Ｃｈａｐｔｅｒ16　プログラムの作り方

16-1　プログラム言語とは〔解答・解説〕

問１　エ

問２　イ
〔解説〕ア，ウはＷｅｂブラウザが，エはＷｅｂサーバが必要となる。

問３　ウ
〔解説〕ア　ＪａｖａＳｃｒｉｐｔの説明
イ　Ｊａｖａサーブレットは、サーバに常駐して稼働し、ＣＧＩは呼出しごとにプロセスの生成と解放を繰り返す
エ　ＪａｖａＲＭＩの説明

問４　ア
〔解説〕イ　動的なＷｅｂページの構築のために、ＨＴＭＬ内に記述されるスクリプト言語
ウ　ローカルコンピュータ上のＪａｖａ　Ｒｕｎｔｉｍｅ　Ｅｎｖｉｒｏｎｍｅｎｔ（ＪＲＥ）で実行されるプログラムを指す
エ　Ｗｅｂを通してブラウザにダウンロードされクライアント側で実行されるＪａｖａで書かれたプログラムのこと

問５　エ
〔解説〕Ｐｅｒｌ(パール)は、テキスト処理用のプログラム言語。インタプリタ言語であり、ＵＮＩＸやＷｉｎｄｏｗｓなど多くのプラットフォーム上で動作する。以前はＷｅｂページのＣＧＩの記述によく用いられていた。
ア　Ｗｉｎｄｏｗｓ環境用の言語処理系もある
イ　デスクトップアプリケーションも作成できる
ウ　付属のＰｅｒｌデバッガでデバッグ可能

問６　エ
〔解説〕ＪａｖａＳｃｒｉｐｔはＨＴＭＬに組み込まれ、Ｗｅｂページに動きを加えたりユーザからの入力のタイミングで任意の処理を行わせることができます。

問７　イ

問８　ウ
〔解説〕ア　構文解析のフェーズで行う処理
イ　字句解析のフェーズで行う処理
エ　最適化のフェーズで行う処理

問 9　ア
〔解説〕＜S＞::＝　0 1　又は　0＜S＞1

↓

0　0 1　1

↓

0　0 0 1 1　1

↓

…

と定義され，0と1が同数個並んだものとなる。

問 10　ウ
〔解説〕＜式＞::＝＜変数＞｜(＜式＞＋＜式＞)｜＜式＞＊＜式＞より，＋記号を使う場合は必ず()で囲む必要がある。
ア　(A＋(B＋C)＊D)でなければならない
イ　((A＋B)＋(C＋D))でなければならない
エ　(A＊B＋C＊D)でなければならない

問 11　イ
〔解説〕ア　＜符号＞＜数字列＞は定義されていない
ウ　＜符号＞＜数字列＞E＜符号＞＜数字列＞は定義されていない
エ　＜符号＞＜数字列＞E＜数字列＞は定義されていない

問 12　ア
〔解説〕各肢を展開していき<パラメタ指定>に帰結するか否かを調べていく。
※英字の並びは<パラメタ>で表される
ア　　((<パラメタ>，<パラメタ>)，<パラメタ>)
→　((<パラメタ指定>，<パラメタ>)，<パラメタ>)
→　(<パラメタ指定>，<パラメタ>)
→　<パラメタ指定>
最終的に<パラメタ指定>になるのでこれが正解。

イ　　((<パラメタ>，<パラメタ>))
→　((<パラメタ指定>，<パラメタ>))
→　(<パラメタ指定>)
外側の括弧を外すことができないので不適切。

ウ　(<パラメタ>，(<パラメタ>))
これ以上変形できないので不適切。

エ　　(<パラメタ>)
→　(<パラメタ指定>)
括弧を外すことができないので不適切。

問 13　エ
〔解説〕ア　先頭文字が"__"なので＜変数名＞には合致しない。
イ　＜変数名＞に置き換えられる部分はない。
ウ　先頭文字が＜数字＞なので＜変数名＞には合致しない。

16-2 コンパイラ方式でのプログラム実行手順〔解答・解説〕

問 1　ア

問 2　エ
〔解説〕字句解析：
　プログラムを表現する文字の列を、意味のある最小の構成要素の列に変換する
構文解析：
　言語の文法に基づいてプログラムを解析し、文法誤りがないかチェックする
意味解析：
　変数の宣言と使用とを対応付けたり、演算におけるデータ型の整合性をチェックする
最適化：
　ジスタの有効利用を目的としたレジスタ割付けや、不要な演算を省略するためのプログラム変換を行う

問 3　ウ
〔解説〕コンパイラは、ソースコードを翻訳して、機械語の目的プログラム(実行ファイル／ロードモジュール)を生成する言語プロセッサ。実行ファイルを生成する前のソースコードを機械語に翻訳する過程において、ソースコードの解析が行われ、効率的かつ、実行時間やメモリ使用量などを最少化するようにソースコードを調整する最適化処理が行われる。

問 4　ア

問 5　イ
〔解説〕動的リンクライブラリは，ライブラリ（関数やデータの集まり）のリンクをコンパイル時ではなく実行時に行うことをいう。

問 6　ウ
〔解説〕ア　コンパイラの機能
イ　ローダの機能
エ　デバッガの機能

問 7　イ
〔解説〕ア　エミュレータは、異なるアーキテクチャのコンピュータ用のプログラム命令を解読しながら実行するマイクロプログラムです。
ウ　最適化コンパイラは、通常のコンパイラよりも、最適化作業に優れ、処理をさらに高速に行うことができる形式にコンパイルすることができるソフトウェアです。
エ　ジェネレータは、ソースコードを記述しなくても、処理条件の入力・処理・出力・引数などのパラメータを指定することで自動的にプログラムを生成する言語プロセッサです。

16-3 構造化プログラミング〔解答・解説〕

問 1　ウ

問 2　エ
〔解説〕選択肢の制御構造は、繰り返し処理を行う「イ」「エ」、分岐処理を行う「ア」「エ」に分類できる。
さらに条件判定を処理の後に行っている「イ」がｄｏ－ｗｈｉｌｅ型、「エ」がｗｈｉｌｅ型となるため正解は「エ」になる。

問 3　エ
〔解説〕ア　繰返し処理の最後で判定を行う
イ　２つの処理のどちらかを選択する
ウ　複数の処理のどれか１つを選択するだけで，並列処理は行わない

16－4　変数は入れ物として使う箱〔解答・解説〕

問 1　エ
〔解説〕平均値を求める式は，Ｓ／Ｎ
四捨五入するためには，Ｓ／Ｎ＋０.５
整数値で求めるから，［Ｓ／Ｎ＋０.５］

問 2　ウ
〔解説〕この再帰関数は拡張されたユークリッドの互除法を用いて整数m，ｎの最大公約数(Greatest Common Divisor:gcd)を求めるもの、

```
ｇｃｄ(１３５，３５)//ｎ>０
ｇｃｄ(３５，１３５ ｍｏｄ ３５)　　//ｎ=３０
ｇｃｄ(３０，３５ ｍｏｄ ３０)　　//ｎ=５
ｇｃｄ(５，３０ ｍｏｄ ５)=３０　　//ｎ=０
```

４回の呼び出しで最大公約数である５を得られることがわかる。

問 3　ア
〔解説〕call by value ともいう。

問 4　イ
〔解説〕ア　実引数の値は変わらない
ウ　実引数は変数でも定数でもよいが，仮引数は変数だけである
エ　仮引数は呼び出される関数の中だけで有効であるが，実引数は関数の呼出し側でも有効である

16－5　アルゴリズムとフローチャート〔解答・解説〕

問 1　ア
〔解説〕総和を求めるための変数はゼロクリアしておく。

問 2　イ

〔解説〕ａとｂに適用な値を代入してアルゴリズムをトレースし、結果を選択肢の記述を比較することで答えを導きく。ここではａ＝２０，ｂ＝８とする。

```
x ← ２０
y ← ８
//ループ開始
ｔ ← ｍｏｄ(２０，８) //ｔ＝４
x ← ８
y ← ４
ｔ ← ｍｏｄ(８，４) //ｔ＝０
x ← ４
y ← ０
//ループ終了
//処理終了
x＝４
```

ａ＝２０、ｂ＝８なので、結果ｘはａとｂの最大公約数になっている。

問 3　イ

〔解説〕流れ図の結果を出すと以下の４通りとなる

Ａ が真、Ｂ が真＝＝＞結果は ｅｘｉｔ２
Ａ が真、Ｂ が偽＝＝＞結果は ｅｘｉｔ１
Ａ が偽、Ｂ が真＝＝＞結果は ｅｘｉｔ１
Ａ が偽、Ｂ が偽＝＝＞結果は ｅｘｉｔ２

ア　Ａ ＡＮＤ Ｂ については、Ａ が真、Ｂ が真＝＝＞結果は ｅｘｉｔ１より矛盾する。
イ　Ａ ＸＯＲ Ｂ については、成立
ウ　Ａ ＮＡＮＤ Ｂ については、Ａ が偽、Ｂ が偽＝＝＞結果は ｅｘｉｔ１より矛盾する。
エ　Ａ ＯＲ Ｂ については、Ａ が真、Ｂ が真＝＝＞結果は ｅｘｉｔ１より矛盾する。

問 4　ア

〔解説〕Ｘが１１０１，Ｙが１０１１の場合，乗算は次のように行われる。

（１回目）
```
   １１０１
× １０１１ ← Ｙ（最下位ビットが１なので加算）
   １１０１ ← Ｘ
```

（２回目）
```
   １１０１
×   １０１ ← Ｙを１ビット右シフト（最下位ビットが１なので加算）
   １１０１
 １１０１０ ← Ｘを１ビット左シフトしたもの
```

（３回目）
```
   １１０１
×     １０ ← Ｙを２ビット右シフト（最下位ビットが０なので加算せず）
   １１０１
 １１０１０
```

（４回目）
```
   １１０１
×        １ ← Ｙを３ビット右シフト（最下位ビットが１なので加算）
   １１０１
 １１０１０
１１０１０００ ← Ｘを３ビット左シフトしたもの
```

ａ：Ｙの最下位ビットが１の場合だけ加算を行うための比較であり，Ｙの最下位ビットとなる。
ｂ：１ビット処理が終わるたびに，Ｘは１ビット左シフト，Ｙは１ビット右シフトされる。

問５　ウ

〔解説〕この流れ図の処理は、ｘが９０より大きくなるまでｘを倍増させた後、ｘから９０を引いて終了というもの。ｘを倍増する回数は、０倍、２倍、４倍、８倍、１６倍、３２倍、６４倍、１２８倍が考えられる。それぞれの場合について、方程式を立てて以下のように解いてみる。

［ｘを０倍］

ｘ － ９０ ＝ ｘは成り立ちようがない。

［ｘを２倍］

２ｘ － ９０ ＝ ｘ

ｘ ＝ ９０

［ｘを４倍］

４ｘ － ９０ ＝ ｘ

３ｘ ＝ ９０

ｘ ＝ ３０

［ｘを１６倍］

１６ｘ － ９０ ＝ ｘ

１５ｘ ＝ ９０

ｘ ＝ ６

※８倍、３２倍、６４倍、１２８倍は小数になる

問 6　ウ

〔解説〕例えばＭに 2 を代入する。解き方としてはまず左のアルゴリズムを解き、その結果の x の値が右の式でも出力される条件はどれかを考えていくことになる。

[左の流れ図]

<<開始>>

1 → x　　//x=1

ループ条件 n:2 増分 -1、n=1 で終了

x×n → x (1×2 →x)　　//x=2

ループ先頭に戻る n－1 → n　　//n=1

n=1 になったのでループ終了

<<終了>>　　//x=2

左の流れ図をトレースしてみると、結果 x の値は 2 になることがわかる。

[右の流れ図]

<<開始>>

1→x　　//x=1

1 → n　　//n=1

x×n → x(1 × 1 →x)　　//x=1

n＋1 → n (1＋1 →n)　　//n=2

ドの条件でもここまでの工程は同じである。ここからは各選択肢ごとにみて行く。

ア　(分岐条件 n＜M)

n＜M (2＜2) は No →ループ先頭へ戻る。以後ループ内で n の値は加算されていくだけなので (n ＜ M)の条件を満たすことは永遠になく無限ループとなってしまう。

イ　(分岐条件 n ＞ M -1)

n＞M－1 (2＞2－1) は Yes →流れ図は終了する。 結果 x の値は 1 で、左の流れ図とは異なる値となる。

ウ　(分岐条件 n＞M)

n＞M (2＞2) は No →ループ先頭へ戻る。　　//x=1

x×n → x (1×2 →x)　//x=2

n＋1 → n (2＋1 →n)　//n=3

n＞M (3＞2) は Yes →流れ図は終了する。　　//x=2

結果 x の値は 2 で、**左の流れ図と同じ値**になる。

エ　(分岐条件 n＞M＋1)

n＞M＋1 (2＞2＋1) は No →ループ先頭へ戻る。　　//x=1

x×n → x (1×2 →x)　//x=2

n＋1 → n (2＋1 →n)　//n=3

n＞M＋1 (3＞2+1) は No →ループ先頭へ戻る。　　//x=2

x×n → x (2×3 →x)　//x=6

n＋1 → n (3＋1 →n)　//n=4

n＞M＋1 (4＞2＋1) は Yes →流れ図は終了する。　　//x=6

結果 x の値は 6 で、左の流れ図とは異なる結果となる。

16-6 データの持ち方〔解答・解説〕

問 1　ウ

〔解説〕リスト［東京，品川，名古屋，新大阪］を［東京，新横浜，名古屋，新大阪］に変化させる操作なの

（1）　新横浜のポインタ部 A（5，2） に名古屋（3）に設定する。

東京のポインタ部 A（1，2） の指している A（2，2） の値（3）を 新横浜のポインタ部 A（5，2） に 入れる。

3　→　A（5，2）

（2）　東京のポインタ部を新横浜（5） を設定する。

5　→　A（1，2）

問 2　エ

〔解説〕末尾データの削除は，末尾ポインタを用いて末尾データを削除した後，新しく末尾データとなる一つ手前のデータまで先頭ポインタからたどらなければならない。

ア　先頭へのデータの追加は先頭ポインタ操作と追加するデータのポインタ操作で行える

イ　先頭のデータの削除は先頭ポインタの操作だけで行える。

ウ　末尾へのデータの追加は末尾データのポインタ操作と末尾ポインタ操作で行える

問 3　ア

〔解説〕イ　配列を使ったリストに要素を途中に挿入する場合，後ろの要素を順にずらす必要があるので，挿入場所によって処理時間は異なる

ウ　配列を使ったリストの要素は要素番号で参照することができるので，参照時間は一定になる

エ　配列を使ったリストは格納領域に順に要素を格納していくので，次の要素を指し示すための領域（ポインタ領域）は必要ない

問 4　ウ

問 5　ウ

〔解説〕再帰呼出し時やサブルーチン呼出し時の戻り番地の格納には，ＬＩＦＯ構造のスタックが用いられる。

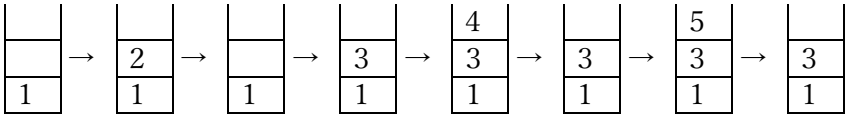
問 6　ア

問 7　ウ
〔解説〕スタックを 3 つ用意すると、CK よりも先に出力したい A を退避させておくことができる。

push A：［A］［ ］［ ］
push C：［A］［C］［ ］
push K：［A］［C］［K］
push S：［A］［C］［K，S］
pop　：［A］［C］［K］→ S を出力
push T：［A］［C］［K，T］
pop　：［A］［C］［K］→ T を出力
pop　：［ ］［C］［K］→ A を出力
pop　：［ ］［ ］［K］→ C を出力
pop　：［ ］［ ］［ ］→ K を出力

したがって、S，T，A，C，K という順に文字を出力するためには、最低限スタックが 3 個必要。

問 8　ア
〔解説〕スタック領域は，次のように変化していく。



問 9　イ
〔解説〕アは b， d， c， a，ウは b， c， d， a，エは b， a， d， c の順番に出力される

問 10　ウ
〔解説〕〔データの追加〕について、最後尾の要素へのアクセスするための手間を考える。最後尾ポインタのある「ウ」「エ」では最後尾の要素へ1回でアクセスできるが、最後尾ポインタがない「ア」「イ」では最後尾の要素を参照するために先頭から順番にリストを辿っていく必要があり、リスト内の要素数に比例して手間が掛かる。したがって「ウ」「エ」に実装に優位性がある。

「ウ」と「エ」は先頭・最後尾要素へのアクセスの手間は同じだが、「エ」のリストが循環型になっているためポインタ付け替え回数に違いがある。「ウ」の実装では上記したポインタの付け替えだけで済むが、「エ」ではそれに加えて以下の処理が発生する。

・データの追加時
　追加要素の次要素へのポインタに先頭要素のアドレスを設定する
・データの取出し
　最後尾の要素の次要素へのポインタを先頭ポインタと同じに設定する

問 11　ウ
〔解説〕それぞれのパターンを検証する
［A→B→C］
　ｐｕｓｈ（A）→ｐｏｐ→ｐｕｓｈ（B）→ｐｏｐ→ｐｕｓｈ（C）→ｐｏｐ の順序で出力可
［A→C→B］
　ｐｕｓｈ（A）→ｐｏｐ→ｐｕｓｈ（B）→ｐｕｓｈ（C）→ｐｏｐ→ｐｏｐの順序で出力可
［B→A→C］
　ｐｕｓｈ（A）→ｐｕｓｈ（B）→ｐｏｐ→ｐｏｐ→ｐｕｓｈ（C）→ｐｏｐ の順序で出力可
［B→C→A］
　ｐｕｓｈ（A）→ｐｕｓｈ（B）→ｐｏｐ→ｐｕｓｈ（C）→ｐｏｐ→ｐｏｐの順序で出力可
［C→A→B］
　ｐｕｓｈ（A）→ｐｕｓｈ（B）→ｐｕｓｈ（C）→ｐｏｐ→ｐｏｐ×　Ｂより先にＡは出力不可
［C→B→A］
　ｐｕｓｈ（A）→ｐｕｓｈ（B）→ｐｕｓｈ（C）→ｐｏｐ→ｐｏｐ→ｐｏｐ の順序で出力可

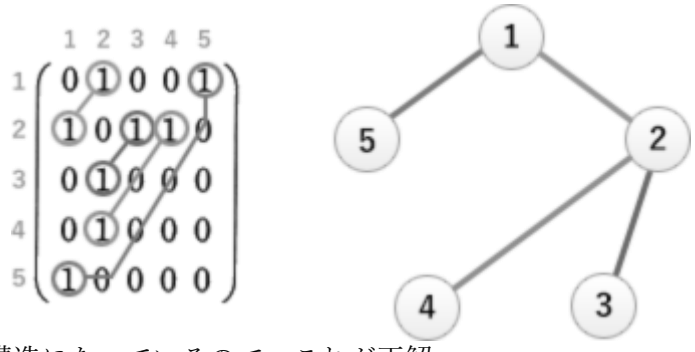
以上の５通りになる。

問 12　ウ
〔解説〕スタックではデータを挿入するＰＵＳＨ命令、取り出すＰＯＰ命令を使用してデータ操作を行う。
スタックは後入れ先出し(ＬＩＦＯ)のデータ構造
空のスタックを命令通りに操作していくと、

ＰＵＳＨ　１	１
ＰＵＳＨ　５	１，５
ＰＯＰ	１
ＰＵＳＨ　７	１，７
ＰＵＳＨ　６	１，７，６
ＰＵＳＨ　４	１，７，６，４
ＰＯＰ	１，７，６
ＰＯＰ	１，７
ＰＵＳＨ　３	１，７，３

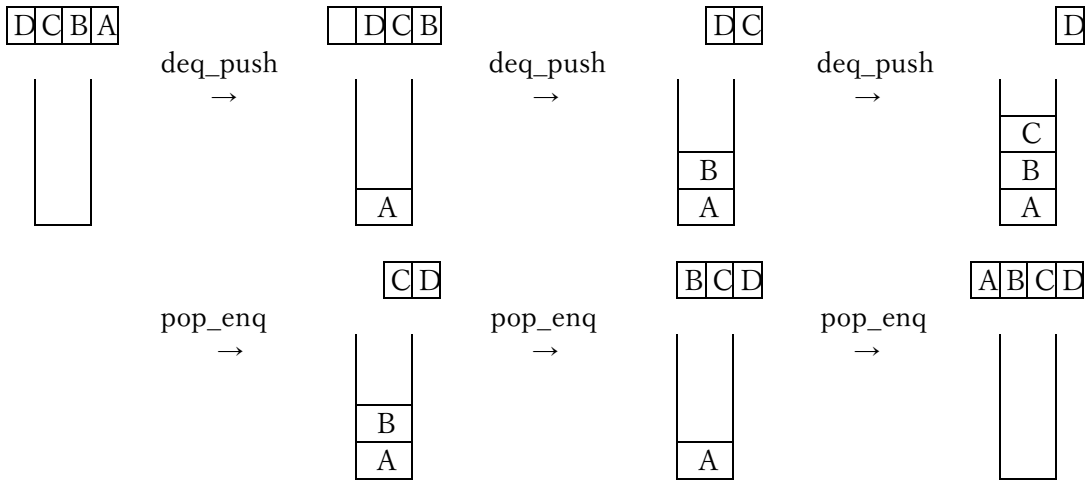
操作結果は「ウ」の状態になる。

問 13　イ
〔解説〕ノード間のエッジは"１－２","１－５","２－３","２－４"の４つ。



"２"を根とした木構造になっているので、これが正解。

問 14　イ
〔解説〕ｄｅｑ_ｐｕｓｈを３回，ｐｏｐ_ｅｎｑを３回実行すればよい。



問 15　ア
〔解説〕ＰＵＳＨ操作では配列の最終要素の添え字＋１の位置に新要素を追加、ＰＯＰ操作では配列の最終要素の添え字に位置する要素を削除することを行う。
つまり最低限「スタックに最後に入った要素を示す添字の変数」を保持していればスタック構造が実現できることになる。

問 16　イ
〔解説〕スタックとキューの内容は次のように変化していく。
１：push(a)…a をスタックに挿入する
２：push(b)…b をスタックに挿入する
３：enq(pop())…スタックから取り出したデータ(b)をキューに挿入する
４：enq(c)…c をキューに挿入する
５：push(d)…d をスタックに挿入する
６：push(deq())…キューから取り出したデータ(b)をスタックに挿入する
７：x←pop()…スタックから取り出したデータ(b)を x に代入する
したがって x に代入されるデータは b になる。

問 17　エ
〔解説〕a(ｉ，ｊ)＝2ｉ＋ｊ より、
a(１，１)＝２×１＋１＝３ …①
a(２，２)＝２×２＋２＝６ …②

①，②を設問の配列式に代入

a(a(１，１)×２，a(２，２)＋１)
＝a(３×２，６＋１)
＝a(６，７)
＝２×６＋７
＝<u>１９</u>

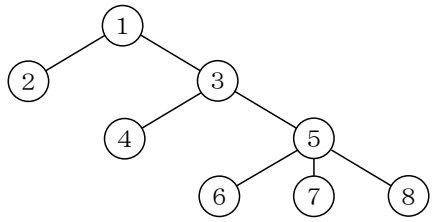
問 18　ア
〔解説〕配列を用いたリストでは各要素の位置を"先頭から何番目"というように指定して参照できる。リスト構造では、目的の要素に到達するまで先頭または後方から順番にポインタをたどっていく必要がある。

問 19　ウ
〔解説〕ア　挿入と削除の度に２種類のポインタを付け替える必要があるのでオーバーヘッドは増加する。
イ　項目を追加する場所に制限はない。
ウ　正解。片方向リンクでは、リストの最後の方にデータを挿入する場合には最初から順にポインタをたどっていく必要があるが、双方向リンクであれば、リストの後ろの方から逆にリストをたどることができるので、少ない参照回数で項目の追加・削除を行うことができる。
エ　項目を取り外す場所に制限はない。

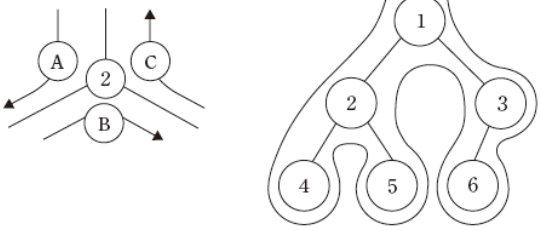
16-7 木(ツリー)構造〔解答・解説〕

問 1　イ
〔解説〕節点の総数が１５である２分探索木は４階層であるから，比較回数は４となる。

問 2　ウ
〔解説〕配列Ａを木構造で表してみると下図のようになり，葉（子をもたない節）の数は５となる。



問 3　エ
〔解説〕下図は配列番号６までを問題の方式に沿って２分木で表現したものである。行きがけながら２のノードをＡのタイミングで、通り掛けならＢのタイミングで、帰りがけならＣのタイミングで調べると各選択肢は以下のようになる



ア　１２４５３６となる
イ　４５２６３１となる
ウ　４２５１６３となる

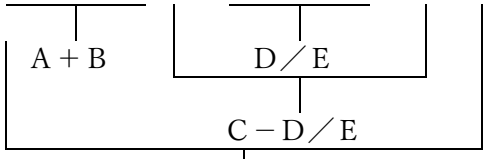
なお幅優先は深さの浅いノードから、同じ深さで左にあるノードから順に調べる。
エ　１２３４５６となる

問 4　ウ
〔解説〕ア，イ　１２が１０の左部分木に属している
エ　１２が１５の右部分木に属している

問 5　イ
〔解説〕① ルートノード 1 を探索
② ノード 1 の左部分木の節点 2 を探索
③ 節点 2 の左部分木のノード 4 を探索
④ 節点 4 の右部分木のノード 5 を探索
⑤ ノード 1 の右部分木の節点 3 を探索
⑥ ノード 3 に左部分木はないので右部分木のノード 6 を探索

問 6　エ
〔解説〕① 8 を根に置く
② １２＞８なので，１２を８の右部分木に置く
③ ５＜８なので，５を８の左部分木に置く
④ ３＜８，３＜５なので，３を５の左部分木に置く
⑤ １０＞８，１０＜１２なので，１０を１２の左部分木に置く
⑥ ７＜８，７＞５なので，７を５の右部分木に置く
⑦ ６＜８，６＞５，６＜７なので，６を７の左部分木に置く

問 7　ウ
〔解説〕A　B　＋　C　D　E　／　－　＊



(A＋B)＊(C－D／E)＝(１＋３)＊(５－４／２)＝１２

問 8　ウ
〔解説〕逆ポーランド表記法で表現された式の中で、[文字,文字,演算子]のパターンになっている箇所に注目。
問題中の式"EF－G÷CD－AB＋÷＋"では、EF-, CD-及び AB+がそれに当たる。
EF－G÷CD－AB＋÷＋

この[文字,文字,演算子]のパターンにマッチした場所を、[文字,演算子,文字]に変える。
(E－F)G÷(C－D)(A＋B)÷＋

上記のように変換した部分については、そのかたまりを一つの項として扱うので括弧で括っておくとわかりやすい。

再び、式の中から[文字,文字,演算子]のパターンになっている箇所を探す。先ほど変換した部分についても一つの文字(項)として見てみると、(E－F)G÷と(C－D)(A＋B)÷がパターンにマッチする。
(E－F)G÷(C－D)(A＋B)÷＋

先ほどと同じように、パターンにマッチした部分を、[文字,演算子,文字]に変える。
((E－F)÷G) ((C－D)÷(A＋B))＋

まだ右端に ＋の演算子が残っているので、再び同じ処理を繰り返し、[文字,演算子,文字]に変換する。
((E－F)÷G)＋((C－D)÷(A＋B))　（ウ）

問 9　イ

〔解説〕Y＝(A＋B)×(C－(D÷E))を、一つずつ順番に逆ポーランド表記法に変換する。

1：まず括弧内のA＋B と D÷Eを変換する。

Y＝AB＋×(C－DE÷)

2：次にもう一つの括弧内の(C－DE÷)を変換する。

Y＝AB＋×CDE÷－

3：次に右辺でまだ演算をしていない、"×"の左側と右側で演算する。

先程と同様に「AB＋」×「CDE÷－」⇒AB＋CDE÷－×と考える。

Y＝AB＋CDE÷－×

4：最後に左辺と右辺を"＝"で演算して逆ポーランド表記法への変換が完了。

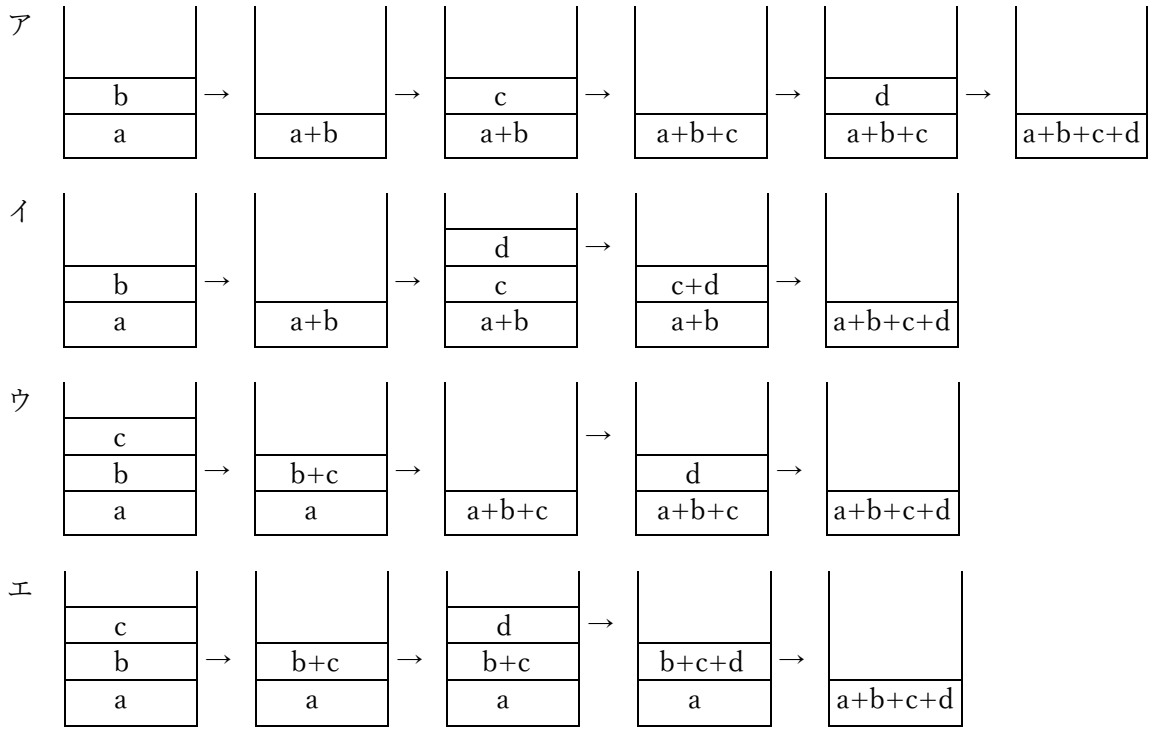
YAB＋CDE÷－×＝　(イ)

問 10　ア

〔解説〕左部分木の空白になっている3つの節には、それぞれ4より小さい値が入るので「3，2，1」が入ることになる。残る数字は「6，7，8，9，１０」の５つだが、問題の大小関係を満たすためには節ａの値に、５より大きく、残る５つの数字の中で最も小さい値が入る必要があるため節ａ＝６が適切である。また節ｂの値は、「左の子の値＜親の値＜右の子の値」という条件から考えると、残る４つの数字の中で最も小さい７が収まることになる。

問 11　ア

〔解説〕スタックによる逆ポーランド表記法では，演算子が出現した時点で直近の２つの値を式に置き換えるので，各式のスタックの状態は以下のように変化する。



問 12　ウ

〔解説〕前置記法：演算子をオペランドの前に置く方法である。 節に入ったときにそこの記号を書くので、アは間違い。

中置記法：演算子をオペランドの間に置く方法である。よって、イは間違い。

後置記法：演算子をオペランドの後に置く方法である。 節から上に戻るときにそこの記号を書くのでウが正しい。

問 13　ウ
〔解説〕２分探索木では，各節がもつデータは「その節から出る左部分木にあるどのデータよりも大きく、右部分木のどのデータよりも小さい」という条件があり、これを利用して効率的なデータ探索を可能にしている。
ア　節点となる要素９の左部分木に要素１０と要素１１が存在することになるので不適切
イ　節点となる要素１０の左部分木に要素１１が存在することになるので不適切
エ　節点となる要素１４の右部分木に要素１３が存在することになるので不適切

問 14　エ
〔解説〕１となっているのは第１行第２列，第１行第３列，第２行第１列，第２行第４列，第３行第１列，第３行第４列，第４行第２列，第４行第３列なので，$V1$ と $V2$，$V1$ と $V3$，$V2$ と $V4$，$V3$ と $V4$ を結ぶ枝が存在する。

16-8　データを探索するアルゴリズム〔解答・解説〕

問 1　ウ
〔解説〕ア，イ　Ｘと同じ値が配列中にない場合，ｋにはＮ＋１が設定される
エ　Ｘと同じ値が配列の１番目とＮ番目にある場合，ｋには１が設定される

問 2　ウ
〔解説〕ア　２分探索法の説明
イ　別のキー値から同一の格納アドレスが求められてしまうことがある
エ　探索時間は表全体の大きさにかかわらず常に一定

問 3　イ
〔解説〕整列されているデータ数が４倍になると，元の個数になるまで２分割を２回する必要があるから，最大探索回数は２回増えることになる。

問 4　イ
〔解説〕ア　２分探索法では、探索対象のデータが昇順または降順に整列されている必要がある
ウ　ハッシュ法では、探索対象のデータがハッシュ関数で計算された位置に格納されていなければならない
エ　モンテカルロ法は、乱数を用いた試行を繰り返すことで問題に対する近似値を得る方法

問 5　ウ
〔解説〕$[\log_2 2000]+1$
$=[\log_2 2+\log_2 1000]+1$
$=[\log_2 1000]+2$

$=\left[\dfrac{\log_{10} 1000}{\log_{10} 2}\right]+2$

$=\left[\dfrac{3}{0.301}\right]+2=11$回

（別解）２０００を商が０になるまで割り続けた除算回数を数えてもよい

問 6　ア
〔解説〕２分探索法の計算量は，平均探索回数が［$\log_2 n$］，最大探索回数が［$\log_2 n+1$］なので，
オーダ表記法で表すと，O($\log n$)（ア）となる。

※$\log_2 n = \dfrac{\log_{10} n}{\log_{10} 2} = \dfrac{\log n}{\log 2} = \dfrac{\log n}{0.301}$　なので，O($\log n$)となる。

問 7　イ
〔解説〕ハッシュ値が一様に平均的に分布すれば同一のハッシュ値となる確率が最も低くなる。

問 8　イ
〔解説〕中央値を求める処理であるから，(上限＋下限)／２となる。

問 9　ウ
〔解説〕a　中央値＜探索値の場合であるから，中央値＋１を新しい下限値とする
b　中央値＞探索値の場合であるから，中央値－１を新しい上限値とする

16-9 データを整列させるアルゴリズム〔解答・解説〕

問 1　ア
〔解説〕次のような例で考えてみる。

	1	2	3	4	5
A	4	7	9	2	6

このとき，n＝5

			1	2	3	4	5
i＝1	j＝5	A[5]＜A[4]ではないので交換しない	A	4	7	9	2	6
j＝4		A[4]＜A[3]なので交換	A	4	7	2	9	6
j＝3		A[3]＜A[2]なので交換	A	4	2	7	9	6
j＝2		A[2]＜A[1]なので交換	A	2	4	7	9	6

以上のように，A[1]が最小値になる。

問 2　ウ
〔解説〕ア　基本挿入法の説明
イ　基本選択法の説明
エ　バブルソート(基本交換法)の説明

問 3　ア
〔解説〕(1)：H←9÷3＝3
(2)：要素ごとが互いに3つずつ離れた要素から成る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
(4)：H≠0なので2に戻る

(2)：要素ごとが互いに1つずつ離れた要素から成る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
(3)：H←1÷2＝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。よって(3)の処理が繰り返される回数は2回

16-11 オブジェクト指向プログラミング〔解答・解説〕

問 1　エ

問 2　ア

問 3　エ
〔解説〕同じ指示(メッセージ)を異なるオブジェクト(営業課長と営業担当者)に送ったときに、異なるサービスを返している。このような特性はオブジェクト指向ではポリモーフィズムと呼ばれる。

問 4　ウ
〔解説〕インヘリタンス(継承)とは、オブジェクト指向で上位クラスの属性やメソッドを引き継いだ下位クラスをつくることを言う。アは集約、イはカプセル化、エはクラス化の説明である。

問 5　ア
〔解説〕イはオーバライド，ウはインヘリタンス，エはポリモアフィズムの説明である。

問 6　エ
〔解説〕ア　抽象クラスでは、複数のクラスに共通するメソッドの(実際の処理内容を含まない)意味だけを定義することが可能
イ　カプセル化によってデータと手続きをオブジェクト内部にまとめることで相互依存性は低くなる
ウ　下位クラスが独自に定義した性質は上位クラスに影響を与えないので、上位クラスを変更する必要はない

問 7　イ
〔解説〕ア　複数のクラス間で定義できる
ウ　依頼しなくても操作を実行できる
エ　部品オブジェクトが集約オブジェクトの属性と操作を共有する

問 8　ア
〔解説〕多相性（ポリモーフィズム）とは，同じメッセージを送ってもオブジェクトごとに異なる操作が行われることをいい，オーバライド（スーパクラスのメソッドをサブクラスで上書きすること）などによって実現することができる。

問 9　ウ

問 10　ア
〔解説〕イ　カプセル化により，オブジェクト外部からオブジェクト内部のデータを操作できないようにする
ウ　下位クラスに必要な機能や性質の全てが上位クラスに含まれるわけではない
エ　データをデータ辞書に登録する必要はない

問11　ウ
〔解説〕つまりこの問題では、同じ指示(メッセージ)を異なるオブジェクト(営業課長と営業部員)に送ったときに、異なるサービスを返している。このような特性はオブジェクト指向では多相性と呼ばれる。

問 12　エ
〔解説〕「関数呼び出しと関数本体を実行時に結びつける」というオブジェクト指向言語の特徴を、動的結合(動的束縛)という。また、静的結合は、実行時ではなくコンパイル時に実行される手続きが決定される性質である。

問 13　イ
〔解説〕ア　説明が逆で、クラスはインスタンスの仕様を定義したもの
ウ　１つのインスタンスは、そのクラスの仕様を表すただ１つのクラスと関連付けられる
エ　１つのクラスから、複数のインスタンスを生成できる

問 14　イ
〔解説〕ア　オーバーライドは、スーパークラスで定義されたメソッドをサブクラスで再定義すること
ウ　カプセル化は、オブジェクト内の詳細な仕様や構造を外部から隠蔽すること
エ　汎化は、複数のクラスの共通する性質をまとめて，抽象化したクラスを作ること

問 15　イ
〔解説〕[多相性]
同じメソッドが，クラスによって異なる動作をすることであり，ポリモーフィズムともいう。
[オーバライド]
スーパークラス（上位クラス）で定義されたメソッドをサブクラス（下位クラス）で再定義すること。これにより動作（振る舞い）を変更することができる。したがって，同一のメッセージを送ってもオブジェクトごとに違う操作が実現できる。

問 16　エ
〔解説〕ア　構造化と投影が誤り
イ　構造化と連続が誤り
ウ　正規化と分割が誤り

問17　ア
〔解説〕カプセル化は、オブジェクト指向においてデータ(属性)とそのデータに対する手続きをひとつにまとめて扱うことを意味する。

16-12 UML(Unified Modeling Language)〔解答・解説〕

問1　エ
〔解説〕ア　アクティビティ図は、上流行程のビジネスプロセスの流れや下流行程のプログラムの制御フローなどシステムの流れを表せるフローチャートのＵＭＬ版
イ　オブジェクト図は、特定の時点でのオブジェクトのインスタンス間の静的な構造を記述する図
ウ　クラス図は、クラス、属性、クラス間の関係からシステムの構造を記述する静的な構造図

問2　ア
〔解説〕イ　クラス図は、クラス間の静的な関係を表現する図
ウ　ＵＭＬは、プログラム言語ではない
エ　ＵＭＬは、ソフトウェア開発プロセスを標準化したものではない

問3　ウ
〔解説〕ア　システムなどのフローを記述する図
イ　インタフェースを介したコンポーネント同士の関係やコンポーネントの内容を表現する図
エ　時間の経過や状態の変化に応じて状態が変わるようなシステムの振る舞いを記述するときに適した図式化手法

問4　ウ
〔解説〕ＵＭＬ（Unified Modeling Language：統一モデリング言語）では，プログラムの構造図として，クラス図，コンポーネント図，パッケージ図などが，プログラムの振舞い図として，シーケンス図，ユースケース図，状態遷移図，コミュニケーション図などが用いられる。

問5　ウ
〔解説〕ア　Ｅ－Ｒ図の説明
イ　ＤＦＤ(Data Flow Diagram)の説明
エ　ワークフローとしてビジネスプロセスをグラフィカルに記述するＢＰＤ(Business Process Diagram)の説明

問6　ウ
〔解説〕シーケンス図では，システムの動的な振る舞いを表すもので，オブジェクト間の相互作用を時間的な流れの上でわかるようにした図である。

問7　イ

問8　ア
〔解説〕イ　組織の多重度が１なので，社員は１つの組織に必ず所属する
ウ　組織の多重度が１なので，社員は複数の組織に所属することができない
エ　社員の多重度が１..＊なので，１つの組織に複数の社員が所属できる

問 9　イ

〔解説〕ア　社員とプロジェクト内役割分担の関係は多対多なので同じ役割分担とは限らない

ウ　社員に対するプロジェクト参画の多重度は 0 ..＊なので，プロジェクト参画していない社員もいる

エ　社員に対する部門の多重度は 1 なので，複数の部門に所属することはできない

問 10　ウ

〔解説〕部門と社員の関係は"1 対 0 以上"を表現している。

ア　所属する社員が 0 人の部門も登録できる

イ　社員は 1 つの部門に所属する

エ　1 つの部門には複数の社員が所属できる

問 11　エ

〔解説〕ア　1 回の納品に対して複数の請求が発生するので、「納品―請求は、1 対 1 以上」の関係になる

イ　納品に対して請求がない場合が存在するので、「納品―請求は、1 対 0 または 1 」の関係になる

ウ　納品と請求が 1 対 1 で対応するので、「納品―請求は、1 対 1 」の関係になる

問 12　イ

〔解説〕ア　部門クラスとの間に"管理する"の関連名が付いているのは事業部→部門のみなので、事業部のみが部門を管理できる

ウ　部門から見た社員の多重度は 1 以上。即ち社員が所属しない部門は存在しない

エ　部門から見た事業部の多重度は 0 または 1 。即ち事業部に管理されていない部門も存在し得る

問 13　イ

〔解説〕ア　関連は、クラス間の結びつきを表し、クラス間の実線で記述します。

ウ　集約は、クラス間の「集約－分解」を表し、ひし形の白抜きを使用した矢印で記述します。

エ　クラス図にユースケース名は登場しません。

問 14　エ

〔解説〕ア　クラス図は、オブジェクト、属性、オブジェクト間の関係からシステムの構造を記述する静的な構造図。(a)

イ　コラボレーション図は、オブジェクトの振る舞いをオブジェクト間の接続関係に着目して表現した図。(d)

ウ　ステートチャート図は、状態遷移図を基本にして、オブジェクト内に存在する状態や、その状態の移り変わりを表現する図。(c)