ア] には宛先と送信元の両方の [イ] 情報が含まれている。

b [ア] には www.kyoto-su.ac.jp といったドメイン名と呼ばれる文字列が含まれている。

c IP プロトコルでは、転送途中でパケットが失われることがある。

d 二台のコンピュータ（たとえば図 2. の ① と ④）が通信する場合、[エ] の経路設定に間違いがなくても往路（① から ④）と復路（④ から①）では異なる経路を通ることがある。

〔Ⅱ〕 以下の文章を読んで、設問（A）～ 設問（C）に答えなさい。

作成したプログラムが本当に正しく動作するのか確認したい。そのために、プログラムに与える入力と、その入力に対する出力のペアをいくつか用意する。作成したプログラムに用意した入力を与えて得られた出力と、用意しておいた出力が異なっていればプログラムが間違っていることがわかる。逆に、用意したすべての入力に対して，得られた出力がそれぞれ用意した出力と一致すれば、プログラムは正しく作成できたと考えることにする。

例えば、2 つの整数値 n と d を入力とし、n が d で割り切れると 1、割り切れなければ 0 が出力されるプログラム *dividable* について考える。この *dividable* に対する入力と出力のペアの例を表 1 に示す。表 1 に挙げた n と d の値は例であり、入出力のペアが上で述べた *dividable* の説明と合っていれば他の数値であっても良い。例えば、n=12、 d=3、 出力=1 などでも良い。

n	d	出力
4	2	1
9	3	1
7	4	0

表 1 *dividable* への入出力のペアの例

一方、*dividable* への入力のうち d を 0 とすると 0 除算となり，正しい結果を出力できない。このようなエラーとして扱うべき入力のことをエラー入力と呼ぶ。エラー入力の具体的な値はプログラムにより異なるものになる。

これらのことを踏まえて以下の設問に答えよ。

設問（A）

入力として与えられた 2 つの数値 x、 y を比較し、その大小により結果が変わるプログラム *compare* を作成した。このプログラムは、x が y より小さければ −1、x が y より大きければ 1、x と y が等しければ 0 を出力する。この *compare* に対する入出力の例を表 2 にまとめた。表 2 の空欄部分を適切に埋めよ。

x	y	出力
		−1
		0
		1

表 2 *compare* への入出力のペアの例

設問（B）

二次方程式 $ax^2 + bx + c = 0$ の解を求めるプログラム *quadratic* を作成した。a、 b、 c を入力とするとき、解が実数解、重解、虚数解となるような a、 b、 c の組み合わせとその時の出力の例を表 3 にまとめた。表 3 の空欄部分を適切に埋めよ。ただし a は 0 以外であるとする。

a	b	c	出力
			$2, -2$
			3
			$1 \pm \sqrt{3}i$

表 3 *quadratic* への入出力のペアの例

設問（C）

3 つの整数値 *s*1、 *s*2、 *s*3 が与えられたとき、これらを辺の長さとする三角形の種類を出力するプログラム *triangle* を作成した。*triangle* は *s*1、 *s*2、 *s*3 で構成される三角形が、正三角形であれば 1、二等辺三角形であれば 2、その他の三角形であれば 3 を出力する。*triangle* への入出力の例を表 4 にまとめた。表 4 の空欄部分を適切に埋めよ。

*s*1	*s*2	*s*3	出力
			1
			2
			3

表 4 *triangle* への入出力のペアの例

また、エラー入力となる *s*1、 *s*2、 *s*3 の組み合わせについて考え、その例を解答用紙の表 5 に記入するとともに、エラーとなる理由も説明せよ。例が複数考えられる場合、そのうちの 1 つを答えよ。

〔Ⅲ〕 次の文章を読んで、設問（A）～ 設問（B）に答えなさい。

図１のように、丸で表された場所（ノード）と、２つのノードの間を結ぶ線（エッジ）を使って、地図上で都市間の接続などを表現する方法を考える。ここで、ノードの丸の中の記号（A～E）はノードの名前、エッジの横に書かれた数字はノード間を移動するのに必要なコスト（距離）である。

ここで、図１において始点から終点に至る最短経路とその距離（経路上のコストの合計）を求めたい。なお最短経路とは、始点から終点に至る経路のうち距離が最小であるものとする。最短経路を求めるアルゴリズムはカーナビゲーションの経路計算、インターネット上のパケットの配送経路を決める場合など様々な分野で応用されている。

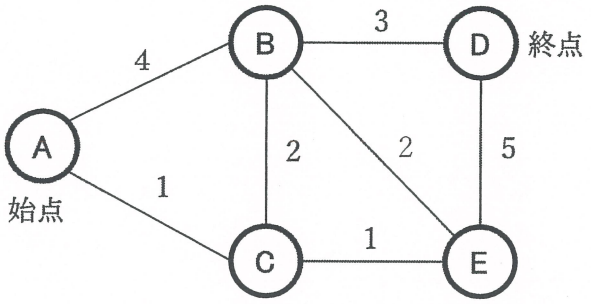

図 1. ノードとエッジを使った経路図

設問（A）

図 1 における始点から終点に向かう最短経路とその距離（コストの合計）を手作業で求めてみると、最短経路は【 ① 】→【 ② 】→【 ③ 】→【 ④ 】となり、その距離は【 ⑤ 】となる。

【 ① 】～【 ⑤ 】に適切な記号や数値を入れなさい。

設問（B）

ノードやエッジの数が増えると、手作業では最短経路を求めるのに時間が掛かりすぎてしまう。そこで最短経路とその距離をコンピュータで求めるための方法について考える。ここでは次のような手順で、すべての経路とその距離を始点のノードを根（ルート）とする木構造で表現していく。

(1) 始点のノードを一番上に描き、まずここを現在注目しているノードとする。

(2) 現在注目しているノードにつながる、全てのエッジとそれにつながるノードを矢印で結んで、現在注目しているノードの下方に追加していく。ただし、既に通ったノードは追加しない。

(3) 追加したノードを現在注目しているノードとし、手順(2)を繰り返す。ただし、追加したノードが終点であった場合はそれ以上は追加せず、ルートからの全てのエッジのコストの合計を求めてノードの下に距離として書く。

(4) 全てのノードについて手順(2), (3)が終了した後、全ての距離の中で最も小さいものが最短経路の距離となり、その終点のノードとルートとの経路が最短経路となる。

例として、図 2-1 に示す経路図に対して上記の手順を適用すると、図 2-2 のような木構造が得られ、最も小さい距離が 6 で、a → c → b → d が最短経路であることが分かる。ここで、最短経路となった終点のノードは二重丸にしている。

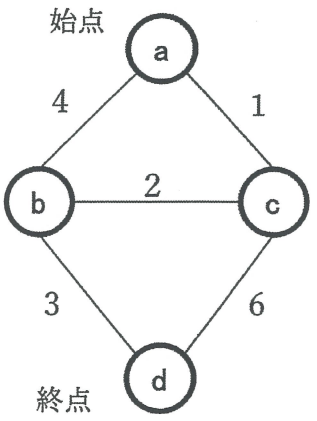

図 2-1. 経路図の例

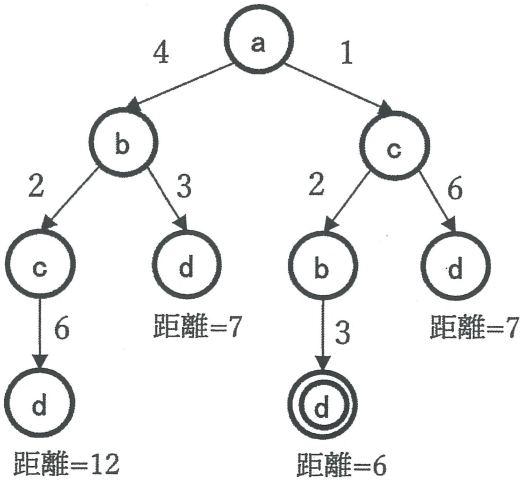

図 2-2. 左の経路図を木構造にして距離を求めたもの

図 1 で示した経路図について上記の手順で木構造を作成し、図 2-2 のようにそれぞれの経路の距離を明記せよ。また、最短経路となった終点のノードは二重丸にせよ。

解答用紙には、作成する木構造の一部が既に描いてあるので、残りの部分を完成させればよい。

〔Ⅳ〕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〔Ⅳ〕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。

ヒカルさんは“end”と“and”のように、文字数が同じだが 1 文字だけ異なる英単語がどれくらい存在するのかが気になって調べることにした。

設問（A）

ヒカルさんはまずそれぞれの単語の先頭から 1 文字ずつ取り出して比較し、異なる文字が幾つあるか数えれば良い、と考えた。それが 1 になる 2 つの単語が「1 文字だけ異なる単語のペア」である。

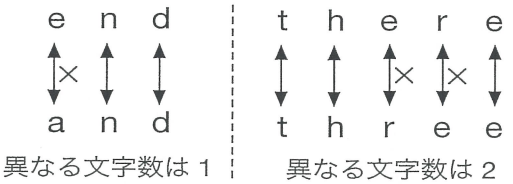

図 1. 異なる文字を数えた例

図 1. に、そのようにして異なる文字を数えた例を示す。“end”と“and”であれば異なる文字数は 1 となる。“there”と“three”であれば異なる文字数は 2 である。

ヒカルさんはこの処理を実現する手順を検討し、図 2. のフローチャートを描いた。変数 word1 と word2 にそれぞれ文字列“end”と“and”を与えた場合の処理をこのフローチャートに従って手作業で追うと、異なる文字の数を意味する変数 count の値が正しく 1 となり、「1 文字違い」と表示されることが確認できた。

図 2. の条件分岐について、空欄 ア と イ に入れる適切な条件式を、フローチャートに現れる変数を用いて解答せよ。

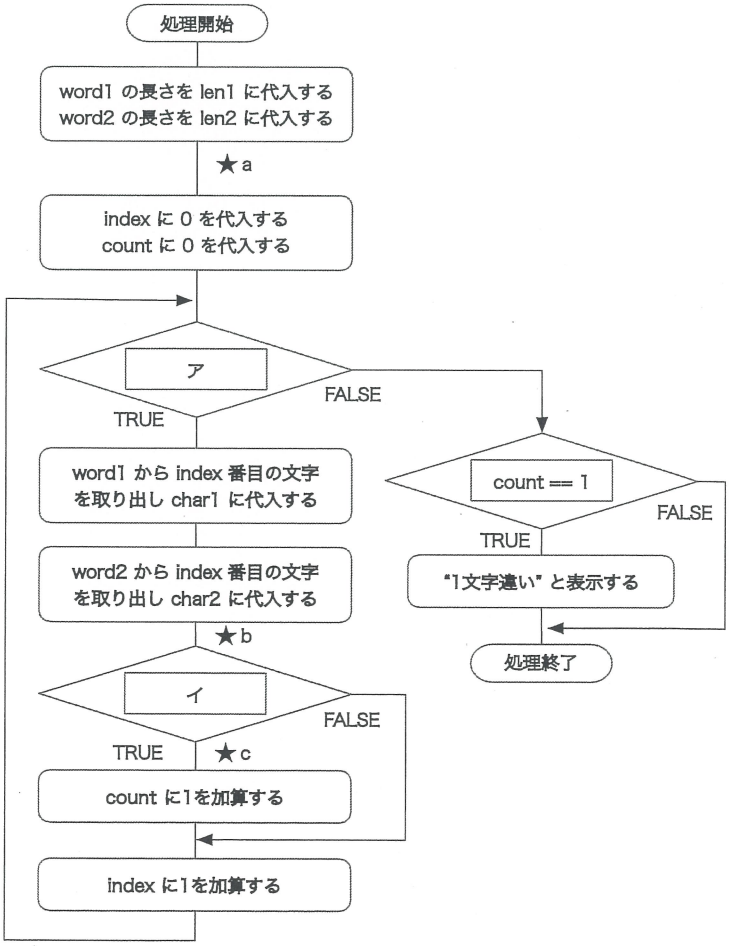

図 2. 1 文字違いであることを判定する手順

設問（B）

ヒカルさんは、図 2. のフローチャートに従ってプログラムを作成し、英単語辞書から適当に 2 単語を抽出して試験的に実行してみた。すると“away”と“aware”、また“audio”と“auditory”などの単語ペアでも 1 文字違いであると出力されてしまうことが分かった。

図 2. のフローチャートのどこかに、図 3.のような条件分岐を挿入することで、このような単語ペアも 1 文字違いではないと判定できる。フローチャートのどこに、どのような条件式をもつ分岐を挿入すれば良いかを解答せよ。

挿入箇所は、図 2. 中の★a, ★b, ★c のいずれかで示せ。挿入する条件分岐の条件式は、設問(A)と同様にフローチャートに現れる変数を用いて解答せよ。条件式が TRUE だった場合の分岐先は挿入したすぐ下の処理、FALSE だった場合は処理終了とする。

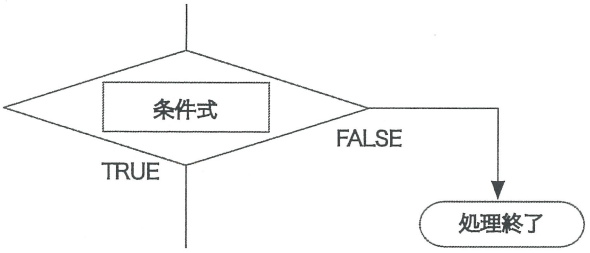

図 3. 挿入すべき条件分岐

設問（C）

ヒカルさんは手元の英和辞典から単語を 100 語抽出してテスト用の単語データを作成した。この単語データに含まれる単語の中で 1 文字違いとなるペアをすべて発見するためには、何回、単語の比較処理が必要か解答せよ。

設問（D）

ヒカルさんが単語データを 80,000 語に拡大し、ここに含まれる単語の中で 1 文字違いとなるペアをすべて発見するようプログラムを修正して実行すると、実行時間が 30 秒ほど掛かってしまった。この実行時間を短縮し得る改善案について説明せよ。

図 2. に示した手順を大きく変えず、無駄な処理を省くような修正案もあり得るし、まったく異なるアルゴリズムによる高速化案もあり得るだろう。思いつく手法のなかで、解答者がより優れていると考える二案まで解答に含めて良いものとする。

ただし「並列処理を行えば良い」といった、極端に短く、具体的な手法の説明を一切含まないような解答については、問題に対する理解度を評価する材料がほとんど含まれないことを理由として評価が極端に低くなることに注意せよ。

〔V〕 以下の文章を読んで、設問（A）～ 設問（C）に答えなさい。ただし、この問〔V〕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。

ある教育玩具はプログラムで制御することができ、表示装置として７セグメントLED（図１）が取り付けられている。この LED 表示装置は７つの発光部分を持ち、どの箇所を発光させるかによって数字、およびある程度の英文字や記号を表すことができる。図１の例は数字の「４」を表示している。この教育玩具には２桁の数字が表示できるように２個の７セグメント LED が取り付けられている。

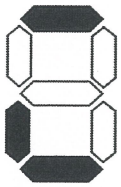

図１．７セグメントLED

この教育玩具は７セグメント LED の表示をプログラムから制御するための特殊な変数を備えており、この変数に書き込まれた値で７つの LED のどれが点灯するかが決まる。2 桁の７セグメント LED の左側と右側に対応する変数はそれぞれ led0 と led1 である。

なお、これらの変数に３や４といった整数値を書き込んでも数字の「３」や「４」が表示されるわけではなく、想定外の数箇所が点灯するだけである。適切に数字を表示させるには、その数字の形を構成する LED だけを点灯させる値を用意しておき、その値をこれらの変数に書き込む必要がある。

そこで、以下のプログラムでは７セグメント LED に数字を表示するための整数データをあらかじめ配列 digit に用意している。digit[0] の値を変数 led0 または led1 に格納すると７セグメント LED に数字「０」が表示される。同様に、digit[1] から digit[9] には数字の「１」から「９」を表示するための値が格納されている。なお、変数 led0 または led1 に整数値 0 を書き込んだ場合には、７つの LED をすべて消灯させることができる。

設問（A）

プログラム１は、変数 num の値を２桁の２セグメント LED に表示する。なお、変数 num の値は 0 から 99 の範囲にあるものとする。また、t0, t1 は変数で、関数 int() は実数値の小数点以下を切り捨てた整数値を返す。たとえば num = 32 の場合、左側の LED に「３」、右側の LED に「２」が表示される。

```
t0 = int(num / 10)
t1 = num % 10
led0 = digit[t0]
led1 = digit[t1]
```

プログラム１

ところで、プログラム１は変数 num の値が 10 より小さい場合でも左側の桁に「０」を表示してしまう。プログラム１を修正し、変数 num の値が 10 より小さい場合には、左側の LED を点灯させないようにしなさい。なお、必要に応じて変数 a を追加して使ってもよい。

設問（B）

友人から、ランダムな数を表示させるパーティーゲーム用のプログラムを作成して欲しいと頼まれた。内容は次の通り。

「３秒間、ランダムな数字を次々に表示する。その後、1 から 36 の間の整数をランダムに１つ選び、３回点滅させる。」

試作したものがプログラム２である。rnd()と sleep() はあらかじめ用意されている関数で、関数 rnd() は実行のたびに 0 から 32767 までの範囲の整数値をランダムに返す。関数 sleep() は指定した秒数だけ実行を停止する。n, val, t0, t1 は変数である。

```
for (n = 0; n < 30; n = n + 1)
    led0 = rnd() % 10
    led1 = rnd() % 10
    sleep(0.1)    # 0.1秒実行を停止
end
val = rnd() % 36
t0 = digit[int(val / 10)]
t1 = digit[val % 10]
for (n = 0; n < 3; n = n + 1)
    led0 = t0
    led1 = t1
    sleep(0.5)   # 0.5秒実行を停止
end
```

プログラム２

ところが実行の様子を見て、依頼と異なる動作を指摘された。問題点は次の３つである。

(1) 最初の３秒の間に、数字ではない表示が次々に出る。

(2) 最後に選ばれた値が０の場合がある。

(3) 点滅の動作が行われず、選ばれた値が表示されたままになる。

これらの点を修正したプログラムを記述しなさい。なお、9 以下の整数を表示する際、左の桁に０を表示するのは問題ない。また、必要に応じて変数 i, j, a, b を追加して使ってもよい。

設問（C）

図２に、制御用の変数に格納される値と７セグメント LED の点灯箇所の関係を示す。

制御用の変数 led0、led1 には８ビットの整数値を格納でき、下位７ビットが LED の点灯状態に対応している。たとえば最下位ビットが１の場合、図の a の位置の LED が点灯し、0 ならば消灯する。図２は数字の３と２を表すように LED を点灯させる例を示しており、それぞれの変数には 16 進表記で 4F、5B の整数値を格納すればよい。

図３のように数字の「４」を表示させるために変数 led0 または led1 に格納するべき値は何か、16 進表記で答えなさい。

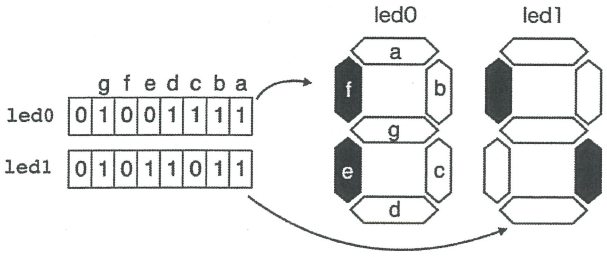

図２．制御用の変数と７セグメントLEDの関係

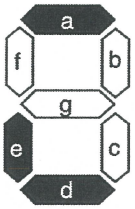

図３．数字の「４」の表示

（白紙）

プログラムの記法の説明

問〔Ⅳ〕および〔Ⅴ〕で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	条件式が真である間、end までの命令を実行する。
if 条件式 ... else ... end	条件式が真のとき、else までの命令を実行し、偽のとき、else から end までの命令を実行する。else を記述しない場合、条件式が真のときに end までの命令を実行し、条件式が偽のときは何もしない。
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。

配列

配列名 [式]	含まれる要素の数があらかじめ決められた配列を利用できる。 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。

比較演算子

A == B	AとBの値が等しい。
A != B	AとBの値が等しくない。
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。

算術演算子

+	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。

複数の算術演算子が混在した式では * と / 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。
また、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。

プログラムの例

(1) 1＋2＋3＋… と加算を繰り返し、その値を変数 s に代入する。値が 100 を超えたら終わる。

```
s = 0
i = 1
while s <= 100
    s = s + i
    i = i + 1
end
```

```
# 左と同じプログラムを for を使って書いたもの
s = 0
for (i = 1; s <= 100; i = i + 1)
    s = s + i
end
```

(2) 与えられた要素数 n の配列 a の内容を、同じ大きさの配列 b にコピーする。ただし、配列 a の要素で負の数があれば、b には代わりに 0 を代入する。

```
for (i = 0; i < n; i = i + 1)
    if a[i] >= 0
        b[i] = a[i]
    else
        b[i] = 0  # aの要素が負数の場合
    end
end
```