2022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１１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 〕　次の説明を読んで、空欄 ア ～ ケ に入れるのに最も適当なものを、下のそれぞれの解答群のうちから一つずつ選びなさい。

現在のコンピュータはそのほとんどが ア と呼ばれるモデルに基づいたものである。図1にその典型的なモデルを図示する。

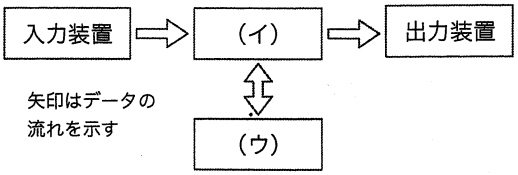


図1. コンピュータのモデル図

図中の イ が ウ に格納された エ を読み出し、逐次的に実行する。この エ の列のことを一般に オ と呼ぶ。

我々が日常的に使っているスマートフォンやパソコンも ア のコンピュータである。しかし現在ではそれらの大部分は カ ことに注意が必要である。

ア ～ オ の解答群：

1. パソコン	2. スパコン	3. 機械語
4. ノイマン型	5. デスクトップ型	6. ノート型
7. ハーバード型	8. ハードディスク	9. フラッシュメモリ
10. 主記憶装置	11. 補助記憶装置	12. CPU
13. CPU 命令	14. GPU	15. ネットワーク
16. データ	17. ソースプログラム	18. ソフトウェア

カ の解答群：

1. 高速化のためにハードディスクなどの補助記憶装置を使っていない
2. 処理能力向上のために CPU を複数搭載している
3. 小型化のために CPU をクラウド上に移動している
4. ソフトウェアのアップデートによって CPU を変更できる
5. セキュリティ対策のために主記憶装置を使っていない

近年、ディープ・ラーニング（深層学習）と呼ばれる手法によって、画像認識などの処理能力が格段に向上した。しかしディープ・ラーニングは内部的には［キ］の繰り返しであり、［ク］で実行するには不向きなため、高速化のために［ケ］と呼ばれる半導体チップが使われることが増えた。

［キ］～［ケ］の解答群：

1. 論理演算	2. 高速処理	3. 行列演算
4. ネットワーク処理	5. 画像処理	6. CPU
7. GPU	8. TPU	9. SSD
10. DPU	11. 5G	12. IoT

〔II〕 次の説明を読んで、設問(A)、(B)に答えなさい。

データの探索（記録したデータの中に指定する値が存在するかどうかを調べる作業）を高速化するために、データを関連づけて配置する手法がある。図1はその一例である。丸印はデータの存在を示し、数字はその値を意味する。探索処理は start が指すデータから始め、左下あるいは右下に向かって線をたどることで行う。

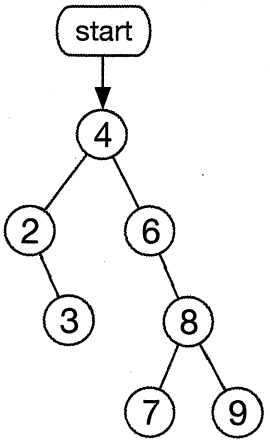


図 1.

データを結ぶ左下および右下向けの線について説明する。それぞれのデータ（例えば4）から左下に出ている線の先には、必ず自分のもつ値(4)より小さな値(2)が置かれている。右下に出ている線の先には、必ず自分のもつ値(4)より大きな値(6)が置かれている。

このようなデータ配置で、ある値を探索する手順について図2を用いて説明する.

たとえば入力値8が存在するかどうかを調べる場合、まず start が指すデータ4に注目し、その値(4)と入力値(8)を比較する。入力値の方がより大きいため、右下に線をたどって進み、6と入力値(8)を比較する。結果、より大きいため更に右下に進んで比較すると、その値(8)が入力値(8)と一致した。なお、もし比較結果がより小さい場合は左下に進めば良い。

つまり3回の比較作業で8は存在することがわかり、探索処理は完了した。

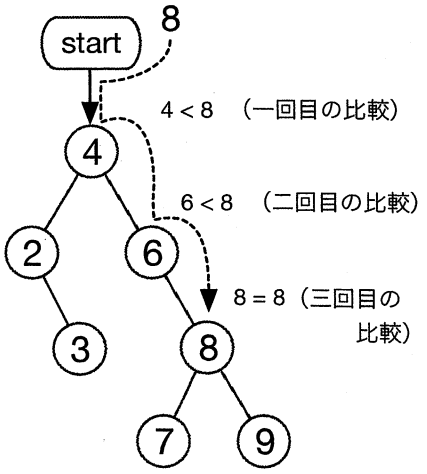


図 2.

設問 (A)

以下の空欄 ア ～ エ にあてはまる数値を答えなさい。

図1および図2に示したデータを対象に、上に示した手順によって以下のそれぞれの入力値が存在することを確かめるために必要な比較作業は何回になるか。

・入力値が 9 の場合、比較は ア 回

・入力値が 3 の場合、比較は イ 回

同様に以下のそれぞれの入力値が存在しないことを確かめるために必要な比較作業は何回になるか。

・入力値が 10 の場合、比較は ウ 回

・入力値が 5 の場合、比較は エ 回

設問（B）

以下の空欄 [オ] にあてはまる数値を答えなさい。また説明を読んで解答欄の図を完成させなさい。

ある値を探索する処理において、データの配置を工夫すると最悪ケースの比較回数を少なくできる場合がある。

たとえば図3左側の状態から、右側の状態に配置を変えることで、当初9を探索するために3回の比較作業が必要だったところを、[オ] 回に減らすことができる。

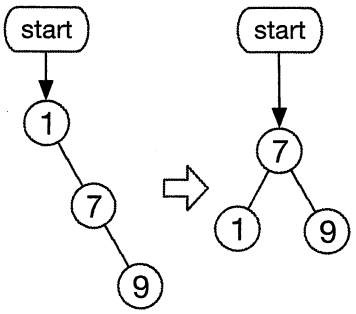


図3.

以下の図4，図5のデータ配置においても、同様に工夫できる部分があるので、そのように変形させる。解答欄のデータを示す丸印の中に数値を記入して図を完成させよ。

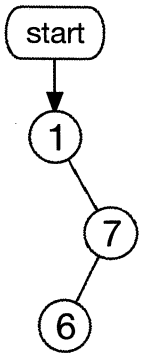


図4.

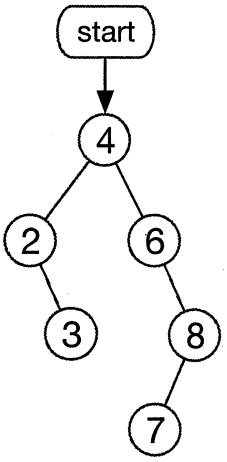


図5.

〔Ⅲ〕 以下の文章を読み、設問（A）～ 設問（D）に答えなさい。

論理値（真と偽、あるいは 1 と 0）のみを値とする変数または式 A と B に対し、A と B の論理和を A or B、論理積を A and B で表す。また、A の論理否定を not A で表す。式の中に or, and, not が同時に現れた時は、not, and, or の順に優先して評価し、また式の中では必要に応じて（ ）が使えるものとする。

変数が持つ真偽値の組み合わせに対して、その変数を含む式の値を表の形式で示したものを真理値表と呼ぶ。たとえば、表 1 は変数 A と B のすべての値の組み合わせに対して、式 A or B の値を示す真理値表である。

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

表 1　A or B の真理値表

設問（A）

論理値を持つ任意の変数または式 A, B, C について、(式 1) の等式が常に成り立つ。これを分配法則と呼ぶ。

$$A \text{ and } (B \text{ or } C) \quad = \quad A \text{ and } B \text{ or } A \text{ and } C \qquad \text{(式 1)}$$

分配法則が成り立つことを確認するために、表 2 のような真理値表を作成した。この真理値表は A, B, C のすべての値の組み合わせについて、(式 1) の両辺の値を調べようとするものである。空欄をすべて埋めて真理値表を完成させなさい。

A	B	C	B or C	A and (B or C)	A and B	A and C	A and B or A and C
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0					
1	0	1					
1	1	0					
1	1	1					

表 2　分配法則の真理値表

設問（B）

論理値を持つ任意の変数または式 A と B について、(式 2) の等式が常に成り立つ。これをド・モルガンの法則と呼ぶ。

$$\text{not A or not B} \quad = \quad \text{not(A and B)} \qquad \text{(式 2)}$$

ド・モルガンの法則は（式 3）のように表現することもできる。

$$\text{not A and not B} \quad = \quad \text{not(A or B)} \qquad \text{(式 3)}$$

（式 2）と（式 3）は同じ等式の別の表現である。（式 2）において A = not X, B = not Y とおいて式を変形すれば（式 3）を導くことができる。このことを示しなさい。

設問（C）

2つの実数に対して ＞, ≧, ＜, ≦ を用いて比較演算を行うことができる。比較演算の結果は真または偽である。比較演算と論理演算を組み合わせると、複数の実数の間の関係を表す式を記述できる。たとえば次の式は、変数 x が 0 より大きくかつ 5 以下であることを示す（x が 0 より大きくかつ 5 以下である場合かつそのときに限って式が真になる）。

$$(0 < x) \text{ and } (x \leqq 5)$$

m ＜ n である 2 つの実数 m と n に対して、m 以上かつ n 以下である実数の集合を**範囲**と呼び、[m, n] で表す。

範囲 [m, n] と実数 a, b を考える。ただし、 a ＜ b とする。このとき、実数 a と b の両方が同時に範囲 [m, n] に含まれることを示す式（実数 a と b がどちらも範囲 [m, n] に含まれる場合かつそのときに限り真になる式）は次のように記述できる。空欄を埋めて式を完成させなさい。

$$(\ \square\ ) \text{ and } (\ \square\ )$$

設問（D）

範囲 [m, n] と範囲 [p, q] について考える。この 2 つの範囲が重ならない（2 つの範囲に同時に含まれる実数が存在しない）ことを示す式は次の（式 4）のように記述できる。

$$(n < p) \text{ or } (q < m) \qquad \text{(式 4)}$$

（式 4）を否定し、ド・モルガンの法則を適用して変形すると（式 5）のような結果が得られる。空欄を埋めて式を完成させなさい。また、（式 5）が範囲 [m, n] と範囲 [p, q] のどのような関係を示すのか（どのような場合に式の値が真となるのか）、説明しなさい。

$$\text{not}((n < p) \text{ or } (q < m))$$

$$= \ (\ \square\ ) \text{ and } (\ \square\ ) \qquad \text{(式 5)}$$

〔IV〕 以下の文章を読んで設問（A）～（D）に答えなさい。

図１は制服を着る順序の制約を示したものである。円で囲まれている「シャツ」や「ネクタイ」を仕事と呼ぶことにする。ある仕事を行うためには，その仕事に矢印で示された順序の制約を満たす必要がある。ただし，仕事は一つずつ行わなければならない（並行にはできない）。例えば，「ジャケット」を着るという仕事は，「ベルト」と「ネクタイ」を着るという仕事の後でないと行えない。

図１の制約を満たした上で，実行可能な順序の例としては，（1）シャツ，（2）ネクタイ，（3）ズボン，（4）ベルト，（5）ジャケット，（6）靴下，（7）靴，が挙げられる。

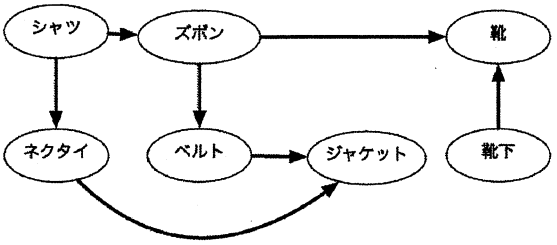


図 1 服を着る順序の制約の例

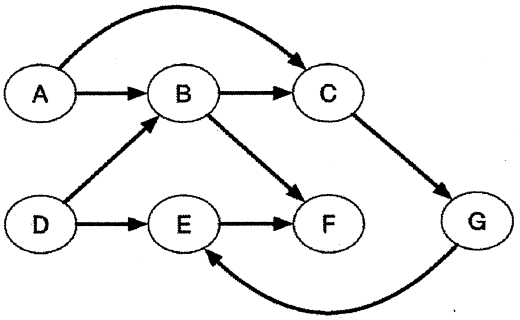


図 2 仕事 A～G の実行順序の制約条件

設問（A）

図２は仕事 A～G の実行順序の制約を示したものである。図２の制約を満たした上で，仕事を行う順序を示しなさい（複数解答が考えられる場合はそのうちの一つを答えなさい）。

設問（B）

図２の制約を満たした仕事の実行順序は全部で何通りあるか答えなさい。

設問（C）

設問（A）を解くにあたって考慮すべき事項を列挙しなさい。

設問（D）

図２に矢印を新たに追加するか既存の矢印を削除することで順序付けができなくなる場合がある。そのような矢印を解答欄の図に追加するか削除（矢印に×印を付ける）して，順序付けができない理由を説明しなさい（複数解答が考えられる場合はそのうちの一つを答えなさい）。

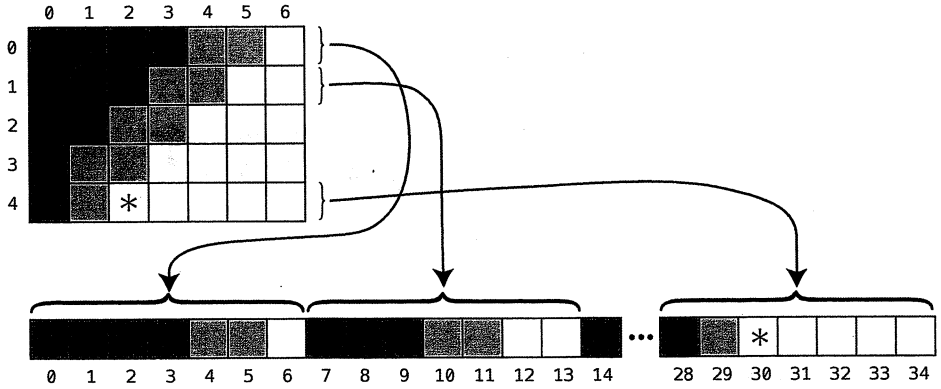
問題は、
次のページに続きます。

〔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。

ピクセル（画素）の集合として表される静止画像を、ラスタ形式の画像と呼ぶ。以下では、各ピクセルの明るさを１バイトで表す白黒（グレースケール）画像を考える。

プログラムでさまざまな大きさのラスタ形式の画像データを扱う場合、２次元配列ではなく、１次元配列を用いるのが一般的な方法である。多くの画像形式では、画像の行（横方向のピクセルの列）のデータを、最も上の行から下の行へ、左のピクセルから右へ向かってデータを並べる。したがって、画像の最も左上のピクセルが配列の先頭で、右下のピクセルが最後になるように格納される。横幅が W ピクセルの画像では、m 行目、n 列目の位置にあるピクセルは配列の先頭から (m×W+n)番目に格納される（先頭は０番目と数える）。

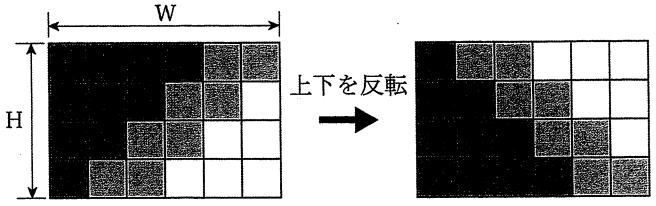
たとえば、下図の例は横が 7 ピクセル、縦が５ピクセルの画像であり、35 バイトで表現できる。4 行目、2 列目（図の＊印）のピクセルは配列では (4×7+2)=30 番目となる。



以下、本設問における配列は、すべて１次元配列であるとする。

設問（A）

横 W ピクセル、縦 H ピクセルの白黒画像が、W×H バイトの大きさを持つ配列 a に格納されているとする。配列 a と同じ大きさの配列 b が存在するとき、配列 a の画像データを、上下に反転（垂直方向に反転）させて配列 b に格納したい。

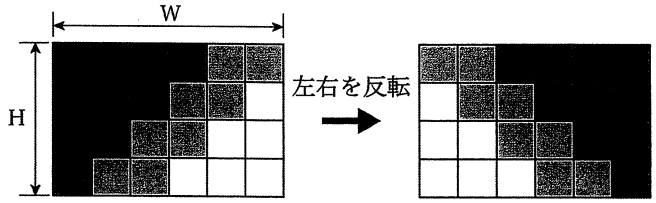


このために次のようなプログラムを作成した。ここで、変数 W, H, x, y, r, s はすでに用意されているとする。空欄部分に適切な記述を補いなさい。

```
for (y = 0; y < H; y = y + 1)
    [          ]
    for (x = 0; x < W; x = x + 1)
        b[r + x] = a[s + x]
    end
end
```

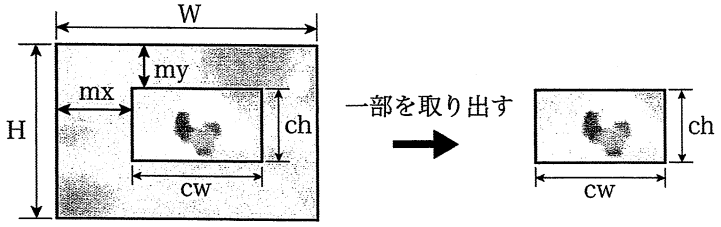
設問（B）

横 W ピクセル、縦 H ピクセルの白黒画像が、W×H バイトの大きさを持つ配列 a に格納されているとする。配列 a の内容を書き換えて、画像データを左右に（水平方向に）反転させるプログラムを記述しなさい。ただし配列 a 以外には配列は使わないものとする。変数 W, H はすでに用意されており、さらに変数 x, y, r, s, t, h, i を使ってもよい（すべてを使う必要はない）。



設問（C）

横 W ピクセル、縦 H ピクセルの白黒画像が、W×H バイトの大きさを持つ配列 a に格納されているとする。配列 a の一部分を取り出して新しい画像データを配列 b に作成したい。取り出す部分の左上のピクセルは my 行目、mx 列目で、取り出す部分の縦横の大きさはそれぞれ ch, cw ピクセルになるようにする。変数 W, H, my, mx, ch, cw がすでに正しく与えられているとして、配列 a の指定部分を配列 b に格納するプログラムを記述しなさい。配列 b は必要な大きさをもち、さらに、必要に応じて変数 x, y, r, s, t, h, i を使ってもよいとする（すべてを使う必要はない）。



プログラムの記法の説明

問〔V〕でプログラムの記述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。
print(式)	式の値を表示する。

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] という記法で表現する。

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

プログラム中に現れる /* と */ に囲まれた文字列はコメントとみなし実行されないものとする。

プログラムの例

(1) 1＋2＋3＋… と加算を繰り返して、その値を表示する。合計が 100 を超えたら終わる。

```
s = 0 /* 変数sに0を代入する */
i = 1 /* 変数iに1を代入する */
while s <= 100
    s = s + i
    print(s)
    i = i + 1
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
    end
end
```