「Java によるアルゴリズムとデータ構造の基礎」
演習問題解答

（2019 年 7 月 18 日現在）

1-1　（正解）イ。

1-2　（正解）イ。

1-3　（正解）イ。

1-4　（正解）ア。

1-5　（正解）ア。　6 行目の変数 i は for 文の外にあり，コンパイルエラーとなる。

1-6　（正解）エ。　拡張 for 文の用法を問う問題である。

2-1　（正解）ウ。「未知数の個数の 3 乗に比例する計算時間が掛かる」ので，100 元連立一次方程式の計算量を 100^3 とすると，1,000 元連立一次方程式では $1{,}000^3 = (100 \times 10)^3 = 100^3 \times 10^3$ の計算量となる。したがって，1,000 倍の計算時間を要するので，同一のコンピュータでは $2 \times 1{,}000 = 2{,}000$ 秒となる。しかし，4 倍の演算速度をもつコンピュータを用いるので，計算時間は 1／4 である 500 秒になる。

2-2　（正解）ウ。　$M = 2$ としてトレースしてみる。左図では $x = 2$ となる。右図では，ループ前の初期化処理で，$x = 1$, $n = 1$ となる。

まず，ループ 1 回目で，$x \times n = 1 \times 1 \rightarrow x = 1$，$n + 1 = 1 + 1 \rightarrow n = 2$ である。

ウ　正解。$n > M$ は $2 > 2$ となり，ループ 2 回目で $x \times n = 1 \times 2 \rightarrow x = 2$，$n + 1 = 2 + 1 \rightarrow n = 3$ となる。$n > M$ は，$3 > 2$ となりループを脱出する。このとき，$x = 2$ であり，左図の $x = 2$ と同一である。

2-3　（正解）イ。ド・モルガンの法則を利用して式を展開する。

ド・モルガンの法則は，$\overline{X \cdot Y} = \overline{X} + \overline{Y}$ または，$\overline{X + Y} = \overline{X} \cdot \overline{Y}$ である。

イ　正解。　$\overline{(\overline{X \cdot X}) \cdot (\overline{Y \cdot Y})} = \overline{(\overline{X}) \cdot (\overline{Y})} = X + Y$

2-4　（正解）エ。左の流れ図では，手続きは $(A > 0) \cdot (B > 0)$ の場合に実行される。右の流れ図は，（$(A > 0) \cdot (B > 0)$ の否定であるので，ド・モルガンの法則より $\overline{(A > 0) \cdot (B > 0)} = \overline{(A > 0)} + \overline{(B > 0)}$ が得られるので，エが正解。

2-5　（正解）エ。

2-6　（正解）ウ。

ア　$1 + 2 + \cdots + (N-1)$ なので，誤り。

イ　ループせず総和の計算になっていないので，誤り。

ウ　$1 + 2 + \cdots + N$ なので，正解。

エ　整数の総和の計算になっていないので，誤り。

2-7　（正解）エ。　このアルゴリズムは，素因数分解を用いて 10 進数から 2 進数に変換する方法を表したものである。

たとえば，$j = 2$ として 2 進数に変化すると，“10”が得られ，NISHIN[1] = 0, NISHIN[2] = 1 となるので，それぞれの処理をトレースする。

ア　j div 2 = 2 div 2 = 1→ j = 1，j mod 2 = 1 mod 2 = 1　→ NISHIN[1] = 1 となり，誤り。

イ　j div 2 = 2 div 2 = 1 → NISHIN[1]=1 となり，誤り。

ウ　j mod 2 = 2 mod 2 = 0 → j = 0，j div 2 = 0 div 2 = 0 →NISHIN[1] = 0 ；
j mod 2 = 0 mod 2 = 0 → j = 0，j div 2 = 0 div 2 = 0 →NISHIN[2] = 0 となり，誤り。

エ　j mod 2 = 2 mod 2 = 0 → NISHIN[1] = 0 ; j div 2 = 2 div 2 = 1 → j = 1 ;
j mod 2 = 1 mod 2 = 1 → NISHIN[2] =1 ; j div 2 = 1 div 2 = 0　・・・
となり正解。

3-1　（正解）ア。　長さ m の配列 X の後に，長さ n の配列 Y を格納した配列 Z を作成するアルゴリズムである。ループ 1 では，　$X[k] \rightarrow Z[k]$ とし，ループ 2 のインデックス k の初期値が 1 であることに着目すると，$Y[k] \rightarrow Z[m+k]$ である。したがって，アが正解。

3-2　（正解）イ。　「隣同士 $T(j)$ と $T(j-1)$ 交換」の処理があるので，「バブルソート」であることがわかる。昇順に整列するので，「$T(j) < T(j-1)$」の場合に交換が必要である。したがって，イが正解。

3-3　（正解）エ。　実行前後の図を比べると，処理内容は配列のデータの位置を右に 90° 回転していることがわかる。 したがって，i 方向の位置は j 方向で“i ”になり，j 方向の位置は i 方向の“7 － j ”になる。したがって，正解は“$B(j, 7 - i) \leftarrow A(i, j)$”である。したがって，エが正解。

4-1　（正解）イ。　$F(231, 15) = F(15, 231 \bmod 15) = F(15, 6) = F(6, 15 \bmod 6) = F(6, 3) = F(3, 6 \bmod 3) = F(3, 0) = 3$

したがって，イが正解。

4-2　（正解）エ。　$f(4, 2) = f(3, 1) + f(3, 2) = 1 + 3 = 6$

ここで，$f(3, 1) = f(2, 0) + f(2, 1) = 1 + f(1, 0) + f(1, 1) = 1 + 1 + 1 = 3$

$f(3, 2) = f(2, 1) + f(2, 2) = f(1, 0) + f(1, 1) + 1 = 1 + 1 + 1 = 3$

4-3　（正解）イ。　$F(5) = 5 \times G(4)$ ，$G(4) = 4 + F(3)$，$F(3) = 3 \times G(2)$，$G(2) = 2 + F(1)$，$F(1) = 1$ である。 したがって，上式を逆にたどり $G(2) = 3$，$F(3) = 9$, $G(4) = 4 + 9 = 13$，$F(5) = 5 \times 13 = 65$ であるので，イが正解。

4-4　（正解）イ。　再帰とは，実行中に自分自身を呼び出すことであり，再帰呼出しを行っても正しい結果を返すことができる性質をもつプログラムを「再帰的プログラム」と呼ぶ。

4-5　（正解）エ。　分割統治法の説明であるので，エが正解。

4-6　（正解）ウ。　n の階乗は，$n! = n \times (n-1)!$であるから，ウが正解。

5-1　（正解）ウ。　変更前　東京→品川→名古屋→新大阪，変更後　東京→新横浜→名古屋→新大阪

「東京」から「新横浜」，「新横浜」から「名古屋」へのリンクをはればよい。すなわち，A(1, 2)=5 で A(5, 2)=3 とすれば良いので，ウが正解。

5-2　（正解）イ。　変更前　東京→新横浜→熱海→浜松→名古屋，変更後　東京→新横浜→熱海→静岡→浜松→名古屋

「熱海」のポインタを 150 に，「静岡」のポインタを 70 とすればよい。したがって，イが正解。

5-3　（正解）エ。

5-4　（正解）ウ。　初期データは，A→E→C→G→B である。

3 番目（C）と 4 番目（G）との間に値が H を挿入するので，next[3] = 8 として，next[8] = 7 であればよい。したがって，ウが正解。

5-5　（正解）ア。

イ　先頭へのデータの挿入は，配列ではデータの移動が必要であるため，誤り。

ウ　配列では削除や挿入の操作を効率的に行えないので，誤り。

エ　データの移動は，配列ではデータの移動が必要であるので，誤り。

5-6　（正解）ア。

イ　配列では，挿入や削除はデータの移動を伴うので，処理時間は要素数に依存するので，誤り。

ウ　配列では要素をメモリ上の連続領域に格納するので，中間要素のインデックスはデータサイズから計算でき，参照時間は一定になるので，誤り。

エ　ポインタで実現するリストの特徴の説明であるので，誤り。

5-7　（正解）ウ。　A→K→T を A→G→K→T に変更することを考える。A の次ポインタ（a）を 400 に，G の次ポインタ（x）を 300 する。また，G の前ポインタ（y）を 100 に，K の前ポインタ（f）を 400 にする必要がある。

X と y は対象外であるので，ウが正解。

5-8　（正解）エ。

ア　挿入するデータの次ノードへの参照に現在の先頭ポインタの値をセット，リストの先頭ポインタに挿入するデータのポインタをセット。ポインタを参照する回数は 1 回。

イ　現在の先頭のデータが持つ次ノードへの参照を，先頭ポインタにセット。先頭ポインタ→先頭データで 1 回。

ウ　末尾ポインタから末尾データへ飛び，末尾データの次ノードへの参照を追加するデータにする。末尾ポインタに追加するデータの参照をセットする。末尾ポインタ→末尾データで 1 回。

エ　正解。末尾データを削除には，末尾の 1 つ前のノードがもつ次ノードへの参照を空にする。末尾ポインタに末尾の一つ前のデータのポインタをセットする。単方向リストなので末尾のデータから逆方向の参照はできず，先頭から末尾の一つ前まで順番にポインタをたどっていく必要があるので，参照回数が多くなる。

6-1　（正解）ア。　キューは，データを先入れ先出し（FIFO）で蓄える。

6-2　（正解）ア。　スタックは，データを後入れ先出し（LIFO）で蓄える。

6-3　（正解）イ。　実行された 10 命令を一つずつ戻りながら，最初の PUSH 命令までさかのぼっていく。

1. PUSH　スタックの一番上にあるデータ「192」を取り除く。

2. POP　取り出されたデータ「?」(値が不明)をスタックに積む。
3. POP　取り出されたデータ「?」をスタックに積む。
4. PUSH　スタックの一番上にあるデータ「?」を取り除く。
5. PUSH　スタックの一番上にあるデータ「?」を取り除く。
6. PUSH　スタックの一番上にあるデータ「55」を取り除く。
7. PUSH　スタックの一番上にあるデータ「326」を取り除く。
8. POP　取り出されたデータ「?」をスタックに積む。
9. PUSH　スタックの一番上にあるデータ「?」を取り除く。
10. これが最初の PUSH 命令である。スタックの一番上にあるデータは「7」なので，1番目の命令では「7」が PUSH されたことがわかる。したがって，正解はイ。

6-4　（正解）ウ。　スタックの状態の推移は，以下のようになる。

PUSH 1 ｛1｝→ PUSH 5 ｛5,1｝→ POP｛1｝ → PUSH 7｛7,1｝ → PUSH 6｛6,7,1｝→ PUSH 4 ｛4,6,7,1｝→ POP｛6,7,1｝ → POP｛7,1｝ → PUSH 3 ｛3,7,1｝ したがって，ウが正解。

6-5　（正解）ウ。

6-6　（正解）ウ。　キューの状態の推移は，以下のようになる。

ENQ1｛1｝→ENQ2｛2,1｝→ENQ3｛3,2,1｝→DEQ｛3,2｝→ENQ4｛4,3,2｝→ENQ5｛5,4,3,2｝→DEQ｛5,4,3｝→ENQ6｛6,5,4,3｝→DEQ｛6,5,4｝→DEQ｛6,5｝

したがって，次の DEQ の操作で取り出される値は 5 である。正解はウ。

6-7　（正解）イ。　変数 *p* は現在の配列の要素数を保持している変数である。*f*(*x*)は配列の最後に引数である *x* を代入する関数であり，*g*( )は配列の最後の要素を返す関数と考えることができる。「配列の最後にデータを追加する」および「最後に追加されたデータを取り出す」という 2 つの操作を合わせると LIFO（後入れ先出し）のデータ構造が実現される。したがって，イが正解。

6-8　（正解）イ。

6-9　（正解）ウ。

6-10　（正解）ウ。　スタックの状態の推移は，以下のようになる。

ア　push(A)｛A｝→pop( ){}→ push(B)｛B｝→ push(C)｛C,B｝→push(D)｛D,C,B｝→ pop( )｛C,B｝となるので，B の取り出しはできないので，誤り。

イ　push(A)｛A｝→push(B)｛B,A｝→pop( ){A}→push(C)｛C,A｝→push(D)｛D,C,A｝→pop( ){ C,A }となり，A の取り出しはできないので，誤り。

ウ　push(A)　｛　A　｝　→push(B)　｛　B,A　｝　→push(C)　｛　C,B,A　｝　→pop( ){ B,A }→pop( ){ A }→push(D) { D, A }
→pop( ){ A}→pop( ) { } となり，ウが正解。

エ　push(A)｛A｝→push(B)｛B,A｝→ push(C)｛C,B,A｝→push(D)｛D,C,B,A｝→ pop( )｛C,B,A｝→ pop( )｛B,A｝となり，A の取り出しはできないので，誤り。

7-1　（正解）ウ。

7-2　（正解）ア。

7-3　（正解）ア。

7-4　（正解）ア。

前順走査なので，節点 → 左部分木 → 右部分木の順に走査する。

7-5　（正解）ウ。

7-6　（正解）ウ。　与えられたプログラムを設問の 2 分木に適用すると次のようになる。最初に 2 分木の根である"+"から開始。

・[proc(+)]　"+"の左には"a"があるので，Proc(a)が呼び出される。
・[Proc(a)]　"a"の左右にはノードがないので，a を出力し，Proc(+)に戻る。
・[proc(+)]　"+"の右には"*"があるので，Proc(*)が呼び出される。
・[Proc(*)]　"*"の左には"-"があるので，Proc(-)が呼び出される。
・[Proc(-)]　"-"の左には"b"があるので，Proc(b)が呼び出される。
・[Proc(b)]　"b"の左右にはノードがないので，b を出力し，Proc(-)に戻る。
・[Proc(-)]　"-"の右には"c"があるので，Proc(c)が呼び出される。
・[Proc(c)]　"c"の左右にはノードがないので，c を出力し，Proc(-)の処理に戻り，-を出力し，Proc(*)に戻る。
・[Proc(*)]　"*"の右には"d"があるので，Proc(d)が呼び出される。
・[Proc(d)]　"d"の左右にはノードがないので，d を出力し，Proc(*)の処理に戻り，*を出力し，Proc(+)に戻る。
・Proc(+)は，+を出力し，処理が終了。

出力された記号を順番に並べると「*abc*－*d**＋」となり，ウが正解。

7-7　（正解）ウ。　この木構造において深さ(k)が増加すると節点数(n)は次のように変化する。$k=1\rightarrow n=3$　　$k=2\rightarrow n=7$　　$k=3\rightarrow n=15$

ここでは選択肢の式に $k=3$ を代入して，n が 15 になるか否かで判定する。

ア　$n=k(k+1)+1=3(3+1)+1=12+1=13$ で，誤り。

イ　$n=2^k+3=2^3+3=8+3=11$ で，誤り。

ウ　正解。$n=2^{k+1}-1=2^4-1=16-1=15$

エ　$n=(k-1)(k+1)+4=(3-1)(3+1)=8$ で，誤り。

8-1　（正解）ア。　時間計算量は，「2 分探索」は O(log n) ，「線形探索」は O(n)，ハッシュ探索は O(1)である。したがって，アが正解。

8-2　（正解）イ。　2 分探索において，N個の要素を探索する場合，最大比較回数は，「$\log_2 N+1$」である。データの個数が 4 倍になると「 $\log_2 4N+1$ 」となり，$\log_2 4N=\log_2 2^2+\log_2 N=2+\log_2 N$であるから，2 回増えることになる。したがって，イが正解。

8-3　（正解）イ。　2 分探索法はソート済（昇順または降順）のデータに対する探索法である。したがって，イが正解。

8-4　（正解）ウ。　2 分探索法はソート済（昇順または降順）のデータに対する探索法である。したがって，ウが正解。

8-5　（正解）ウ。　2 分探索法は，整列されたデータの中央の値と対象データを比較し，前方にデータがある場合は，前半分のデータの中央のデータと比較し，後方にデータがある場合は，後半分のデータの中央のデータと比較する。この操作を繰り返すことによって，検索する方法である。

A(k) と x を比較し，A(k)<x の場合は，　k+1 を lo に入れる。A(k)>x の場合は，$k-1$ を hi に入れる。したがって，ウが正解。

8-6　（正解）ア。　2 分探索法は，O(log n)である。したがって，アが正解。

8-7　（正解）ウ。　9 個のデータを 10 進数に変換し，ハッシュ値を求める。

・1A)h =26)$_{10}$ より，mod(26, 8) = 2　・35)h =53)$_{10}$ より，mod(53, 8) = 5

・3B)h =59)$_{10}$ より，mod(59, 8) = 3　・54)h =84)$_{10}$ より，mod(84, 8) = 4

・8E)h =142)$_{10}$ より，mod(142, 8) = 6

・A1)h =161)$_{10}$ より，mod(161, 8) = 1

・AF)h =175)$_{10}$ より，mod(175, 8) = 7

・B2)h =178)$_{10}$ より，mod(178, 8) = 2・・・衝突発生

したがって，ウが正解。

8-8　（正解）ウ。

8-9　（正解）ア。　$55550)_{11} = 5\times11^4+5\times11^3+5\times11^2+5\times11^1 = =(11^3+11^2+11+1)\times55 = 80520$

80520 の下 4 けた「0520」に 0.5 を乗じると参照すべきアドレス 0260 が求められる。したがって，アが正解。

8-10　（正解）イ。

表に従って，キー“SEP”を 1 文字ずつ数値に割り当てる。 “S”= 19, “E”= 5, “P” = 16 であるので， 19 ＋ 5 ＋ 16 ＝ 40，　したがって，ハッシュ値 h ＝ 40 ÷ 27 ＝ 1 余り 13。

ア　APR→ 1＋ 16＋ 18 ＝35　ハッシュ値 h ＝ 35 ÷ 27 ＝ 1 余り 8

イ　FEB→ 6＋ 5＋ 2 ＝13　ハッシュ値 h ＝ 13 ÷ 27 ＝ 0 余り 13

ウ　JAN→ 10＋ 1＋ 14 ＝25　ハッシュ値 h ＝ 25 ÷ 27 ＝ 0 余り 25

エ　NOV→ 14＋ 15＋ 22 ＝51　ハッシュ値 h ＝ 51 ÷ 27 ＝ 1 余り 24

したがって，イが正解。

8-11　（正解）イ。　ハッシュ関数に"54321"を当てはめると，mod(5＋4＋3＋2＋1, 13)＝mode(15,13) = 2 となる。

したがって，イが正解。

8-12　（正解）ウ。　ハッシュ値は，「キー値をハッシュ表の大きさ 10 で割った余り」なので，キー値の 1 の位がそのままハッシュ値になる。

まず，1 件目のレコードがハッシュ表の任意の場所に格納される。

次に，2 件目のレコードのハッシュ値が 1 件目のものと衝突しないためには，キー値の 1 の位が異なっていればよいので，衝突しない確率は 9／10 になる。

最後の 3 件目のレコードが格納されるときには，衝突が起こらないためには 1，2 件目両方のキー値の 1 の位が異なる必要があるため，衝突しない確率は 8／10 になる。

したがって，3 件のレコードすべてが衝突しない確率は，「1，2 件目が衝突しない確率」と「1，2 件目と 3 件目が衝突しない確率」の積なので，

(9／10)×(8／10)＝0.72 となる，したがって，ウが正解。

9-1　（正解）ア。　バブルソートのアルゴリズムである。行 2～3 の処理が初めて終了したとき，A[1] が最小値になる。したがって，アが正解。

9-2　（正解）ウ。

ア　クイックソートは，基準の値を決めて，基準より小さい値を前に，基準より大きい値を後ろに入れ替える方法であるので，誤り。

イ　選択ソートは，データの中で最も小さいものを探して交換する方法であるので，以下のようになる。したがって，誤り。

(4, 1, 3, 2)
(1, 4, 3, 2)　　最小値 1 と 4 を交換
(1, 2, 3, 4)　　最小値 2 と 4 を交換

ウ　挿入ソートは，データの整列部分の適切な位置に未整列の部分の先頭の値を挿入していく方法であるので，以下のようになる。したがって，正解。

(4, 1, 3, 2)
(1, 4, 3, 2)　　1 を 4 の前に挿入
(1, 3, 4, 2)　　3 を 1 と 4 の間に挿入
(1, 2, 3, 4)　　2 を 1 と 3 の間に挿入

エ　バブルソートは，隣り合うデータを比較し，順番が逆の場合は交換する方法であるので，以下のようになる。したがって，誤り。

(4, 1, 3, 2)
(1, 4, 3, 2)　　4 と 1 を交換
(1, 3, 4, 2)　　4 を 3 を交換
(1, 3, 2, 4)　　4 を 2 を交換
(1, 2, 3, 4)　　3 を 2 を交換

9-3　（正解）エ。

9-4　（正解）ウ。　隣り合うデータを比較し，必要であれば交換を行うという一連の操作を(n - 1)回行う。全体の比較回数は，(n - 1) + (n - 2) + ・・・+ 1 = n (n - 1) / 2 となる。したがって，ウが正解。

10-1　（正解）エ。

図の整列過程を見ると，大きさが 1 の部分文字列の併合を繰り返して最終的な整列済み文字列にしていることがわかる。このように整列対象を大きさ 1 の部分文字列に分割した後，隣り合う要素ごとに整列と併合を繰り返しながら整列を行う手法がマージソートである。したがって，エが正解。

10-2　（正解）エ。

10-3　（正解）イ。

外側ループ：1 から n－1 まで反復，内側ループは，i+1 から n まで反復。

i が 1 の時，内側ループは，j ＝ 2, 3, 4, ..., n－1, n まで反復（ n － 1 回）。

i が 2 の時，内側ループは，j ＝ 3, 4, ..., n－1, n まで反復（ n － 2 回）。

i が 3 の時，内側ループは，j ＝ 4, ..., n－1, n まで反復（ n － 3 回）。

・・・

i が n－1 の時，内側ループは，j ＝ n （1 回）。

したがって，「＊印の処理（比較）」は，

1 ＋ 2 ＋ 3 ＋ ... ＋ n－1 ＝ n（ n－1）／2 となり，イが正解。

10-4　（正解）ア。

手順に沿って整列を行うと次のようになる。

(1) H←9÷3＝3

(2) 要素ごとがたがいに 3 つずつ離れた要素からなる 3 つの部分文字列に分解し，それぞれ整列を行う。

［部分列 1］7, 3, 4　→ 整列 → 3, 4, 7
［部分列 2］2, 1, 5　→ 整列 → 1, 2, 5
［部分列 3］8, 9, 6　→ 整列 → 6, 8, 9

(3) H←3÷3＝1　　・・・　1 回目

(4) H≠0 なので 2 に戻る。

(5) 要素ごとがたがいに 1 つずつ離れた要素からなる 3,4,7,1,2,5,6,8,9 を整列し，1,2,3,4,5,6,7,8,9 とする。

(6) H←1÷3＝0（小数点未満切り捨て）　　・・・　2 回目

この時点で H が 0 になるためデータ列の整列は完了する。

したがって(3)の処理が繰り返される回数は 2 回となるので，アが正解。

10-5　（正解）ア。　このアルゴリズムは，データ群を，枢軸（pivot）より大きいデータ群と小さいデータ群とに分割し，それぞれのデータ群に対して要素が一つになるまで，同じ操作を再帰的に適用するものである。

10-6　（正解）ア。

10-7　（正解）ウ。