Section 1-2 数の表現

知識確認

数学で用いられる数の分類として，次のようなものがある。

・自然数
　正の整数（0 を含まない 1 以上の値）を意味する数

・約数
　ある自然数を割り切れる自然数の集まり
　　例．自然数 8 の約数は 1，2，4，8 である。

・素数
　1 とその数以外の約数をもたない，2 以上の自然数
　　例．自然数 7 の約数は 1 と 7 だけであり，7 は素数である。

・最大公約数（G.C.D：Greatest Common Divisor）
　二つ以上の自然数に共通する約数のうち，最大値となる数
　　例．自然数 8（約数：1，2，4，8）と 12（約数：1，2，3，4，6，12）の
　　最大公約数は 4 である。

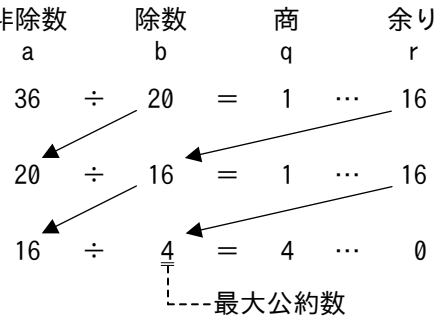
・最小公倍数（L.C.M：Least Common Multiple）
　二つ以上の自然数に共通する倍数のうち，最小値となる数。
　　例．自然数 8（倍数：8，16，24，…）と 12（倍数：12，24，36，…）の
　　最小公倍数は 24 である。

ユークリッドの互除法

二つの自然数の最大公約数を求める手法の一つに「ユークリッドの互除法」がある。ユークリッドの互除法は，「自然数 a を自然数 b で割った余りを r としたとき，a と b の最大公約数は，b と r の最大公約数と等しい」という定理を利用した手法である。ユークリッドの互除法を用いた，自然数 a と b の最大公約数を求める手順は，次のとおりである。

〔手順〕
(1) a を b で割った余りを r とする。
(2) a に b を代入し，b に r を代入する。
(3) 手順(1)と手順(2)を繰り返し，割った余りが 0 になったときの除数（割る数）が，最大公約数となる。

例．36 と 20 の最大公約数をユークリッドの互除法で求める。



整数の下位（上位）から任意の桁数の数値を取り出す処理

n 桁の 10 進数の整数 a について，次の計算によって，下位（または上位）から m 桁の数値を取り出すことができる。

下位 m 桁： a ÷ 10^m の余り
上位 m 桁： a ÷ $10^{(n-m)}$ の商

例．8 桁の整数 12,345,678 を1,000（＝10^3）で割った場合
12345678 ÷ 1000 ＝ 商 12345 余り 678
上位 5 桁　下位 3 桁

平方根

a の平方根とは，2 乗したときに a となる数のことを表す。このとき，a が正の数であれば，平方根は正の数と負の数の二つが存在し，正の数の方を「正の平方根」，負の数の方を「負の平方根」という。例えば，9 の平方根は，3 または −3 の二つが存在し，3 が正の平方根，−3 が負の平方根となる。

なお，数学ではルート記号（√ ）を用いて，正の平方根を$\sqrt{a}$，負の平方根を$-\sqrt{a}$と記述する。

基礎演習

問1　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

手続 gcd は，引数で与えられた二つの正の整数の最大公約数を表示する。
手続 gcd を gcd(144, 60) として呼び出すと，プログラム中の /*** α ***/ の行は［　　］回実行して終了する。

〔プログラム〕

```
○gcd(整数型: value1, 整数型: value2)
  整数型: large, small
  large ← value1
  small ← value2
  while (large が small でない)  /*** α ***/
    if (large が small より大きい)
      large ← large − small
    elseif (large が small より小さい)
      small ← small − large
    endif
  endwhile
  large を出力
```

解答群

ア　4　　イ　5　　ウ　6　　エ　7

問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

手続 proc は，引数で与えられた二つの正の整数の計算結果を表示する。

手続 proc を proc(a, b) として呼び出すと，プログラム中の /*** α ***/ の行で［　　］が出力される。

〔プログラム〕

```
○proc(整数型: a, 整数型: b)
  整数型: x, y, t
  x ← a
  y ← b
  while (y が 0 でない)
    t ← x ÷ y の余り
    x ← y
    y ← t
  endwhile
  x を出力  /*** α ***/
```

解答群

ア　a × b
イ　a ÷ b の商
ウ　a と b の大きい方に最も近い素数
エ　a と b の最小公倍数
オ　a と b の最大公約数
カ　a と b の小さい方に最も近い素数

問３　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

手続 primeNumber は，2 以上 100 以下の素数を表示する。

〔プログラム〕

```
○primeNumber()
  整数型: k, m, count
  for (k を 2 から 100 まで 1 ずつ増やす)
    count ← 0
    for (m を 2 から k まで 1 ずつ増やす)  // α
      if ( [　　] )
        count ← count + 1
        if (count が 2 以上)
          αの行から始まる繰返し処理を終了する
        endif
      endif
    endfor
    if (count が 1 と等しい)
      k を出力
      ”　”を出力
    endif
  endfor
```

解答群

ア　k ÷ m の余り が 0 と等しい
イ　k ÷ m の余り が 0 と等しくない
ウ　k ÷ m の余り が 1 と等しい
エ　k ÷ m の余り が 1 と等しくない
オ　k ÷ m の商 が 0 と等しい
カ　k ÷ m の商 が 1 と等しい

問4　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

関数 getMonth は，引数で与えられた日付データから月の部分を返す。ここで，日付データは 8 桁の 10 進数（yyyymmdd）とし，yyyy は西暦年，mm は月，dd は日を示す。

〔プログラム〕

```
○整数型: getMonth(整数型: date)
  整数型: d ← date
  [　　]
  d ← d ÷ 100 の商
  return d
```

解答群

ア　d ← 100 ÷ d の余り
イ　d ← 10000 ÷ d の余り
ウ　d ← 1000000 ÷ d の余り
エ　d ← d ÷ 100 の余り
オ　d ← d ÷ 10000 の余り
カ　d ← d ÷ 1000000 の余り

問5　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

関数 calcDistance は，引数で与えられた二つの点 A(x1, y1) と B(x2, y2) について，座標平面上の線分 AB の長さを返す。ここで，引数 x1, y1, x2, y2 は，式 x1 ≦ x2, y1 ≦ y2 がそれぞれ成立しているものとする。

〔プログラム〕

```
○実数型: calcDistance(実数型: x1, 実数型: y1, 実数型:x2, 実数型: y2)
  実数型: dx, dy
  dx ← x2 − x1
  dy ← y2 − y1
  return［　　］
```

解答群

ア　(dxの2乗 ＋ dyの2乗) の正の平方根
イ　(dxの2乗 − dyの2乗) の正の平方根
ウ　(dyの2乗 − dxの2乗) の正の平方根
エ　dxの2乗 ＋ dyの2乗
オ　dxの2乗 − dyの2乗
カ　dyの2乗 − dxの2乗

実践演習

問題　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

手続 specifyPrimeNumber は，引数で与えられた数値以下の素数を出力する。ここで，引数で与えられる数値は 2 以上の整数とする。

〔プログラム〕

```
○specifyPrimeNumber(整数型: maxNum)
  整数型: num, n
  論理型: flag
  "2" を出力
  " " を出力
  for (num を 3 から maxNum まで 2 ずつ増やす)
    flag ← true
    for (n を 3 から [　　] まで 2 ずつ増やす)  // α
      if (num ÷ n の余り が 0 と等しい)
        flag ← false
        αの行から始まる繰返し処理を終了する
      endif
    endfor
    if (flag が true と等しい)
      num を出力
      " " を出力
    endif
  endfor
```

解答群

ア　maxNum
イ　maxNum ÷ 2 の商
ウ　maxNum の正の平方根の整数部分
エ　num
オ　num ÷ 2 の商
カ　num の正の平方根の整数部分

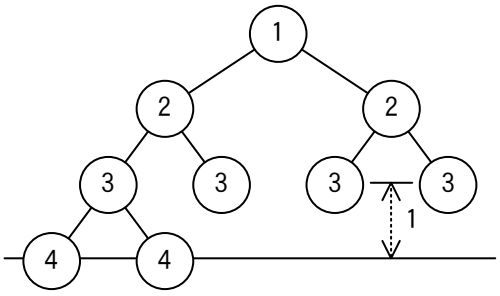
Section 2-6 木構造③

知識確認

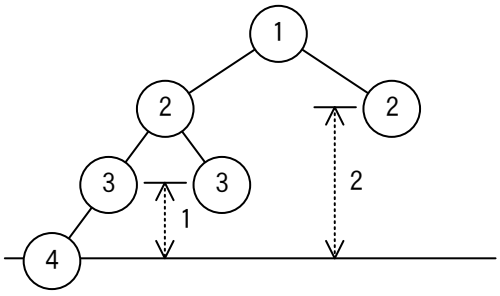
完全 2 分木

完全 2 分木は，すべての葉が左詰めになっていて，根からすべての葉に至るまでの枝の数（レベル）が等しいか，高々 1 しか違わない 2 分木である。

・完全 2 分木の例



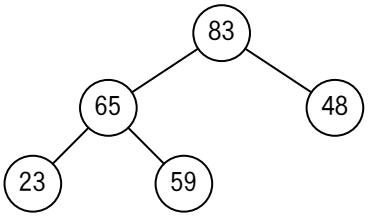
・完全 2 分木でない例



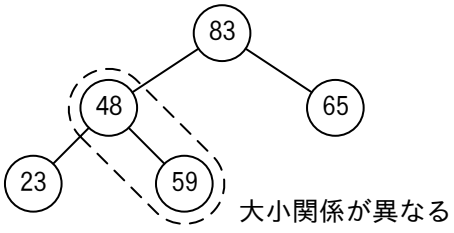
ヒープ

ヒープは，親ノードと子ノードの間に一定の大小関係が成立する完全 2 分木である。

・ヒープの例



・ヒープでない例



左図の例では，すべてのノードにおいて「親ノードの値 ＞ 子ノードの値」が成立するヒープとなっている。

一方，右図の例では，1 箇所だけ「親ノードの値 ＜ 子ノードの値」となっている。このように，1 箇所でも大小関係が異なるノードが存在する場合，その木構造はヒープではなくなる。

また，ヒープは，根の要素番号を 1 とし，親ノードの要素番号を i，左の子ノードの要素番号を {2 × i}，右の子ノードの要素番号を {2 × i + 1} とすることで，配列で表現することができる。左ページのヒープを配列で表現すると，次のようになる。

	[1]	[2]	[3]	[4]	[5]
配列 heap	83	65	48	23	59

なお，あるノードの要素番号を j とすると，その親ノードの要素番号は {j ÷ 2}（小数点以下切捨て）で求められる。

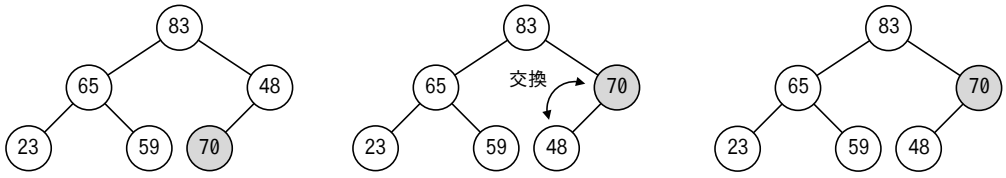
ヒープの再編成

新しいノードを追加したり，根を取り出したりした場合，完全 2 分木の構成を保つために「再編成」が行われる。それぞれの手順について，ここでは左ページのヒープを例に説明していく。

・ノードの追加

末尾にノードを追加し，追加ノードとその親ノードの大小関係が正しくなるまで，ノードの比較／交換を繰り返す。

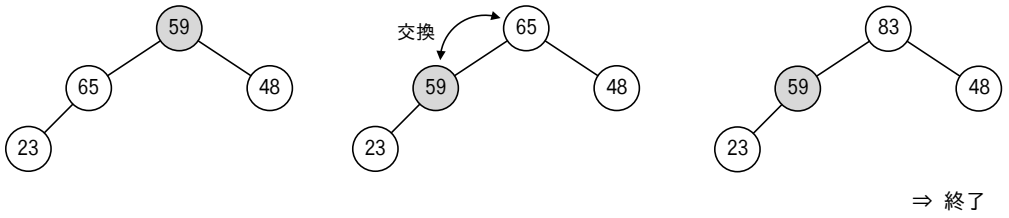
(1) 末尾にノードを追加　(2) 追加ノード ＞ 親ノード　(3) 追加ノード ＜ 親ノード



・根の取出し

根を取り出した後，末尾のノードを根があった位置に移動し，移動したノードとその子ノードの大小関係が正しくなるまで，比較／交換を繰り返す。

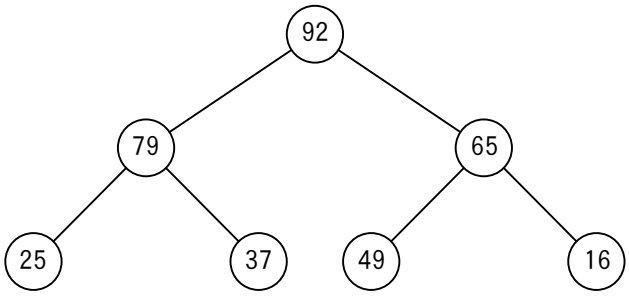
(1) 末尾ノードを根に移動　(2) 移動ノード ＜ 左の子ノード　(3) 移動ノード ＞ 子ノード



※ 根の取出しで，親ノードと左右の子ノードとの大小関係が両方とも正しくない状態になった場合，交換後にもう一方の子ノードとの大小関係も正しくなる方の子ノードと交換する（上図の例では，より大きい方の子ノードと交換する）。

基礎演習

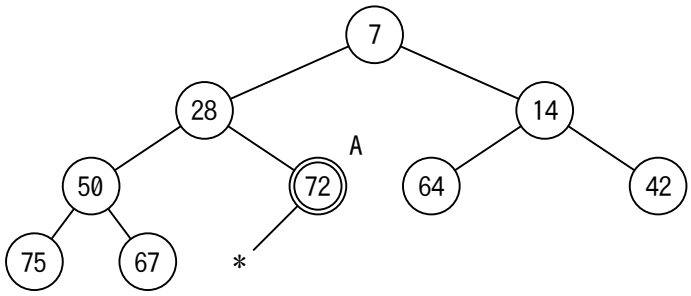
問1　親の値と子の値に一定の大小関係が成立するヒープにおいて，ヒープの配列表現を，根を tree[1] とし，親 tree[node] の左の子を tree[2 × node] に置き，右の子を tree[2 × node ＋ 1] に置くことで実現する場合，図のヒープの配列表現として，適切なものはどれか。ここで，配列の要素番号は 1 から始まる。



解答群

	tree[1]	tree[2]	tree[3]	tree[4]	tree[5]	tree[6]	tree[7]
ア	92	65	16	79	49	25	37
イ	92	65	79	49	37	16	25
ウ	92	79	49	37	65	25	16
エ	92	79	65	25	37	49	16

問２　親の値が子の値より小さいヒープがある。このヒープへの挿入は，要素を最後尾に追加し，その値が親の値よりも小さい間，親と子を交換することを繰り返せばよい。次のヒープの ＊ の位置に要素 10 を追加したとき，A の位置に来る要素はどれか。



解答群

ア　7　　イ　14　　ウ　28　　エ　72

問３　次のプログラム中の［　a　］，［　b　］に入れる正しい答えの組合せを，解答群の中から選べ。ここで，配列の要素番号は 1 から始まる。

手続 appendNode は，親の値が子の値よりも大きいヒープに，引数で与えられた数値 num をもつノードを追加する。

ヒープは，大域の整数型の配列 data を用いて表現する。配列 data では，ヒープの根の値を data[1] とし，親 data[i] の左の子の値を data[i × 2] に格納し，右の子の値を data[i × 2 + 1] に格納する。また，ヒープに格納されているノードの数は，大域の変数 n に記録されている。

配列 data と変数 n は図に示す状態であり，配列 data の未使用要素には −1 が格納されている。このとき，手続 appendNode の引数 num に 99 を与えてプログラムを実行した場合，プログラム中の /*** α ***/ の命令は［　　］回実行される。

	[1]	[2]	[3]	[4]	[5]	[6]	[7]
配列 data	97	76	85	62	38	25	−1

n ［6］

図　配列 data と変数 n を用いたヒープ

〔プログラム〕

```
大域: 整数型の配列: data
大域: 整数型: n

○appendNode(整数型: num)
  整数型: i, j, tmp
  i ← n + 1
  data[i] ← num
  j ← i ÷ 2
  while (j が 1 以上 かつ data[i] が data[j] より大きい)
    tmp ← data[i]
    data[i] ← data[j]
    data[j] ← tmp
    i ← j  /*** α ***/
    j ← i ÷ 2
  endwhile
  n ← n + 1
```

解答群

ア　0　　イ　1　　ウ　2　　エ　3

実践演習

問題　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，配列の要素番号は 1 から始まる。

関数 getRoot は，親の値が子の値よりも大きいヒープから，根の値を取り出して返す。なお，根の値を取り出したヒープは，末尾の子の値を根の値として移動した後，大小関係が正しくなるように再構築する。

プログラムにおいて，ヒープは整数型の配列 heap を用いて表現し，配列 heap 及び配列 heap に格納されているノード数 n は大域に定義している。また，配列 data の未使用要素には −1 が格納されている。

〔プログラム〕

```
大域: 整数型の配列: heap
大域: 整数型: n

○整数型: getRoot()
  整数型: i, j, tmp, ret
  ret ← heap[1]
  heap[1] ← heap[n]
  heap[n] ← −1
  n ← n − 1
  i ← 1
  while ((heap[i] が heap[i × 2] より小さい) または
         (heap[i] が heap[i × 2 + 1] より小さい))
    if ( [　　] )
      j ← i × 2
    else
      j ← i × 2 + 1
    endif
    heap[i] と heap[j] を交換する
    i ← j
  endwhile
  return ret
```

解答群

ア　(i × 2 + 1) が n より大きい
イ　(i × 2 + 1) が n より小さい
ウ　heap[i × 2] が heap[i × 2 + 1] より大きい
エ　heap[i × 2] が heap[i × 2 + 1] より小さい

Section 3-1 行列

知識確認

行列は，いくつかの数を長方形状，または正方形状に並べたものである。横に並んだひとまとまりを「行」，縦に並んだひとまとまりを「列」といい，行列を構成する一つひとつの要素を「成分」という。

また，行数と列数が等しい（正方形状に並んだ）行列を「正方行列」という。

$$\begin{bmatrix} 2 & 5 & 2 \\ 0 & 1 & 7 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & 0 \\ 5 & 2 & 3 \\ 4 & 1 & 1 \end{bmatrix}$$

2 行 3 列の行列　　3 行 3 列の正方行列

行列の積（行列どうしの乗算）

行列 A と B の積 AB の演算では，A の i 行目の各成分と B の j 列目の各成分の積の和が，積 AB の i 行 j 列目の成分となる。

例えば，2 行 2 列の正方行列 A と B の積 AB は，次の手順で求められる。

(1) AB の 1 行 1 列目の成分 ＝ A の 1 行目の各成分と B の 1 列目の各成分の積の和

$$AB = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & \\ & \end{bmatrix}$$

(2) AB の 1 行 2 列目の成分 ＝ A の 1 行目の各成分と B の 2 列目の各成分の積の和

$$AB = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ & \end{bmatrix}$$

(3) AB の 2 行 1 列目の成分 ＝ A の 2 行目の各成分と B の 1 列目の各成分の積の和

$$AB = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & \end{bmatrix}$$

(4) AB の 2 行 2 列目の成分 ＝ A の 2 行目の各成分と B の 2 列目の各成分の積の和

$$AB = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

なお，行列の積 AB は，A の列数と B の行数が等しい場合のみ定義できる。また，A と B を入れ替えても積が定義できる場合について，AB ＝ BA は必ずしも成立しない。

さまざまな行列

行列には，次のような種類がある。

・単位行列

右下がりの対角線上の成分がすべて 1 で，それ以外の成分がすべて 0 の正方行列を「単位行列」といい，一般的には I や E で表す。単位行列 I は，任意の正方行列 A に対して AI ＝ IA ＝ A（乗算しても同じ行列になる）という性質をもつ。

2 行 2 列 の単位行列 $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$　　3 行 3 列 の単位行列 $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

・逆行列

ある正方行列と乗算すると単位行列になる正方行列を「逆行列」という。例えば，ある正方行列 A と B について，AB ＝ BA ＝ I という関係が成り立つ場合，B は A の逆行列（または A は B の逆行列）と言える。ただし，すべての正方行列に逆行列が必ず存在するとは限らない（逆行列が存在する正方行列を特に「正則行列」という）。

2 行 2 列の正方行列 A の逆行列 A^{-1} は，次のように公式化できる。

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \qquad AA^{-1} = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

・転置行列

ある行列を対角線で折り返した行列を「転置行列」という。例えば，3 行 4 列の行列 A の転置行列 A^T は，次のように i 行 j 列の成分 a_{ij} を j 行 i 列に移動したものとなる。

転置前の行列 $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$　　転置後の行列 $A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ a_{14} & a_{24} & a_{34} \end{bmatrix}$

基礎演習

問1　次に示す 2 行 3 列の行列 A と 3 行 2 列の行列 B の積 AB はどれか。

$$A = \begin{bmatrix} 2 & 5 & 3 \\ 4 & 1 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 3 \\ 4 & 1 \\ 5 & 2 \end{bmatrix}$$

解答群

ア $\begin{bmatrix} 4 & 20 & 15 \\ 12 & 6 & 2 \end{bmatrix}$

イ $\begin{bmatrix} 4 & 12 \\ 20 & 6 \\ 15 & 2 \end{bmatrix}$

ウ $\begin{bmatrix} 16 & 28 & 9 \\ 12 & 26 & 13 \\ 18 & 37 & 17 \end{bmatrix}$

エ $\begin{bmatrix} 39 & 17 \\ 37 & 20 \end{bmatrix}$

問2　次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

右下がりの対角線上の成分がすべて 1 で，それ以外の成分が 0 の正方行列（行数と列数が等しい行列）を単位行列という。また，ある行列と乗算すると単位行列になる行列を逆行列という。

関数 isInverse は，引数として与えられた二つの 3 行 3 列の正方行列について，一方の行列がもう一方の行列の逆行列であれば 1 を，そうでなければ 0 を返す。ここで，引数の指定に誤りはないものとする。

クラス Matrix は行列を表すクラスであり，メンバ変数として行列が格納された実数型の二次元配列 matVal をもつ。クラス Matrix の説明を図に示す。

コンストラクタ	説明
Matrix(実数型の二次元配列: qMatVal)	引数 qMatVal で メンバ変数 matVal を初期化する。

メソッド	戻り値	説明
mult(Matrix: multiplier)	Matrix	行列どうしを乗算した結果の行列を返す。戻り値のメンバ変数には，メンバ変数 matVal に，引数 multiplier がもつメンバ変数を乗算した結果の行列が格納される。
equals(Matrix: target)	整数型	メンバ変数 matVal と引数 tagrget がもつメンバ変数の内容を比較し，等しければ 1 を，等しくなければ 0 を返す。

〔プログラム〕

```
○整数型: isInverse(Matrix: matrixA, Matrix: matrixB)
  Matrix: matrixI ← Matrix({{1, 0, 0}, {0, 1, 0}, {0, 0, 1}})
  if ( [　　] )
    return 1
  else
    return 0
  endif
```

解答群

ア　matrixA.equals(matrixI.mult(matrixB)) が 0 と等しい
イ　matrixA.equals(matrixI.mult(matrixB)) が 1 と等しい
ウ　matrixB.equlas(matrixI.mult(matrixA)) が 0 と等しい
エ　matrixB.equlas(matrixI.mult(matrixA)) が 1 と等しい
オ　matrixI.equals(matrixA.mult(matrixB)) が 0 と等しい
カ　matrixI.equals(matrixA.mult(matrixB)) が 1 と等しい

問３　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，配列の要素番号は 1 から始まる。

関数 calc は，2 行 2 列の行列が格納された二つの整数型の二次元配列 matrixA と matrixB を引数として受け取り，演算結果を格納した整数型の二次元配列 result を戻り値として返す。

関数 calc を calc({{4, 1}, {5, 7}}, {{9, 6}, {7, 4}}) として呼び出すと，プログラム中の /*** α ***/ の行で［　　］が返される。

〔プログラム〕

```
○整数型の二次元配列: calc(整数型の二次元配列: matrixA,
                           整数型の二次元配列: matrixB)
  整数型: i, j, k
  整数型の二次元配列: result ← {{0, 0}, {0, 0}}
  for (i を 1 から 2 まで 1 ずつ増やす)
    for (j を 1 から 2 まで 1 ずつ増やす)
      for (k を 1 から 2 まで 1 ずつ増やす)
        result[i, j] ← result[i, j] + matrixA[i, k] × matrixB[k, j]
      endfor
    endfor
  endfor
  return result  /*** α ***/
```

解答群

ア　{{43, 28}, {94, 58}}

イ　{{43, 94}, {28, 58}}

ウ　{{66, 48}, {51, 35}}

エ　{{66, 51}, {48, 35}}

実践演習

問題　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，配列の要素番号は 1 から始まる。

関数 transpose は，引数 matrix で整数型の二次元配列として与えられた 3 行 3 列の行列の転置行列を返す。

関数 transpose を transpose({{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}) として呼び出すと，プログラム中の /*** α ***/ の行を 2 回実行した直後の配列 transMatrix の内容は，［　　］である。ここで，配列 transMatrix は，転置行列が格納される 3 行 3 列の整数型の二次元配列とする。

〔プログラム〕

```
○整数型の二次元配列: transpose(整数型の二次元配列: matrix)
  整数型の二次元配列: transMatrix ← matrix
  整数型: i, j
  for (i を 1 から 3 まで 1 ずつ増やす)
    for (j を 1 から i まで 1 ずつ増やす)
      transMatrix[i, j] と transMatrix[j, i] を交換する
    endfor
  endfor  /*** α ***/
  return transMatrix
```

解答群

ア　{{1, 2, 3}, {4, 5, 8}, {7, 6, 9}}
イ　{{1, 2, 7}, {4, 5, 8}, {3, 6, 9}}
ウ　{{1, 4, 3}, {2, 5, 6}, {7, 8, 9}}
エ　{{1, 4, 7}, {2, 5, 6}, {3, 8, 9}}
オ　{{1, 4, 7}, {2, 5, 8}, {3, 6, 9}}