FE

平成21年度　秋期
基本情報技術者試験
午後　問題

試験時間	13:00 ～ 15:30（2時間30分）

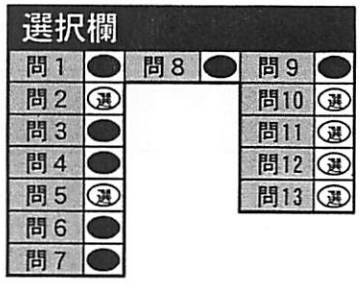
注意事項

1. 試験開始及び終了は，監督員の時計が基準です。監督員の指示に従ってください。
2. 試験開始の合図があるまで，問題冊子を開いて中を見てはいけません。
3. この注意事項は，問題冊子の**裏表紙**に続きます。必ず読んでください。
4. **答案用紙への受験番号などの記入は，試験開始の合図があってから始めてください。**
5. 問題は，次の表に従って解答してください。

問題番号	問1 ～ 問7	問8	問9 ～ 問13
選択方法	5問選択	必須	1問選択

6. 答案用紙の記入に当たっては，次の指示に従ってください。
   (1) B又はHBの黒鉛筆又はシャープペンシルを使用してください。訂正の場合は，あとが残らないように消しゴムできれいに消し，消しくずを残さないでください。
   (2) 答案用紙は光学式読取り装置で処理しますので，答案用紙の**マークの記入方法**のとおりマークしてください。
   (3) **受験番号欄**に，**受験番号**を記入及びマークしてください。正しくマークされていない場合，答案用紙の**マークの記入方法**のとおりマークされていない場合は，採点されません。
   (4) **生年月日欄**に，受験票に印字されているとおりの**生年月日**を記入及びマークしてください。正しくマークされていない場合は，採点されないことがあります。
   (5) **選択した問題**については，右の例に従って，**選択欄**の**問題番号**の(選)をマークしてください。マークがない場合は，採点の対象になりません。問1～問7について，6問以上マークした場合は，はじめの5問を採点します。問9～問13について，2問以上マークした場合は，はじめの1問を採点します。

〔問1，問3，問4，問6，問7，問9を選択した場合の例〕



   (6) 解答は，次の例題にならって，解答欄にマークしてください。

〔例題〕　次の［　　］に入れる正しい答えを，解答群の中から選べ。

秋の情報処理技術者試験は，［ a ］月に実施される。

解答群

ア　8　　イ　9　　ウ　10　　エ　11

正しい答えは“ウ　10”ですから，次のようにマークしてください。

例題	a	(ア) (イ) ● (エ)

裏表紙の注意事項も，必ず読んでください。

〔問題一覧〕

●問 1～問 7（7 問中 5 問選択）

問題番号	出題分野	テーマ
問 1	ハードウェア	半加算器と全加算器
問 2	データベース	倉庫内の保管棚を用いた書類管理
問 3	ネットワーク	データ送信とその符号化
問 4	情報セキュリティ	利用者認証
問 5	ソフトウェア設計	航空券発券システム
問 6	IT サービスマネジメント	インシデント及び問題の管理
問 7	システム戦略	情報システムの効果見積り

●問 8（必須問題）

問題番号	出題分野	テーマ
問 8	データ構造及びアルゴリズム	数値計算と計算誤差

●問 9～問 13（5 問中 1 問選択）

問題番号	出題分野	テーマ
問 9	ソフトウェア開発（C）	多倍長整数の加算
問 10	ソフトウェア開発（COBOL）	売上データのマスタへの反映と対前年同月比表示
問 11	ソフトウェア開発（Java）	携帯電話の料金計算
問 12	ソフトウェア開発（アセンブラ）	ビット列の置換え
問 13	ソフトウェア開発（表計算）	勤怠管理と出勤割当て

共通に使用される擬似言語の記述形式

擬似言語を使用した問題では，各問題文中に注記がない限り，次の記述形式が適用されているものとする。

〔宣言，注釈及び処理〕

記述形式		説明
○		手続，変数などの名前，型などを宣言する。
/* 文 */		文に注釈を記述する。
処理	・変数 ← 式	変数に式の値を代入する。
	・手続(引数，…)	手続を呼び出し，引数を受け渡す。
	▲ 条件式 　処理 ▼	単岐選択処理を示す。 　条件式が真のときは処理を実行する。
	▲ 条件式 　処理 1 ―― 　処理 2 ▼	双岐選択処理を示す。 　条件式が真のときは処理 1 を実行し，偽のときは処理 2 を実行する。
	■ 条件式 　処理 ■	前判定繰返し処理を示す。 　条件式が真の間，処理を繰り返し実行する。
	■ 　処理 ■ 条件式	後判定繰返し処理を示す。 　処理を実行し，条件式が真の間，処理を繰り返し実行する。
	■ 変数：初期値，条件式，増分 　処理 ■	繰返し処理を示す。 　開始時点で変数に初期値（式で与えられる）が格納され，条件式が真の間，処理を繰り返す。また，繰り返すごとに，変数に増分（式で与えられる）を加える。

〔演算子と優先順位〕

演算の種類	演算子	優先順位
単項演算	+，−，not	高 ↑
乗除演算	×，÷，%	
加減演算	+，−	
関係演算	＞，＜，≧，≦，＝，≠	
論理積	and	
論理和	or	↓ 低

注　整数同士の除算では，整数の商を結果として返す。％演算子は，剰余算を表す。

〔論理型の定数〕

true，false

問題文中で共通に使用される表記ルール

E-R 図の表記ルールを次に示す。各問題文中に注記がない限り，この表記ルールが適用されているものとする。

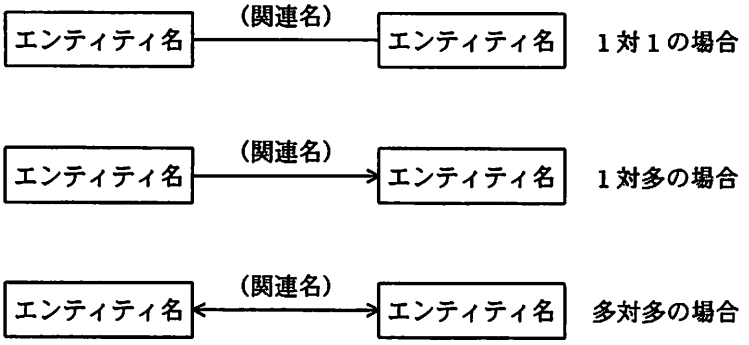


図　エンティティと関連の表記ルール

(1) エンティティを長方形で表す。

(2) 長方形の中にエンティティ名を記入する。

(3) エンティティ間の関連を直線又は矢印で表す。線のわきに関連名を“（関連名）”として記入する。

なお，関連名は省略することもある。

(4) “1対1”の関連は，直線で表す。

“1対多”の関連は，“多”側を指す片方向矢印とする。

“多対多”の関連は，両方向矢印とする。

次の問1から問7までの7問については，この中から5問を選択し，答案用紙の選択欄の(選)をマークして解答してください。

なお，6問以上選択した場合には，はじめの5問について採点します。

問1　半加算器と全加算器に関する次の記述を読んで，設問1～3に答えよ。

(1)　1ビット同士を加算する半加算器の真理値表を，表1に示す。

```
      X
+     Y
-------
  C   Z
```

C：けた上がり

表1　半加算器の真理値表

X	Y	C	Z
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(2)　下位からのけた上がり C_{in} を考慮して1ビット同士を加算する全加算器の真理値表を，表2に示す。

```
      X
      Y
+     Cin
-------
  C   Z
```

C_{in}：下位からのけた上がり

C：けた上がり

表２　全加算器の真理値表

C_{in}	X	Y	C	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

設問１　半加算器を実現する論理回路を，図１に示す。図１中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，AND は論理積，OR は論理和，XOR は排他的論理和，NAND は否定論理積，NOR は否定論理和を表す。

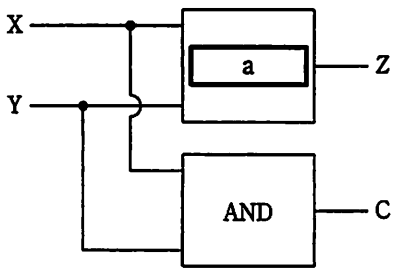


図１　半加算器を実現する論理回路

解答群

ア　AND　　イ　NAND　　ウ　NOR　　エ　OR　　オ　XOR

設問2　全加算器を実現する論理回路について，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。

全加算器は，図2に示すように半加算器を2段に接続して実現する。半加算器1はXとYを加算し，半加算器2は半加算器1の結果とC_{in}を加算する。このとき，半加算器1のけた上がりをC_1，半加算器2のけた上がりをC_2とする。X，Y，C_{in}と，C_1，C_2との関係は表3のとおりになる。

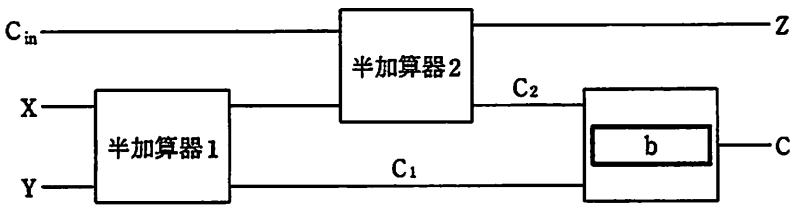


図2　全加算器を実現する論理回路

表3　X，Y，C_{in}と，C_1，C_2との関係

C_{in}	X	Y	C_1	C_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	1
1	1	1	c	

bに関する解答群

ア　AND　　イ　NAND　　ウ　NOR　　エ　OR

cに関する解答群

	C_1	C_2
ア	0	0
イ	0	1
ウ	1	0
エ	1	1

設問3　A，B及びSを2の補数表現による4ビットの符号付2進整数とし，それぞれのビット表現を$A_4A_3A_2A_1$，$B_4B_3B_2B_1$及び$S_4S_3S_2S_1$で表す（符号ビットはA_4，B_4及びS_4）。

図3は，AとBの加算を行い，結果をSに求める加算器であり，半加算器と全加算器で実現されている。ここで，C_1～C_4は半加算器及び全加算器からのけた上がりを表す。

この加算器に，Aとして−1を，Bとして−2（いずれも10進表記）を与えたとき，図3のC_1～C_4の値として正しい組合せを，解答群の中から選べ。

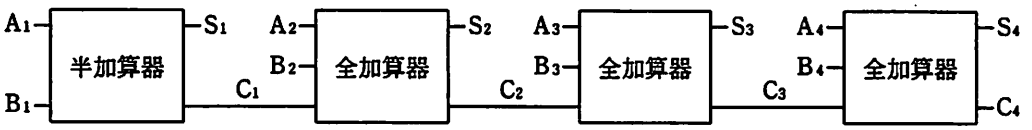


図3　AとBを加算してSを求める加算器

解答群

	C_1	C_2	C_3	C_4
ア	0	1	0	0
イ	0	1	0	1
ウ	0	1	1	0
エ	0	1	1	1
オ	1	0	0	0
カ	1	0	0	1
キ	1	0	1	0
ク	1	0	1	1

問2　倉庫内の保管棚を用いた書類管理に関する次の記述を読んで，設問1～3に答えよ。

A社では，書類の保管管理（以下，書類管理という）を関係データベースを用いて行っている。書類管理の概要と書類管理データベースの概要は，次のとおりである。

〔書類管理の概要〕

(1) 書類は，倉庫に配置された保管棚の中の箱に入れて保管される。書類には，全社を通じて一意となる書類番号が付与されている。

(2) 倉庫は 1棟だけであり，3階建てで，各階は四つの区画に分けられている。各区画には複数の保管棚が配置されている。

(3) 保管棚には，倉庫内でその保管棚を一意に識別できるように，棚番号が付与されている。保管棚は，倉庫内のどの区画に配置してもよい。また，一つの保管棚には，複数の箱がある。

(4) 箱には，一つの保管棚の中で一意に識別できるように保管棚の段と列を組み合わせた箱番号が付与されている。例えば，3段目4列の箱番号は3-4となる。一つの箱には，複数の書類を保管できる。

(5) 書類管理の担当者は，社内の部署から預かった書類を，保管棚のいずれかの箱に入れて保管する。保管の際，全社を通じて一意となる預託番号を付与する。

(6) 預かった書類は返却依頼を受けて返却する。

(7) 倉庫，保管棚及び箱の構成を図1に示す。

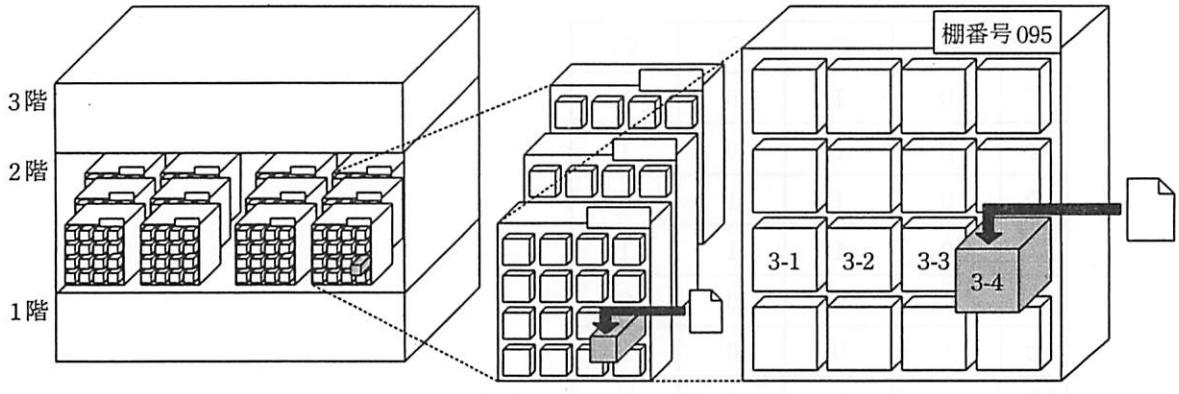


図1　倉庫，保管棚及び箱の構成

〔書類管理データベースの概要〕

(1) 階数表は，倉庫の階番号と階名称を管理する。

(2) 区画表は，倉庫の各階における区画名称を管理する。

(3) 棚表は，保管棚の配置された日付，位置（階番号と区画番号）及び棚名称を管理する。

(4) 箱表は，箱名称を管理する。

(5) 書類預託表は，部署から預かった書類の保管状況（預託日と返却日）を管理する。預託日には，書類を預かった日付が設定される。返却日には，書類を預かったときにNULLが設定され，書類を返却したときにその日付が設定される。

(6) 書類表は，書類の預け元の部署番号と書類名称を管理する。

(7) 部署表は，部署名称を管理する。

(8) A社の書類管理データベースのE-R図を図2に，構造の一部を図3に示す。

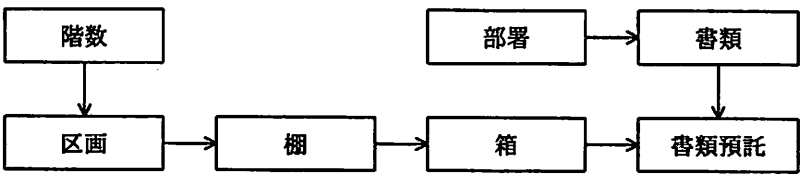


図2 A社の書類管理データベースのE-R図

階数表（階番号，階名称）
区画表（区画番号，階番号，区画名称）
棚表（棚番号，階番号，区画番号，棚名称，棚配置日）
箱表（箱番号，棚番号，箱名称）
書類預託表（預託番号，書類番号，箱番号，棚番号，預託日，返却日）
書類表（書類番号，部署番号，書類名称）
部署表（部署番号，部署名称）

注 下線はキー項目を表す。

図3 A社の書類管理データベースの構造（一部）

設問1　書類を保管していない保管棚が各階に幾つあるかを検索する。次のSQL文中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
SELECT 棚表.階番号, COUNT(棚表.棚番号) FROM 棚表
  WHERE 棚表.棚番号
      [ a ] (SELECT 書類預託表.棚番号 FROM 書類預託表
                WHERE 書類預託表.返却日 IS NULL)
      [ b ] 棚表.階番号
```

解答群

ア EXISTS	イ GROUP BY	ウ IN
エ LIKE	オ NOT IN	カ ORDER BY

設問2　預かってから1年を超え，未返却である書類を検索する。

検索日を2009年10月18日とする場合，次のSQL文中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
SELECT 書類預託表.書類番号, 書類表.書類名称, 書類預託表.預託日,
       棚表.階番号, 棚表.区画番号, 書類預託表.棚番号,
       書類預託表.箱番号
  FROM 棚表, 書類預託表, 書類表
  WHERE [ c ]
        AND [ d ]
        AND 書類預託表.返却日 IS NULL
        AND 書類預託表.預託日 <= '20081018'
```

解答群

ア　書類預託表.書類番号 = 書類表.書類番号

イ　書類預託表.書類番号 <> 書類表.書類番号

ウ　書類預託表.預託日 <= 書類預託表.返却日

エ　棚表.棚配置日 = 書類預託表.預託日

オ　棚表.棚配置日 <= 書類預託表.預託日

カ　棚表.棚番号 = 書類預託表.棚番号

キ　棚表.棚番号 <> 書類預託表.棚番号

設問3　A社では倉庫内の利便性を高めるため，保管棚の配置を変更したり，ある箱に入っている書類を別の箱に移し替えたりしている。A社の書類管理データベースの説明として，正しい答えを解答群の中から選べ。

解答群

ア　ある箱に入っている書類を，同一の保管棚の別の箱に移す場合，箱表と書類預託表のそれぞれを更新すればよい。

イ　ある箱に入っている書類を，別の保管棚の別の箱に移す場合，書類預託表の棚番号だけを更新すればよい。

ウ　保管している書類の保管棚とその位置を検索する場合，書類番号だけでは検索できない。

エ　保管棚の配置されている区画名称を変更する場合，区画表は更新しなくてよい。

オ　保管棚の配置を変更する場合，書類預託表は更新しなくてよい。

問3　データ送信とその符号化に関する次の記述を読んで，設問1～3に答えよ。

(1) 機器Aにはセンサが一つ接続されており，接続されたセンサから4バイト（1バイトは8ビット）の符号付整数で表される値（以下，測定値という）を1秒当たり100回取得する。

(2) 機器Aは，図に示す構造のパケットに測定値を格納し，ネットワークを経由して送信する。一つのパケットには，連続する複数の測定値を格納する。ネットワークはデータの送信に十分な帯域をもつ。

(3) パケットは，150バイトのヘッダと測定値の列で構成される。ただし，パケットの最大長は1,478バイトとする。

(4) 一つのパケットに格納する測定値の個数はヘッダに格納され，(3)の条件を満たす範囲で，任意に設定できる。

(5) 機器Aは，設定した個数分の測定値をセンサから取得後，遅滞なく送信する。

(6) 機器Aは，測定値の取得と送信を同時に行うのに十分な能力をもつ。

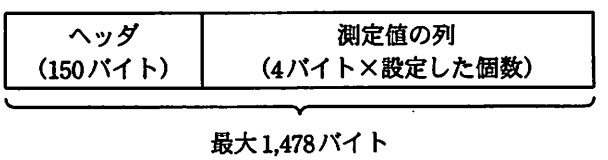


図　パケットの構造

設問1　1パケットに格納する測定値の個数と単位時間当たりの送信量（ヘッダと測定値の総量）の関係の記述として正しい答えを，解答群の中から選べ。

解答群

ア　1パケットで送信する測定値の個数が多いほど，単位時間当たりの送信量は多くなる。

イ　1パケットで送信する測定値の個数が多いほど，単位時間当たりの送信量は少なくなる。

ウ　1パケットで送信する測定値の個数が変わっても，単位時間当たりの送信量は変わらない。

設問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。

一つのパケットには，最大［ a ］秒分の測定値を格納できる。

また，測定値の送信に必要なネットワーク帯域 w は次の式で表せる。ただし，1パケットに格納する測定値の個数を n とする。

w＝［ b ］×8×(150＋［ c ］)　ビット／秒

aに関する解答群

ア　1.66　　イ　3.32　　ウ　6.64　　エ　13.28　　オ　26.56

b，cに関する解答群

ア　100　　イ　150　　ウ　1,200　　エ　4n　　オ　32n

カ　100n　　キ　1／n　　ク　100／n　　ケ　n　　コ　n／100

設問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。

測定値の時刻による変動は小さいことが多く，例えば，全体の70％の測定値は一つ前の測定値との差が，$-128 \sim 127$（$-2^7 \sim 2^7-1$）の範囲にあることが分かった。

そこで，測定値を次の方法で圧縮して送ることにする。

① パケットの先頭に格納する測定値は，これまでどおり格納する。

② 2番目以降に格納する測定値は，一つ前の測定値との差を，表の“圧縮符号のビット長”で示す長さ（差の値によって異なる）に符号化し，パケットにビット単位で詰めて格納する。例えば，2番目以降に格納する測定値のビット数は，一つ前の測定値との差が10ならば9ビットに，200ならば18ビットになる。

なお，圧縮後の測定値の列のビット長は，ヘッダに設定する。

表　差の符号化方式と出現確率

差の範囲	$-2^7 \sim 2^7-1$	$-2^{15} \sim 2^{15}-1$（$-2^7 \sim 2^7-1$は除く）	$-2^{23} \sim 2^{23}-1$（$-2^{15} \sim 2^{15}-1$は除く）	$-2^{31} \sim 2^{31}-1$（$-2^{23} \sim 2^{23}-1$は除く）
圧縮符号	0 差（8ビット）	10 差（16ビット）	110 差（24ビット）	111 差（32ビット）
圧縮符号のビット長	9	18	27	35
出現確率	70％	25％	4％	1％

一つ前の測定値との差の分布は，表の“出現確率”のとおりであるとすると，2番目以降の測定値の圧縮符号のビット長の期待値は，測定値一つ当たり［　d　］ビットである。

解答群

ア　9.0　　イ　12.23　　ウ　15.575　　エ　22.25　　オ　32.0

問4　利用者認証に関する次の記述を読んで，設問1，2に答えよ。

X社では，社外の端末から社内のサーバへのリモートログインを可能にするため，利用者認証の方式を検討している。社内では，利用者IDとパスワードをサーバに送信する方式を使用しており，そのパスワードの強化を含め，次の三つの方式の安全性を検討している。

〔方式1：利用者IDとパスワード方式〕

端末は，利用者が入力した利用者IDとパスワードをサーバに送信する。サーバは利用者IDから登録されているパスワードを検索し，送信されたパスワードと照合することによって，ログインの可否を応答する。利用者IDとパスワード方式を図1に示す。

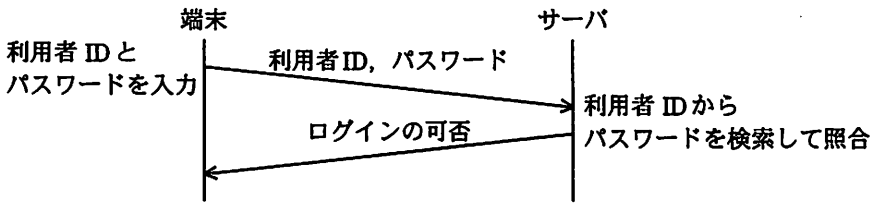


図1　利用者IDとパスワード方式

〔方式2：チャレンジレスポンス方式〕

端末は，利用者が入力した利用者IDをサーバに送信する。サーバは，利用者IDを受信すると，ランダムに生成したチャレンジと呼ばれる値cを端末に送信する。端末は，利用者が入力したパスワードpとチャレンジcから，ハッシュ値$h(p, c)$を計算して，レスポンスの値としてサーバに送信する。サーバは，利用者IDから登録されているパスワードp'を検索し，端末と同じハッシュ関数hを使って計算したハッシュ値$h(p', c)$とレスポンスの値とを照合することによって，ログインの可否を応答する。ここで，ハッシュ関数hは公知のものであり，どの端末でも計算可能とする。チャレンジレスポンス方式を図2に示す。

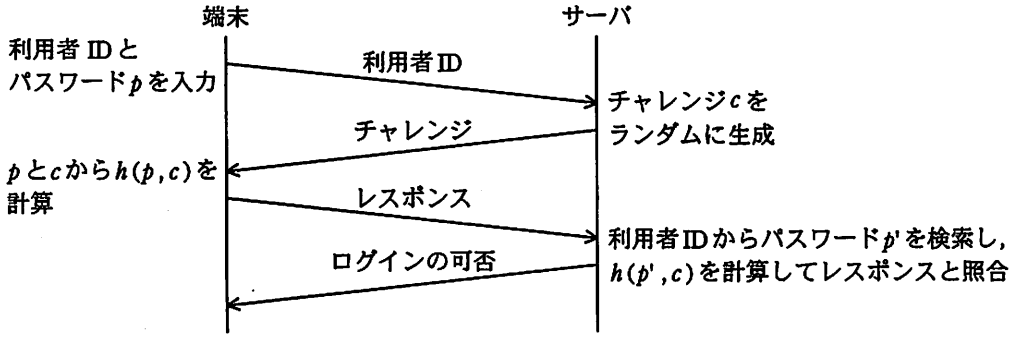


図2　チャレンジレスポンス方式

〔方式3：トークン（パスワード生成器）方式〕

利用者には，自身の利用者IDが登録されたトークンと呼ばれるパスワード生成器を配布しておく。トークンの例を図3に示す。

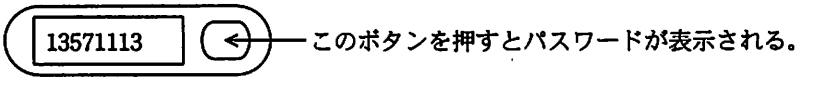


図3　トークンの例

トークンは時計を内蔵しており，関数gを使って，利用者IDであるuと時刻tに応じたパスワード$g(u,t)$を生成し表示することができる。利用者は，利用者IDとトークンが生成し表示したパスワードを入力し，端末はこれらをサーバに送信する。サーバは，利用者IDであるuとサーバの時刻tからトークンと同じ関数gを使って生成したパスワード$g(u,t)$と端末から受信したパスワードとを照合することによって，ログインの可否を応答する。

なお，トークンの時刻とサーバの時刻が同期していることは保証されており，トークンのパスワード表示からサーバにおけるパスワード生成までの遅延も，一定の時間は許容する。トークン方式を図4に示す。

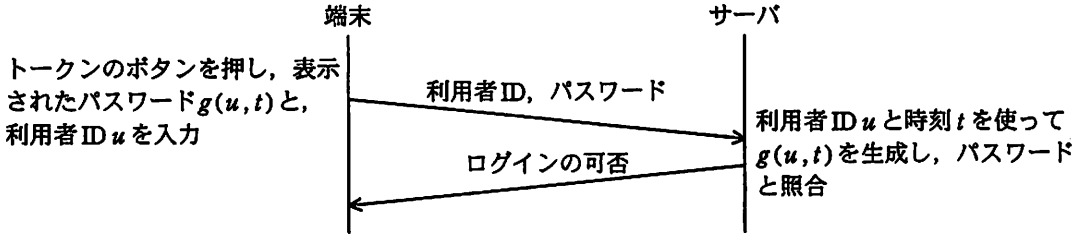


図4　トークン方式

設問1　パスワードの強度に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

方式1，2では，利用者がパスワードを設定する。これらの方式を採用する場合には，容易には推定されないパスワード，すなわち，十分な強度をもつパスワードを，利用者に設定してもらう必要がある。

パスワードの強度を高めるためには，パスワードを長くすることやパスワードに利用する文字の種類を増やすことが考えられる。例えば，英小文字26文字だけからなる8文字のパスワードに対して，総当たり方式による発見に必要な最大時間を1とすると，パスワードの長さを10文字にすれば必要な最大時間は［　a　］となる。また，同じ8文字であっても，英大文字も使用する場合，必要な最大時間は［　b　］となる。

解答群

ア　1.25	イ　2	ウ　208	エ　256
オ　260	カ　676	キ　1,024	

設問2　盗聴のリスクに関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。解答は，重複して選んでもよい。

利用者認証の方式によっては，不正な方法によって入手した情報（例えば利用者 ID とパスワード）をそのまま利用することによって，不正ログインが行われる可能性がある。

(1) 社外からの通信経路上で通信内容が盗聴された場合，盗んだ情報をそのまま利用することによって，利用者がパスワードを変更しない限り，サーバへの不正ログインがいつでも可能になるのは，［ c ］である。ただし，通信経路は暗号化されていないものとする。

(2) 社外からのリモートログインに利用する端末上で，キーボード入力を読み取って，第三者に送信するプログラムが動作していた場合，盗んだ情報をそのまま利用することによって，利用者がパスワードを変更しない限り，サーバへの不正ログインがいつでも可能になるのは，［ d ］である。

(3) 誤って不正なサーバに接続して通常のログイン操作を行った場合，誤接続したサーバ上で端末から送信された情報が盗まれる場合がある。この盗んだ情報をそのまま利用することによって，利用者がパスワードを変更しない限り，サーバへの不正ログインがいつでも可能になるのは，［ e ］である。

解答群

ア　方式1だけ　　イ　方式2だけ　　ウ　方式3だけ

エ　方式1，2だけ　　オ　方式1，3だけ　　カ　方式2，3だけ

キ　方式1，2，3すべて

問5　航空券発券システムに関する次の記述を読んで，設問1～3に答えよ。

オブジェクト指向分析／設計を用いて，航空券発券システムの設計を行う。

航空券発券業務の分析から，図1の分析クラス図を作成した。

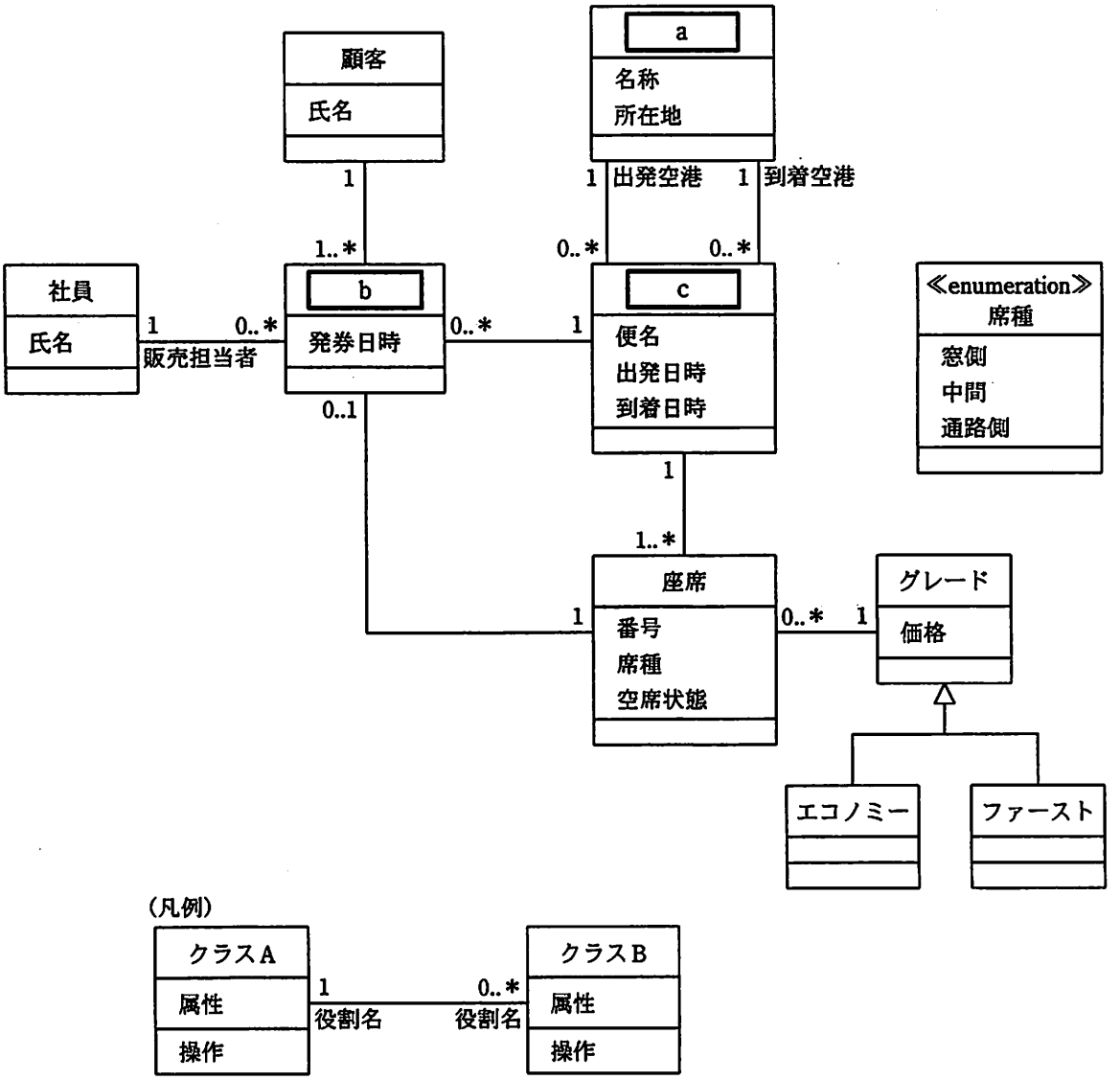
〔航空券発券業務の内容〕

(1) 航空会社の航空券販売担当者（以下，販売担当者という）は，顧客が窓口で申し込んだ内容を基に，航空券発券システムで空席確認及び発券を行う。

(2) 顧客が窓口で申し込む内容は，出発日時，出発地及び到着地となる空港名，便名，グレード（ファースト，エコノミー），人数，席種（窓側，中間，通路側）である。すべての便は直行便である。

(3) 販売担当者は(2)で受け付けた申込み内容を確認し，その情報をシステムに入力する。システムはその便の空席状態を確認する。空席があれば(4)に進み，なければ，顧客は申込み内容を変更して再度申込みをする。

(4) 販売担当者は顧客が希望しているグレードと席種の座席を確保し，顧客情報を登録して航空券を発券する。



(凡例)

長方形はクラスを表す。クラス間を結ぶ直線は，クラス間の関連を表す。
クラス間の多重度は直線の上又は左に示す。この凡例では，クラスAの1個のオブジェクトが，クラスBの0個以上のオブジェクトと関連することを表す。
各クラスの近くで多重度の反対側に役割名が書かれることがある。

図1　分析クラス図

設問1　図1中の［　　］に入れる適切なクラス名を，解答群の中から選べ。

解答群

ア　空港	イ　航空会社	ウ　航空機
エ　航空券	オ　航空券発券システム	カ　便

図1の分析クラス図に，実装を考慮して次の二つのクラスを追加した後，操作を洗い出すために，図2の販売担当者とシステムのオブジェクトとの関係のシーケンス図を作成した。

〔追加したクラス〕

① 航空券発券画面：データを入力する画面クラス

② 航空券発券管理：航空券を発券するための管理クラス

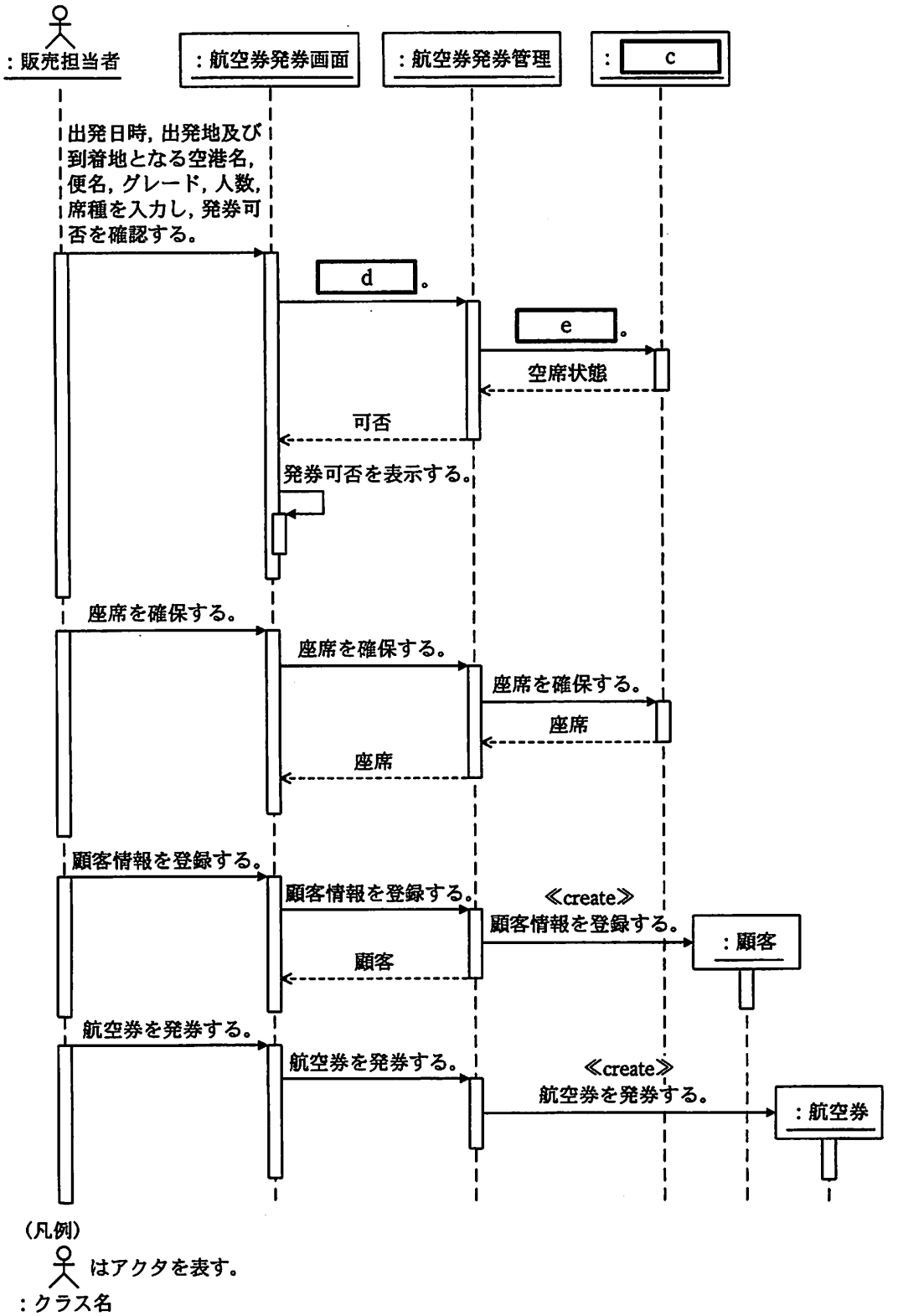


図２　販売担当者とシステムのオブジェクトとのシーケンス図

設問2　図2中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，図2中の［　c　］には設問1の正しい答えが入っているものとする。

d，eに関する解答群

ア　空席を確認する　　イ　航空券を発券する

ウ　顧客情報を登録する　　エ　出発日時を問い合わせる

オ　出発日時を登録する　　カ　発券可否を確認する

設問3　航空券発券画面クラスと航空券発券管理クラスを図3に示す。図3の操作中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，図3中の［　d　］には設問2の正しい答えが入っているものとする。

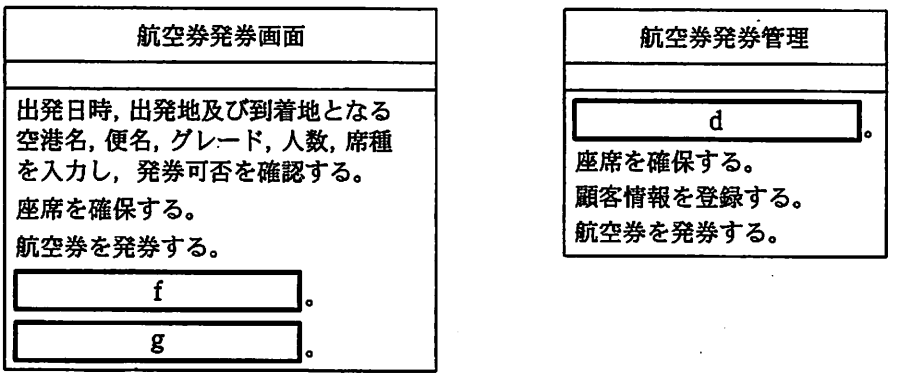


図3　航空券発券画面クラスと航空券発券管理クラス

f，gに関する解答群

ア　空席を確認する　　イ　顧客情報を登録する

ウ　出発日時を登録する　　エ　発券可否を確認する

オ　発券可否を表示する　　カ　便の座席数を確認する

問6　インシデント及び問題の管理に関する次の記述を読んで，設問に答えよ。

流通業のF社では，システム部が受注システムを運用している。このシステムは，F社とその関連会社が利用している。

ある日，朝から受注システムが使えないという状況が発生した。これを知った関連会社の社員がF社の担当窓口に問い合わせたところ，受注システムでアプリケーション障害が発生しており，それが関連会社に通報されていないことが分かった。

サービスの回復後にシステム部で通報ミスの原因を調べたところ，障害時の連絡先一覧表が古く，連絡先には関連会社が含まれていないことが分かった。システム部では，インシデント管理及び問題管理のプロセスが有効に機能しなかったことを反省し，リスク管理部の監査担当者の協力を得て，プロセス全体の見直しを実施した。

監査担当者は，見直しで発見された管理上の問題点とそれらに対する改善勧告を表1のようにまとめて，システム部長に報告した。

なお，F社ではインシデントの発生から問題の分析・解決までを，障害管理データベース（以下，DBという）で管理している。DBの項目は，次のとおりである。

① 障害管理番号（連番）　② 障害の発生日時
③ システム名　④ コンポーネント名
⑤ 障害の状況（文章で記述）　⑥ 解決の方法（文章で記述）
⑦ 解決担当者名　⑧ 解決日時

表1　見直しで発見された管理上の問題点とそれらに対する改善勧告

項番	管理上の問題点	改善勧告
1	障害発生時の関係部門への通報プロセスが有効に機能していない。	障害発生時の通報とフォローについて，体制とプロセスを見直すこと。
2	DB中に，インシデント対応は完了しているが，問題の解決が完了していないことを示す，解決日時が空欄のものが多数ある。	問題の解決を確認するプロセスを整備すること。
3	DBの現在の項目では不足がある。また，DBが十分に活用されていない。	DBの項目を見直し，不足項目を追加すること。また，DBの有効活用を図ること。
⋮	⋮	⋮

管理上の問題点は多数あったが，システム部では，重要と判断した項番1～3について早急に改善することにした。

設問　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。

(1) 通報とフォローのプロセスについては，表2の内容で改善することにした。

表2　通報とフォローのプロセスの改善内容

項番	改善勧告	改善内容（概要）
1	障害発生時の通報とフォローについて，体制とプロセスを見直すこと。	運用責任者が［ a ］を判断。 ・レベル3以上：役員と関連会社に通報し，社内Webに利用者向け状況を掲載。 （発生時，発生後1時間ごと） ・レベル2：システム部長が通報先を判断して通報。 （発生時，発生後適時） ・レベル1：システム部外への通報はしない。

DBに二つの項目を追加する。まず，障害時に最初に判断すべき項目として，項目［ a ］を設け，そのレベルを事前に定義しておく。障害発生時は，そのレベルに応じた通報を行う。また，項目［ b ］を設け，これまで重大障害時にホワイトボードなどに記録していた対応状況や回復状況の内容をDBに記録し，システム部員が状況を共有できるようにする。

(2) 解決日時が空欄の問題があることについては，表3の内容で改善することにした。

表3　解決日時が空欄の問題があることの改善内容

項番	改善勧告	改善内容（概要）
2	問題の解決を確認するプロセスを整備すること。	① 毎週の問題管理委員会でその週に解決予定の問題の解決状況を確認。 ② 長い間未解決のままの問題は毎月1回その処置を検討。

解決日時が空欄の問題を調査したところ，実際に未解決のものと解決日時の登録漏れとが混在していた。そこで，DBに項目［ c ］を設け，これを基準に問題の解決状況を問題管理委員会で毎週フォローする。問題管理委員会は実務担当者で構成し，決定事項をシステム部長に後日報告する。

なお，未解決の問題のうち，長期間残ってしまう［ d ］などは，月1回問題管理委員会で終了扱いとするかどうかを決定する。

また，解決日時の登録漏れの原因として，次のことが分かった。すなわち，解決方針が決まった問題は，問題管理を離れて，その解決のための作業を［ e ］のプロセスとして実施している。そのため，DB中の解決日時の更新を，つい忘れてしまう。そこで，問題の解決作業の場合は，該当する障害管理番号を［ e ］に引き継いで双方の管理が連動するよう，手続を変更する。

(3) DBの項目と使いやすさについては，表4の内容で改善することにした。

表4　DBの項目と使いやすさの改善内容

項番	改善勧告	改善内容（概要）
3	DBの項目を見直し，不足項目を追加すること。また，DBの有効活用を図ること。	① 今回は項番1，2の対応で設けた項目だけを追加。 ② システム管理者向けにDBの表示順序を変更。

不足項目について，今回は項番1，2の対応に必要な項目の追加にとどめる。

次に，現在のDBの内容の表示順序は，障害管理番号の降順で，重要な問題を見分けにくい。そこで，今回追加する項目を含めて表示の順序を見直し，システム管理者向けに，未解決の問題を重要なものから順に表示するため，［ f ］が空欄の問題を［ a ］の降順に並べて表示する機能を追加する。

a～c，fに関する解答群

ア　解決日時　　イ　解決方法の詳細（記述形式）

ウ　解決予定日時　　エ　障害対応の経緯（追記形式）

オ　障害の影響度　　カ　障害の発生日時

dに関する解答群

ア　原因が特定できず，その後再発しない問題

イ　システム部にスキルのある担当者がいないので，解決できない問題

ウ　放置しておいても，業務に大きな支障がない問題

エ　予算不足で，システム変更作業ができない問題

eに関する解答群

ア　キャパシティ管理　　イ　構成管理

ウ　サービスレベル管理　　エ　変更管理

問7　情報システムの効果見積りに関する次の記述を読んで，設問1，2に答えよ。

日用品メーカであるT社の売上高は年間4,000百万円，製品の製造に要する資材の調達費（以下，資材調達費という）は年間2,000百万円である。T社の企画課では，開発を予定している営業支援システム，資材調達システム，契約管理システムの効果を見積もることになった。各システムの概要を，表1に示す。

表1　各システムの概要

システム名	営業支援システム	資材調達システム	契約管理システム
システム概要	市場，顧客，自社商品などの情報を営業員に提供する。	製品製造に必要となる資材最適量を自動計算する。	契約書を一元管理し，その検索や維持更新を効率的に行う。
主管部署	営業課	資材課	法務課
開発費（百万円）	50	30	8
年間運用費（百万円）	2	2	0.4
効果見積り	販売量増加によって，年間の売上高を3％増加	年間の資材調達費を2％削減	法務課の年間労務費を5百万円削減

設問1　資材調達システムの投資回収期間に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

企画課では，資材調達システムについて，開発と運用のための投資を回収できるまでの期間（以下，投資回収期間という）を，年間の売上高が今後も一定であるとして計算した。資材調達システムの投資回収期間は，見積もった［ a ］までの期間を計算することで求められる。資材調達システムの投資回収期間は，［ b ］である。

aに関する解答群

ア　効果が開発費と運用費の合計を上回る

イ　効果が開発費と運用費の合計を下回る

ウ　効果と運用費の合計が開発費を上回る

エ　効果と運用費の合計が開発費を下回る

bに関する解答群

ア　1年未満　　イ　1年以上2年未満

ウ　2年以上3年未満　　エ　3年以上

設問2　各システムの期待効果に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

企画課では，各システムの効果見積りを，営業利益における期待効果（以下，営業利益効果という）として計算することにした。各システムの開発費は5年で定額償却する。また，減価償却費とシステムの運用費は，販売費及び一般管理費（以下，販管費という）に計上する。

T社の年間損益計算の抜粋を，表2に示す。

表2　T社の年間損益計算（抜粋）

単位　百万円

売上高	4,000	
売上原価	3,000	…製品の製造にかかる資材調達費，その他の関連費用
売上総利益	1,000	…売上高－売上原価
販管費	800	…資材の調達や製品の製造にかかった以外の費用
営業利益	200	…売上総利益－販管費

〔営業支援システムを開発した場合〕

営業支援システムの効果は売上高の増加であるが，売上高が増えると，それに伴って資材調達などにかかる売上原価も増える。さらに，営業支援システムの利用に伴って，減価償却費や運用費が発生する。営業支援システムの開発によって，売上高売上原価率（売上原価÷売上高）は変わらず，販管費は減価償却費10百万円と運用費2百万円を合わせて12百万円増えるので，営業利益効果は，［　c　］百万円となる。

〔資材調達システムを開発した場合〕

資材調達システムの効果見積りは資材調達費の２％削減なので，［ d ］を40百万円削減できる。資材調達システムの利用によって，販管費は減価償却費6百万円と運用費2百万円を合わせて8百万円増えるので，営業利益効果は32百万円となる。

〔契約管理システムを開発した場合〕

契約管理システムの利用によって，間接部門である法務課の労務費を5百万円削減できる。一方，減価償却費1.6百万円と運用費0.4百万円を合わせて2百万円が発生する。したがって，［ e ］となる。

〔営業支援システムと資材調達システムを開発し，同時に利用した場合〕

資材調達システムの利用によって［ f ］が期待できる。営業支援システムを開発した場合の営業利益効果は，売上高売上原価率が変わらないという前提で計算したが，［ f ］によって［ c ］百万円［ g ］営業利益効果となる。したがって，営業支援システムと資材調達システムを開発し同時に利用することで得られる営業利益効果は，それぞれの営業利益効果の合計［ g ］。

cに関する解答群

ア　18　　イ　30　　ウ　108　　エ　120

dに関する解答群

ア　売上原価　　イ　売上総利益　　ウ　売上高
エ　営業利益　　オ　販管費

eに関する解答群

ア　売上原価を3百万円削減でき，営業利益効果は3百万円
イ　売上原価を5百万円削減でき，営業利益効果は5百万円
ウ　販管費を3百万円削減でき，営業利益効果は3百万円
エ　販管費を5百万円削減でき，営業利益効果は5百万円

fに関する解答群

ア　売上高売上原価率の上昇　　イ　売上高売上原価率の低下

ウ　売上高の増加　　エ　売上高の減少

gに関する解答群

ア　と等しい　　イ　を上回る　　ウ　を下回る

次の問8は必須問題です。必ず解答してください。

問8　次のアルゴリズムの説明及びプログラムを読んで，設問に答えよ。

方程式の解の一つを求めるアルゴリズムである。任意に定めた解の予測値から始めて，計算を繰り返しながらその値を真の値に近づけていく。この方法は，ニュートン法と呼ばれる。

〔アルゴリズム1の説明〕

3次方程式 $a_3x^3+a_2x^2+a_1x+a_0=0$ の解の一つを，次の手順で求める。

(1) 解の予測値 x，係数 a_3，a_2，a_1，a_0 を読み込む。

(2) $3\times a_3$ の値を b_2 に，$2\times a_2$ の値を b_1 に，$1\times a_1$ の値を b_0 に，それぞれ求める。

(3) 次の①～④の処理を一定の回数繰り返す。

① $a_3x^3+a_2x^2+a_1x+a_0$ の値を求め，これを f とする。

② $b_2x^2+b_1x+b_0$ の値を求め，これを d とする。

③ x，f，d の値を印字する。

④ $x-\dfrac{f}{d}$ の値（解の一つにより近い値となる）を求め，これを新たな x とする。

プログラム1は，このアルゴリズム1を実装したものである。

〔アルゴリズム2の説明〕

アルゴリズム1を一般化して，n 次方程式 $a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0=0$ の解の一つを，次の手順で求める。

なお，方程式によっては解が求められない場合がある。

(1) 次数 n，解の予測値 x，係数 a_n，a_{n-1}，…，a_1，a_0 を読み込む。

(2) $n\times a_n$ の値を b_{n-1} に，$(n-1)\times a_{n-1}$ の値を b_{n-2} に，…，$2\times a_2$ の値を b_1 に，$1\times a_1$ の値を b_0 に，それぞれ求める。

(3) 次の①～④の処理を一定の回数繰り返す。

① $a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ の値を求め，これを f とする。

② $b_{n-1}x^{n-1}+b_{n-2}x^{n-2}+\cdots+b_1x+b_0$ の値を求め，これを d とする。

③ x，f，d の値を印字する。

④ $x-\dfrac{f}{d}$ の値（解の一つにより近い値となる）を求め，これを新たな x とする。

プログラム2は，このアルゴリズム2を実装したものである。

〔プログラム1〕

（行番号）

```
○主プログラム： プログラム１
○整数型： i
○実数型： d, f, x
○実数型： a3, a2, a1, a0, b2, b1, b0

・read(x, a3, a2, a1, a0)    /* x, a3～a0 の値を読み込む。 */
・b2 ← 3.0 × a3
・b1 ← 2.0 × a2
・b0 ← a1
■ i: 1, i ≦ 100, 1          /* 繰返し回数は 100 回とする。 */
│  ・f ← ((a3 × x + a2) × x + a1) × x + a0
│  ・d ←  (b2 × x + b1) × x + b0
│  ・print(x, f, d)          /* x, f, d の値を印字する。 */
│  ・x ← x － f ÷ d
■
/* プログラム１の終わり */
```

〔プログラム2〕

（行番号）

```
○主プログラム： プログラム２
○整数型： i, k, n           /* 1 ≦ n ≦ 9 とする。 */
○実数型： d, f, x
○実数型： a[10], b[10]      /* 添字は 0 から始まる。 */

・read(n, x)                 /* n, x の値を読み込む。 */
■ k: n, k ≧ 0, −1           /* k を n, n−1, …, 0 として繰り返し, */
│  ・read(a[k])              /*  係数 a[k]の値を順に読み込む。 */
■
┌----------------------------┐
|       手順(2)の処理         |
└----------------------------┘
■ i: 1, i ≦ 100, 1          /* 繰返し回数は 100 回とする。 */
│  ┌----------------------------┐
│  |                            |
│  |  手順(3)の①と②の処理     |
│  |                            |
│  └----------------------------┘
│  ・print(x, f, d)          /* x, f, d の値を印字する。 */
│  ・x ← x － f ÷ d
■
/* プログラム２の終わり */
```

設問　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。

(1) 解の予測値 $x = 2.5$，係数 $a_3 = 1$，$a_2 = -3$，$a_1 = -1$，$a_0 = 3$ を与えて，3次方程式 $x^3 - 3x^2 - x + 3 = 0$ の解の一つを求める（解は3，1，−1）。プログラム1をある処理系で実行した結果，図1に示すとおり解の一つである $x = 3$ が近似的に得られた。

(行番号)	x	f	d
1	2.500000	−2.625000	2.750000
2	3.454545	4.969947	14.07438
3	3.101425	8.741682(−1)	9.247965
4	3.006900	5.548452(−2)	8.082941
5	3.000035	2.833717(−4)	8.000425
6	3.000000	7.527369(−9)	8.000000
7	3.000000	0.000000	8.000000
8	3.000000	0.000000	8.000000

注1　数値の後の$(-k)$は，$\times 10^{-k}$ を示す。例えば，5.548452(−2)は，5.548452×10^{-2}，すなわち0.05548452を表す。

注2　表示は有効数字7けた（8けた目を四捨五入）

図1　プログラム1の印字結果

この印字結果の行番号6，7の x の値（網掛けの部分）はいずれも3.000000である。行番号6，7を印字した時点で変数 x に保持されていた実際の値をそれぞれ x_6，x_7 で表すと，［　a　］。

なお，この処理系では，実数型は2進数の浮動小数点形式であって，有効けた数は10進数で十数けた程度であることが分かっている。

(2) プログラム2では，係数 a_k（k：n，$n-1$，…，1，0）の値を配列aの要素a[k]に，b_k（k：$n-1$，$n-2$，…，1，0）の値を配列bの要素b[k]に，それぞれ図2のように格納している。

要素番号	0	1	2	…	n−1	n	…
配列a	a_0	a_1	a_2	…	a_{n-1}	a_n	…

要素番号	0	1	…	n−3	n−2	n−1	…
配列b	$1 \times a_1$	$2 \times a_2$	…	$(n-2) \times a_{n-2}$	$(n-1) \times a_{n-1}$	$n \times a_n$	…

図2　係数 a_k，b_k の値の格納

プログラム2の行番号9～11は，アルゴリズム2の手順(2)の処理である。この部分のプログラムは，次のようになる。

〔プログラム2の一部〕

(行番号)

```
 9  ■ k: n, k ≧ 1, −1   /* k を n, n−1, …, 1 として繰り返す。 */
10  |   ・[    b    ]
11  ■
```

また，行番号13～18は，アルゴリズム2の手順(3)の①と②の処理である。プログラム1では，例えばfの値$a_3x^3+a_2x^2+a_1x+a_0$を求める式を，

```
f ← ((a3 × x + a2) × x + a1) × x + a0
```

と変形して，演算回数を減らす工夫をしている。この部分にも同様の工夫をすると，プログラムは次のようになる。

〔プログラム2の一部〕

(行番号)

```
13  ・f ← a[n] × x + a[n−1]
14  ・d ← [    c    ]
15  ■ k: n−2, k ≧ 0, −1  /* k を n−2, n−3, …, 0 として繰り返す。*/
16  |   ・f ← f × x + a[k]
17  |   ・d ← d × x + [    d    ]
18  ■
```

(3) 次数$n=4$，係数$a_4=1$，$a_3=-8$，$a_2=24$，$a_1=-32$，$a_0=16$として，4次方程式$x^4-8x^3+24x^2-32x+16=0$の解を求める（4個の解がすべて2）。解の予測値を$x=2.00001$として，ある処理系でプログラム2を実行したところ，図3に示すとおりの印字結果となった。

(行番号)	x	f	d
1	2.000010	−3.552714(−15)	3.996803(−15)
2	2.888899	6.243232(−1)	2.809423
3	2.666674	1.975398(−1)	1.185225
4	2.500006	6.250281(−2)	5.000169(−1)
5	2.375004	1.977628(−2)	2.109446(−1)

図3　プログラム2の印字結果

この印字結果の行番号2では，x の値（網掛けの部分）が解である2から遠ざかってしまっている。その原因を調べるため，f を求める式に実際の数値を当てはめて，

$$\underbrace{(((1.0\times 2.00001-8.0)\times 2.00001+24.0)\times 2.00001-32.0)\times 2.00001}_{\text{(A)}}+16.0$$

として，(A) の部分の中間結果を印字するプログラムを作り，同じ処理系で実行した。印字結果は −16.00000 であり，正確な値 −15.99999999999999999 と有効数字7けたで一致した。しかし，行番号1で印字された f の値は，正確な値である 10^{-20}（印字の表記では 1.000000(−20)）とは異なっている。

これらのことから判断して，(A) の部分では演算の過程で［　e　］が徐々に累積し，(A) の計算結果に 16.0 を加算するときに，けた落ちが発生したと考えられる。

aに関する解答群

ア　$x_6=x_7$ である　　イ　$x_6\neq x_7$ である

ウ　$x_6=x_7$ とも $x_6\neq x_7$ ともいえない

bに関する解答群

ア　b[k−1] ← (k−1) × a[k]　　イ　b[k−1] ← k × a[k]

ウ　b[k] ← k × a[k+1]　　エ　b[k] ← (k+1) × a[k+1]

c, dに関する解答群

ア　b[k−1]　　イ　b[k]

ウ　b[k+1]　　エ　b[n−1]

オ　b[n−1] × x　　カ　b[n−1] × x + b[n−2]

eに関する解答群

ア　けたあふれ　　イ　けた落ち

ウ　指数下位けたあふれ　　エ　丸め誤差

次の問9から問13までの5問については，この中から1問を選択し，答案用紙の選択欄の(選)をマークして解答してください。

なお，2問以上選択した場合には，はじめの1問について採点します。

問9　次のCプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

C言語では整数型の変数に格納できる値には上限がある。これを超える正の整数（以下，正の多倍長整数という）の入出力と加算を行うプログラムである。

(1) 正の多倍長整数は，次に示すMP型の構造体を用いて表現する。

```
typedef struct{
          int length;
          long data[ARRAY_MAX];
        }MP;
```

正の多倍長整数を，下位から9けたずつに切り分けて，構造体MPのメンバである配列dataの要素番号の小さい方から順番に値を格納する。例えば，462840598274638592012838441571340076529187231476 41という整数の場合，図のとおりになる。

46284|059827463|859201283|844157134|007652918|723147641

要素番号	5	4	3	2	1	0
配列data	46284	59827463	859201283	844157134	7652918	723147641

図　正の多倍長整数の格納例

構造体のメンバlengthには，実際に値を格納した要素数を入れる。図の場合は6である。

(2) 関数の仕様は，次のとおりである。

```
void set(MP *num, const char str[]);
```

引数：num　MP型の構造体で表現された多倍長整数

str　文字列で表現された多倍長整数であって，1～9の数字で始まる。

機能：文字列で与えられた多倍長整数strを変換して，MP型の構造体numに格納する。numのメンバdataは，変換後の数値を格納するのに十分な要素数が確保されているものとする。

返却値：なし

```
void print(const MP *num);
```

引数：num　MP型の構造体で表現された多倍長整数

機能：多倍長整数を出力する。

返却値：なし

```
void add(const MP *a, const MP *b, MP *c);
```

引数：a，b，c　MP型の構造体で表現された多倍長整数であり，cは，a及びbではないものとする。

機能：二つの多倍長整数a，bの和を多倍長整数cに格納する。cのメンバdataは，加算処理を行うのに十分な要素数が確保されているものとする。

返却値：なし

(3) 次のライブラリ関数を用いる。

```
size_t strlen(const char *s);
```

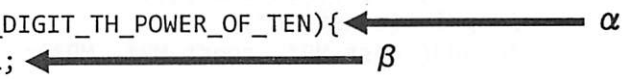
機能：sが指す文字列の長さを計算する。

返却値：終端を示すナル文字に先行する文字の個数を返す。

〔プログラム〕

```
#include <stdio.h>
#include <string.h>

#define ARRAY_MAX 100
#define NUM_DIGIT 9
#define NUM_DIGIT_TH_POWER_OF_TEN 1000000000

typedef struct{
           int length;
           long data[ARRAY_MAX];
        }MP;

void set(MP*, const char[]);
void print(const MP*);
void add(const MP*, const MP*, MP*);

/* 文字列から多倍長整数を扱う構造体に変換 */
void set(MP *num, const char str[]){
   int str_idx = strlen(str) - 1;
   int num_idx = 0;
   int i;
   long mul;

   while( a ){
      num->data[num_idx] = 0;
      mul = 1;
      for(i = 0; b ; i++){
         num->data[num_idx] += mul * (str[str_idx--] - '0');
         mul  c ;
      }
      num_idx++;
   }
   num->length = num_idx;
}

/* 多倍長整数の出力 */
void print(const MP *num){
   int i;

   printf("%ld", num->data[num->length - 1]);
   for( d ){
      /* ゼロ詰めして必ず9けたを表示する。 */
      printf("%09ld", num->data[i]);
   }
   printf("\n");
}
```

```
/* 二つの多倍長整数の加算 */
void add(const MP *a, const MP *b, MP *c){
   int i;
   int i_max;

   if(a->length > b->length){
      i_max = a->length;
   }else{
      i_max = b->length;
   }
   c->data[0] = 0;

   for(i = 0; i < i_max; i++){
      if(i < a->length) c->data[i] += a->data[i];
      if(i < b->length) c->data[i] += b->data[i];

      if(c->data[i] >= NUM_DIGIT_TH_POWER_OF_TEN){
         c->data[i + 1] = 1;
         c->data[i] -= NUM_DIGIT_TH_POWER_OF_TEN;
      }else{
         c->data[i + 1] = 0;
      }
   }
   if(c->data[i] == 0){
      c->length = i;
   }else{
      c->length = i + 1;
   }
}
```



設問1　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。

aに関する解答群

ア　str_idx != 0　　イ　str_idx < 0　　ウ　str_idx <= 0

エ　str_idx > 0　　オ　str_idx >= 0

bに関する解答群

ア　i < NUM_DIGIT

イ　i < NUM_DIGIT && str_idx <= 0

ウ　i < NUM_DIGIT && str_idx >= 0

エ　i < NUM_DIGIT || str_idx <= 0

オ　i < NUM_DIGIT || str_idx >= 0

cに関する解答群

ア %= 10　　イ *= 10　　ウ += 10　　エ -= 10　　オ /= 10

dに関する解答群

ア i = 0; i <= num->length - 1; i++

イ i = 0; i <= num->length - 2; i++

ウ i = num->length - 1; i >= 0; i--

エ i = num->length - 2; i >= 0; i--

設問2　プログラムの動作について，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。

関数 add を用いて以下の (1) に示す二つの多倍長整数を加算する場合，プログラム中の α 部分は［ e ］回実行され，β 部分は［ f ］回実行される。また，以下の (2) に示す二つの多倍長整数を加算する場合，β 部分は［ g ］回実行される。

(1) 231456813|123456789|234567890|478617283

999999999|123456789|009283741

(2) 85923|726438423|734239982|017238764|500000000|321987472

5472937|726325476|292278313|499999999|783917811

e～gに関する解答群

ア 0　　イ 1　　ウ 2　　エ 3

オ 4　　カ 5　　キ 6　　ク 7

問10　次のCOBOLプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

売上伝票ファイルに記録された各支店（A～E）の月単位の売上情報を支店ごとに集計して，売上マスタファイルに格納する月次処理のプログラムである。

なお，集計した当該月の売上データは画面にも表示する。

(1) 売上伝票ファイルは，図1に示すレコード様式の順ファイルである。

伝票番号 6けた	売上日 8けた	支店コード 2けた	売上金額 6けた

図1　売上伝票ファイルのレコード様式

① 売上伝票ファイルには，ある月の1か月分の売上データを格納する。

② 伝票番号には，6けたの数字を格納する。

③ 売上日には，西暦年を4けた，月，日をそれぞれ2けたの数字で格納する。

なお，一つのファイルに含まれるレコードの西暦年と月は，それぞれすべて等しい。

④ 支店コードは，A～Eの支店に対して昇順に割り当てた1～5の数字である。

(2) 売上マスタファイルは，図2に示すレコード様式の，売上年月を主キーとする索引ファイルである。

売上年月 6けた	売上金額					売上合計金額 9けた
	A支店 8けた	B支店 8けた	C支店 8けた	D支店 8けた	E支店 8けた	

図2　売上マスタファイルのレコード様式

① 売上年月には，西暦年が4けた，月が2けたの数字を格納する。

② 売上金額には，当該月の各支店の売上金額を格納する。

③ 売上合計金額には，当該月の全支店の売上金額の合計を格納する。

④ 各支店の1か月分の売上金額は，8けたを超えないものとする。

(3) 集計結果の表示例を図3に示す。

```
C-BRANCH     32,100,000
A-BRANCH     23,000,000
D-BRANCH     18,700,000
B-BRANCH     10,200,000
E-BRANCH      9,800,000
----------------------
TOTAL        93,800,000
```

図3　集計結果の表示例

① 当該月の支店別売上金額を降順に整列して表示する。

② TOTAL には，売上合計金額を表示する。

〔プログラム〕

(行番号)

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  SD  SORT-FILE.
 4  01  SORT-REC.
 5      02  SORT-BR              PIC 9(2).
 6      02  SORT-AMOUNT          PIC 9(8).
 7  FD  SALES-FILE.
 8  01  SALES-REC.
 9      02  SALES-NO             PIC 9(6).
10      02  SALES-DATE.
11          03  SALES-YYYYMM     PIC 9(6).
12          03  SALES-DD         PIC 9(2).
13      02  SALES-BR             PIC 9(2).
14      02  SALES-AMOUNT         PIC 9(6).
15  FD  MAST-FILE.
16  01  MAST-REC.
17      02  MAST-KEY.
18          03  KEY-YYYYMM       PIC 9(6).
19      02                        PIC X(49).
20  WORKING-STORAGE SECTION.
21  01  W-MAST.
22      02  MAST-YYYYMM          PIC 9(6).
23      02  MAST-BRANCH.
24          03  MAST-AMOUNT      PIC 9(8) OCCURS 5.
25          03  MAST-TOTAL       PIC 9(9).
26  77  CNT                      PIC 9(2).
27  77  AMOUNT                   PIC ZZZ,ZZZ,ZZ9.
28  77  READ-FLAG                PIC X(1)  VALUE SPACE.
29      88  DATA-EOF             VALUE "E".
30  77  SORT-FLAG                PIC X(1)  VALUE SPACE.
31      88  SORT-EOF             VALUE "E".
```

```
01  BRANCH                 PIC X(55) VALUE
      "A-BRANCH   B-BRANCH   C-BRANCH   D-BRANCH   E-BRANCH".
01  HEADER REDEFINES BRANCH.
    02  BR-NAME            PIC X(11) OCCURS 5.
PROCEDURE DIVISION.
MAIN-PROC.
    INITIALIZE W-MAST.
    OPEN INPUT SALES-FILE I-O MAST-FILE.
    READ SALES-FILE
         AT END     SET DATA-EOF TO TRUE
         NOT AT END [          a          ]
                    MOVE SALES-AMOUNT TO MAST-AMOUNT(SALES-BR)
    END-READ.
    PERFORM UNTIL DATA-EOF
       READ SALES-FILE
            AT END     SET DATA-EOF TO TRUE
            NOT AT END ADD SALES-AMOUNT TO MAST-AMOUNT(SALES-BR)
       END-READ
    END-PERFORM.
    PERFORM VARYING CNT FROM 1 BY 1 UNTIL CNT > 5
       [          b          ]
    END-PERFORM.
    WRITE MAST-REC FROM W-MAST
       INVALID KEY    DISPLAY "** ERROR OCCURRED **"
       NOT INVALID KEY PERFORM PRT-PROC
    END-WRITE.
    CLOSE SALES-FILE MAST-FILE.
    STOP RUN.
PRT-PROC.
    SORT SORT-FILE [          c          ]
                   INPUT  PROCEDURE IS IN-PROC
                   OUTPUT PROCEDURE IS OUT-PROC.
IN-PROC.
    PERFORM VARYING CNT FROM 1 BY 1 UNTIL CNT > 5
       MOVE CNT TO SORT-BR
       [          d          ]
       RELEASE SORT-REC
    END-PERFORM.
OUT-PROC.
    PERFORM UNTIL SORT-EOF
       RETURN SORT-FILE
          AT END     SET SORT-EOF TO TRUE
          NOT AT END MOVE SORT-AMOUNT TO AMOUNT
                     DISPLAY BR-NAME(SORT-BR) AMOUNT
       END-RETURN
    END-PERFORM.
    DISPLAY "----------------------".
    MOVE MAST-TOTAL TO AMOUNT.
    DISPLAY "TOTAL      " AMOUNT.
```

設問1　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。

a，b，dに関する解答群

ア　ADD MAST-AMOUNT(CNT) TO MAST-TOTAL

イ　ADD SALES-AMOUNT TO MAST-TOTAL

ウ　ADD SALES-AMOUNT TO SORT-AMOUNT

エ　MOVE MAST-AMOUNT(CNT) TO SORT-AMOUNT

オ　MOVE SALES-AMOUNT TO SORT-AMOUNT

カ　MOVE SALES-BR TO SORT-BR

キ　MOVE SALES-YYYYMM TO MAST-YYYYMM

cに関する解答群

ア　ASCENDING KEY SORT-AMOUNT

イ　ASCENDING KEY SORT-BR

ウ　DESCENDING KEY SORT-AMOUNT

エ　DESCENDING KEY SORT-BR

設問2　各支店の当該月の売上金額に加えて対前年同月比を表示するようにプログラムを変更する。このとき，前年同月の売上金額がゼロであるか又は情報の取得に失敗した場合は，“(- %)”と表示する。集計結果の表示例を，図4に示す。次の表中の［　　］に入れる正しい答えを，解答群の中から選べ。

なお，対前年同月比は，999％を超えないものとする。

```
C-BRANCH     32,100,000 (112%)
A-BRANCH     23,000,000 ( 96%)
D-BRANCH     18,700,000 (100%)
B-BRANCH     10,200,000 ( 98%)
E-BRANCH      9,800,000 ( 92%)
-----------------------------
TOTAL        93,800,000 (102%)
```

図4　対前年同月比を追加した集計結果の表示例

表　プログラムの変更内容

処置	変更内容
行番号27と28の間に追加	77　THIS-TOTAL　　PIC 9(9). 77　RATIO　　PIC ZZ9.

行番号71～80を変更	e .
	COMPUTE KEY-YYYYMM = MAST-YYYYMM - 100.
	READ MAST-FILE INVALID KEY　　INITIALIZE W-MAST
	NOT INVALID KEY MOVE MAST-REC TO W-MAST.
	PERFORM UNTIL SORT-EOF
	RETURN SORT-FILE
	AT END SET SORT-EOF TO TRUE
	NOT AT END
	MOVE SORT-AMOUNT TO AMOUNT
	IF MAST-AMOUNT(SORT-BR) = ZERO THEN
	DISPLAY BR-NAME(SORT-BR) AMOUNT " (- %)"
	ELSE
	f
	DISPLAY BR-NAME(SORT-BR) AMOUNT " (" RATIO "%)"
	END-IF
	END-RETURN
	END-PERFORM.
	DISPLAY "----------".
	MOVE THIS-TOTAL TO AMOUNT.
	IF MAST-TOTAL = ZERO THEN
	DISPLAY "TOTAL　　" AMOUNT " (- %)"
	ELSE
	g
	DISPLAY "TOTAL　　" AMOUNT " (" RATIO "%)"
	END-IF.

eに関する解答群

ア　MOVE MAST-AMOUNT(CNT) TO THIS-TOTAL

イ　MOVE MAST-REC TO W-MAST

ウ　MOVE MAST-TOTAL TO THIS-TOTAL

エ　MOVE W-MAST TO MAST-REC

f, gに関する解答群

ア　COMPUTE RATIO = MAST-AMOUNT(SORT-BR) * 100 / SORT-AMOUNT

イ　COMPUTE RATIO = MAST-TOTAL * 100 / THIS-TOTAL

ウ　COMPUTE RATIO = SORT-AMOUNT * 100 / MAST-AMOUNT(SORT-BR)

エ　COMPUTE RATIO = SORT-AMOUNT * 100 / MAST-TOTAL

オ　COMPUTE RATIO = THIS-TOTAL * 100 / MAST-TOTAL

問11　次のJavaプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

携帯電話の利用状況に対して，最も安価な料金プラン（割引サービスを含む）を提示するプログラムである。1か月の料金は，基本料金，通話料金及びパケット料金からなる。パケット料金にはオプションの割引サービスがある。

表1に料金プラン，表2にパケット料金の割引サービス，表3にテスト用の利用状況を示す。

なお，パケット料金を計算するとき，パケット数は100パケット単位に切り上げる。

表1　料金プラン

単位　円

プラン名	基本料金	無料通話分(1)	1分当たりの通話料金	100パケット当たりのパケット料金
A	2,000	1,000	40	20
B	3,000	2,000	30	20

注(1)　この料金分までの通話に対する料金は基本料金に含まれている。

表2　パケット料金の割引サービス

サービス名	内容
S1	サービス料金1,000円を支払うと，10,000パケットまでは無料，10,000パケットを超えた分は，100パケット当たり8円とする。

表3　テスト用の利用状況

通話時間（分）	パケット数
30	1,000
30	10,000
300	1,000
300	10,000

(1) クラス`CellPhonePlan`は，料金プランを表す。メソッド`calculateCharge`は，引数で渡された利用状況に対する料金を返す。

(2) クラス CallingPlan は，通話料金を計算するクラスである。メソッド calculateCharge は，引数で渡された通話時間（分単位に切上げ）に対する料金を返す。

(3) インタフェース PacketPlan は，パケット料金を計算するためのインタフェースである。メソッド calculateCharge は，引数で渡されたパケット数に対する料金を返す。

(4) クラス Measured は，従量制のパケット料金（割引サービスなしの場合）を表す。

(5) クラス Tiered は，パケット料金の割引サービス S1 を表す。

(6) クラス CellPhonePlanner は，利用状況に対して，最も安価な料金プラン（割引サービスを含む）を提示する。メソッド getRecommendedPlan は，与えられた利用状況に対して，最も安価な料金プラン（割引サービスを含む）を返す。メソッド main は，テスト用のメインプログラムである。

プログラムの実行結果を図に示す。

```
minutes:   30, packets:   1000 => A      ( 2400)
minutes:   30, packets:  10000 => A S1   ( 3200)
minutes:  300, packets:   1000 => B      (10200)
minutes:  300, packets:  10000 => B S1   (11000)
```

図　プログラムの実行結果

〔プログラム1〕

```
public class CellPhonePlan {
   private CallingPlan callingPlan;
   private PacketPlan packetPlan;

   CellPhonePlan(CallingPlan callingPlan, PacketPlan packetPlan) {
      this.callingPlan = callingPlan;
      this.packetPlan = packetPlan;
   }
   public int calculateCharge(int minutes, int packtes) {
      return callingPlan.calculateCharge(minutes)
         + packetPlan.calculateCharge(packtes);
   }
   public String getName() {
      return callingPlan.getName() + " " + packetPlan.getName();
   }
}
```

〔プログラム2〕

```
public class CallingPlan {
   private String name;
   private int basicCharge;
   private int included;
   private int callingRate;
   CallingPlan(String name, int basicCharge,
               int included, int callingRate) {
      this.name = name;
      this.basicCharge = basicCharge;
      this.included = included;
      this.callingRate = callingRate;
   }
   public String getName() { return name; }
   public int calculateCharge(int minutes) {
      int callingCharge = [ a ];
      if ([ b ])
         callingCharge = 0;
      return basicCharge + callingCharge;
   }
}
```

〔プログラム3〕

```
public interface PacketPlan {
   String getName();
   int calculateCharge(int packets);
}
```

〔プログラム4〕

```
public class Measured [ c ] {
   private int packetRate;

   Measured(int packetRate) {
      this.packetRate = packetRate;
   }
   public String getName() { return ""; }
   public int calculateCharge(int packets) {
      return packetRate * ((packets + 99) / 100);
   }
}
```

〔プログラム5〕

```
public class Tiered [ c ] {
   private String name;
   private int basicCharge;
   private int allowance;
   private int packetRate;

   Tiered(String name, int basicCharge,
          int allowance, int packetRate) {
      this.name = name;
      this.basicCharge = basicCharge;
      this.allowance = allowance;
      this.packetRate = packetRate;
   }
   public String getName() { return name; }
   public int calculateCharge(int packets) {
      int charge = basicCharge;
      if (packets > allowance)
         charge += packetRate * ([ d ]) / 100);
      return charge;
   }
}
```

〔プログラム6〕

```
public class CellPhonePlanner {
   static final CallingPlan[] callingPlans = {
      new CallingPlan("A", 2000, 1000, 40),
      new CallingPlan("B", 3000, 2000, 30) };
   static final PacketPlan[] packetPlans = {
      new Measured(20),
      new Tiered("S1", 1000, 10000, 8) };
   static final CellPhonePlan[] cellPhonePlans = {
      new CellPhonePlan(callingPlans[0], packetPlans[0]),
      new CellPhonePlan(callingPlans[0], packetPlans[1]),
      new CellPhonePlan(callingPlans[1], packetPlans[0]),
      new CellPhonePlan(callingPlans[1], packetPlans[1]) };

   static CellPhonePlan getRecommendedPlan(int minutes,
                                           int packets) {
      int minCharge = [ e ];
      CellPhonePlan recommended = null;
      for (CellPhonePlan cellPhonePlan : cellPhonePlans) {
         int charge = [ f ];
         if (charge <= minCharge) {
            recommended = cellPhonePlan;
            minCharge = charge;
         }
      }
```

```
        return recommended;
    }

    public static void main(String[] args) {
        int[][] testData = {{30, 1000}, {30, 10000},
                            {300, 1000}, {300, 10000} };

        for (int[] data : testData) {
            int minutes = data[0];
            int packets = data[1];
            CellPhonePlan recommended =
                                getRecommendedPlan(minutes, packets);
            System.out.printf("minutes: %4d, packets: %6d => %-6s (%5d)%n",
                              minutes, packets, recommended.getName(),
                              recommended.calculateCharge(minutes,
                                                          packets));
        }
    }
}
```

設問1　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。

aに関する解答群

ア　callingRate * minutes

イ　callingRate * minutes - included

ウ　included

エ　included - callingRate * minutes

bに関する解答群

ア　callingCharge < 0　　イ　callingCharge < included

ウ　callingCharge > 0　　エ　callingCharge > included

cに関する解答群

ア　extends CellPhonePlan　　イ　extends PacketPlan

ウ　implements CellPhonePlan　　エ　implements PacketPlan

dに関する解答群

ア　allowance + 99　　イ　packets + 99

ウ　packets + allowance + 99　　エ　packets - allowance + 99

e，fに関する解答群

ア　0

イ　Integer.MIN_VALUE

ウ　Integer.MAX_VALUE

エ　cellPhonePlan.calculateCharge(minutes, packets)

オ　recommended.calculateCharge(minutes, packets)

設問2　パケット通信を大量に行う人を対象として，S1の内容を変更したパケット料金割引サービスS2を追加することにした。表4にサービスの内容，プログラム7にS2を表すクラスを示す。プログラム7中の［　　］に入れる正しい答えを，解答群の中から選べ。

表4　新しく追加するパケット料金の割引サービス

サービス名	内容
S2	サービス料金1,000円を支払うと，10,000パケットまでは無料， 10,000パケットを超えて50,000パケットまでは，100パケット当たり10円， 50,000パケット（合計5,000円）を超えた場合は，一律5,000円とする。

〔プログラム7〕

```
public class TieredBounded [ g ] {
   private int maxCharge;

   TieredBounded(String name, int basicCharge, int allowance,
                 int packetRate, int maxCharge) {
      super(name, basicCharge, allowance, packetRate);
      this.maxCharge = maxCharge;
   }
   public int calculateCharge(int packets) {
      int charge = [ h ];
      if (charge > [ i ])
         charge = maxCharge;
      return charge;
   }
}
```

gに関する解答群

ア extends PacketPlan　　イ extends Tiered

ウ implements PacketPlan　　エ implements Tiered

h, iに関する解答群

ア 0　　イ allowance

ウ basicCharge　　エ basicCharge + maxCharge

オ maxCharge　　カ super.calculateCharge(packets)

キ this.calculateCharge(packets)

問12　次のアセンブラプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

連続したn語を16×nビットのビット列とみなし，ビット列Aとする。ビット列Aの(p＋1)ビット目からのqビットを，別のqビットのビット列Bで置き換える副プログラムREPLACEである。置換えの概要を図1に示す。

ここで，p≧0，1≦q≦16，p＋q≦16×nとする。

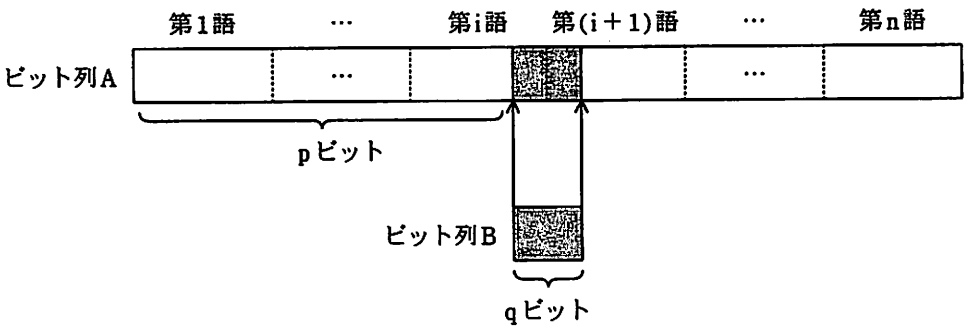


図1　置換えの概要

(1) ビット列Aの先頭アドレスは，GR1に設定されて主プログラムから渡される。

(2) ビット列BはGR0に左詰めで設定され，GR0の残りの部分は0で埋められて主プログラムから渡される。ビット列BとGR0の関係を図2に示す。

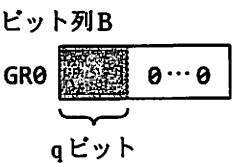


図2　ビット列BとGR0の関係

(3) 値pはGR2に，値qはGR3に設定されて主プログラムから渡される。

(4) 副プログラムから戻るとき，汎用レジスタGR1～GR7の内容は元に戻す。

〔プログラム〕

（行番号）

```
 1  REPLACE START
 2          RPUSH
 3          LD     GR4,GR2      ; GR4←p
 4          SRL    GR4,4        ; GR4←p/16
 5          ADDA   GR1,GR4      ; GR1を置換え対象語(第i語)に位置付ける。
 6          AND    GR2,=#000F
 7          LD     GR4,=16
 8          SUBA   GR4,GR2
 9          LD     GR5,GR0
10          LD     GR6,=#8000
11          SUBA   GR3,=1
12          SRA    GR6,0,GR3
13          LD     GR7,GR6
14          SRL    GR0,0,GR2
15          SRL    GR6,0,GR2
16          SLL    GR5,0,GR4
17          SLL    GR7,0,GR4
18          LD     GR2,0,GR1
19          [e]    GR6,GR2      ;┐ 第i語のうち
20          [f]    GR2,GR6      ;┘ ビット列Bを入れる部分を0にする。
21          OR     GR2,GR0
22          ST     GR2,0,GR1
23          LD     GR2,1,GR1
24          [e]    GR7,GR2      ;┐ 第(i+1)語のうち
25          [f]    GR2,GR7      ;┘ ビット列Bを入れる部分を0にする。
26          OR     GR2,GR5
27          ST     GR2,1,GR1
28          RPOP
29          RET
30          END
```

設問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。

主プログラムから渡されたp，qの値及びGR0の内容は，次のとおりであった。

```
p：   55
q：   12
GR0： 1011000111010000
```

行番号8のSUBAの実行直後におけるGR2の値は［ a ］であり，GR4の値は［ b ］である。

行番号17のSLLの実行直後におけるGR0の内容は［ c ］であり，GR5の内容は［ d ］である。

a，bに関する解答群

ア　1	イ　5	ウ　7
エ　9	オ　11	カ　15

c，dに関する解答群

ア　0000000001011000	イ　0000000101100011
ウ　1010000000000000	エ　1110100000000000

設問2　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。

解答群

ア　ADDL	イ　AND	ウ　LD	エ　OR
オ　SLL	カ　SRL	キ　XOR	

問 13　次の表計算及びワークシートの説明を読んで，設問 1，2 に答えよ。

〔表計算の説明〕

ある企業では，アルバイトの勤務を管理するための勤怠管理表と出勤割当表の作成を手作業で行っていたが，これを表計算ソフトで行うことにした。

〔ワークシート：勤怠管理〕

(1) 表計算ソフトのワークシート“勤怠管理”を作成した。その例を図 1 に示す。

	A	B	C	D	E	F	G
1	ID	0004		氏名	情報太郎		
2							
3	日付	出社予定	出社時刻	退社時刻	勤務時間	勤怠理由	ペナルティ
4	2009-09-01	1	10:00	17:15	6:15	0	0
5	2009-09-02	0			0:00	0	0
6	2009-09-03	1	10:00	12:30	2:00	0	1
7	2009-09-04	1			0:00	0	3
8	2009-09-05	1	12:15	17:00	4:00	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
33	2009-09-30	1	9:30	17:00	6:30	0	0
34	2009-09-31	0			0:00	0	0
35				合計	106:20		6

図 1　ワークシート“勤怠管理”の例

(2) 時刻及び時間の表記は h:mm，hh:mm 又は hhh:mm で表示されるが，内部では 24 時間を 1 とする実数値で記録されている。例えば，10:30 は 10.5／24 ＝ 0.4375，18:00 は 18／24 ＝ 0.75 である。計算にはこの数値を利用する。

(3) 図 1 中の各項目の説明は，次のとおりである。

ID，氏名：アルバイトの識別番号と氏名である。

日付：　1 か月分の日付が yyyy-mm-dd の形式で表示される。日数 30 日以下の月でも 31 日まで列挙する。

出社予定：出社予定日には 1 を，出社予定でない日や 2009-09-31 のように存在しない日には 0 を格納する。

出社時刻：始業時刻は10時である。10時に遅れた場合は遅刻扱いとなる。出社しなかった日，又は存在しない日のセルは空白とする。

退社時刻：終業時刻は17時である。17時前の退社は早退扱いとなる。出社しなかった日，又は存在しない日のセルは空白とする。

勤務時間：出社してから退社するまでの時間である。ただし，12時から13時までは昼休みであり，この時間は勤務時間に含まれない。出社しなかった日は0:00が表示される。また，日をまたいで勤務することはない。

勤怠理由：初期値は0である。欠勤，遅刻，早退について正当な理由がある場合は1を設定する。

ペナルティ：正当な理由なく出社予定日に欠勤した場合は3を，正当な理由なく遅刻又は早退した場合は1を表示する。これら以外の場合は0を表示する。

合計（勤務時間）：当該月の勤務時間の合計を表示する。

合計（ペナルティ）：当該月のペナルティの合計を表示する。

ペナルティの合計は，出勤割当表を作成するときに利用する。

設問1　ワークシート“勤怠管理”に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

各日のペナルティを算出するために，ワークシート“勤怠管理”のセルG4には次の式を入力し，セルG5～G34に複写する。

IF(F4＝1,0,IF(［　a　］,3,
IF(［　b　］,1,0)))

ペナルティの合計を算出するために，セルG35には次の式を入力する。

合計(G4～G34)

各日の勤務時間を算出するために，ワークシート“勤怠管理”のセルE4には次の式を入力し，セルE5～E34に複写する。算術演算のとき，空白セルは0として取り扱われる。

［　c　］

勤務時間の合計を算出するために，セルE35には次の式を入力する。

合計(E4～E34)

aに関する解答群

ア　否定(論理積(B4＝0,C4＝''))　　イ　否定(論理積(B4＝1,C4＝''))

ウ　否定(論理和(B4＝0,C4＝''))　　エ　否定(論理和(B4＝1,C4＝''))

オ　論理積(B4＝1,C4＝'')　　カ　論理和(B4＝1,C4＝'')

bに関する解答群

ア　論理積(B4＝1,論理積(C4≦(10／24),D4≧(17／24)))

イ　論理積(B4＝1,論理和(C4＞(10／24),D4＜(17／24)))

ウ　論理積(C4≦(10／24),D4≧(17／24))

エ　論理和(B4＝1,論理積(C4≦(10／24),D4≧(17／24)))

オ　論理和(B4＝1,論理和(C4＞(10／24),D4＜(17／24)))

カ　論理和(C4＞(10／24),D4＜(17／24))

cに関する解答群

ア　D4－C4－1／24

イ　最小(D4,12／24)－最小(C4,12／24)＋最大(D4,13／24)－最大(C4,13／24)

ウ　最小(D4,13／24)－最小(C4,12／24)－1／24

エ　最小(D4,13／24)－最大(C4,12／24)－1／24

オ　最大(D4,12／24)－最小(C4,12／24)＋最大(D4,13／24)－最小(C4,13／24)

カ　最大(D4,13／24)－最小(C4,12／24)－1／24

キ　最大(D4,13／24)－最大(C4,12／24)－1／24

〔ワークシート：勤務可能調査〕

月末になると翌月の出勤割当表の作成に先立って，アルバイトの申告した勤務可能日に基づいて，ワークシート“勤務可能調査”を作成する。その例を図2に示す。ワークシート“勤務可能調査”では，勤務可能なことを1で，勤務不可能なことを0で表す。また，存在しない日は，全員が0となる。

なお，アルバイトは15名とする。

	A	B	C	D	E	F	G	H	…	P
1		ID								
2	日付	0001	0002	0003	0004	0005	0006	0007	…	0015
3	2009-11-01	1	0	0	1	1	1	1	…	0
4	2009-11-02	1	0	1	0	1	0	0	…	1
5	2009-11-03	1	0	0	1	1	1	1	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…	⋮
32	2009-11-30	0	0	1	0	1	0	1	…	1
33	2009-11-31	0	0	0	0	0	0	0	…	0

図２　ワークシート“勤務可能調査”の例

〔ワークシート：出勤割当表〕

(1) ワークシート“勤務可能調査”に基づいて，ワークシート“出勤割当表”を作成する。その例を図３に示す。ワークシート“出勤割当表”では，勤務を割り当てた場合を1で，割り当てなかった場合を0で表す。

	A	B	C	D	E	F	G	H	…	P	Q
1		ID									
2	日付	0001	0002	0003	0004	0005	0006	0007	…	0015	合計
3	2009-11-01	1	0	0	0	1	1	0	…	0	5
4	2009-11-02	1	0	1	0	1	0	0	…	1	4
5	2009-11-03	1	0	0	0	1	1	0	…	0	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…	⋮	⋮
32	2009-11-30	0	0	1	0	1	0	1	…	1	5
33	2009-11-31	0	0	0	0	0	0	0	…	0	0
34	合計	21	6	8	1	16	14	9	…	17	
35											
36	経験順位	1	2	3	4	5	6	7	…	15	
37	先々月ペナルティ	0	3	2	6	0	0	1	…	0	
38	先々月ペナルティ順位	1	12	9	14	1	1	7	…	1	

図３　ワークシート“出勤割当表”の例

(2) 図３中の各項目の説明は，次のとおりである。

経験順位：業務経験が長い人ほど小さい値が入力されている。同順位の人はいない。

先々月ペナルティ：各アルバイトの先々月のペナルティの合計が入力されている。

先々月ペナルティ順位：各アルバイトの先々月ペナルティを昇順に順位付けする。先々月ペナルティが同じ場合は同順位となる。

(3) 出勤割当表の作成は，月初から月末まで順番に1日分ずつ次の規則に基づいて行う。

(a) その日に出社可能な人が5人以下の場合は，その全員を割り当てる。

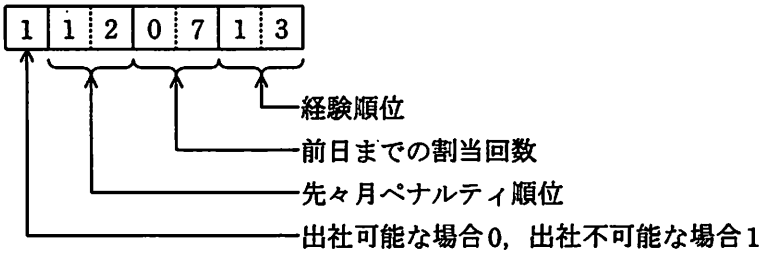
(b) その日に出社可能な人が5人より多い場合は，その中から次の優先順位で5人を割り当てる。

① 先々月ペナルティ順位がより小さい。

② ①が等しい場合は，既に割当てが決められた期間（月初から前日まで）に割り当てられた回数がより少ない。

③ ①，②がともに等しい場合は，経験順位がより小さい。

(4) (3)の規則を定式化するために，出社の可否，先々月ペナルティ順位，前日までの割当回数，経験順位からなる次のような7けたの数値（以下，評価値という）を定義する。



この評価値が小さい人から，順番に出勤を割り当てる。評価値の計算には，ワークシート“出勤割当表”の列AB～APを使用する。

(5) ワークシート“出勤割当表”で用いる関数を表に示す。

表　ワークシート“出勤割当表”で用いる関数

書式	説明
順位(照合値,照合範囲,順序)	順序に0又は1を指定したとき，照合範囲をそれぞれ昇順又は降順に整列し，照合値の順位を返す。照合範囲の中に同じ値が複数個ある場合は同順位とし，次の順位は同順位の個数を加えた値となる。

(6) 複数のワークシート間でデータを参照する場合には，“ワークシート名!セル”又は“ワークシート名!セル範囲”という形式で指定する。

設問2　ワークシート“出勤割当表”に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

先々月ペナルティ順位を算出するために，ワークシート“出勤割当表”のセルB38には次の式を入力し，セルC38～P38に複写する。

順位(B37,$B37～$P37,0)

次に，セルAB3～AP33に評価値を設定する。

セルAB3には次の式を入力し，セルAC3～AP3に複写する。

(((1－勤務可能調査!B3)＊100＋［　d　］)＊100)＊100＋［　e　］

セルAB4には次の式を入力し，セルAB4～AP33に複写する。

(((1－勤務可能調査!B4)＊100＋［　d　］)＊100＋［　f　］)＊100＋［　e　］

出勤者を決定するために，セルB3には次の式を入力し，セルB3～P33に複写する。

IF(論理積(［　　g　　］,
［　　h　　］),1,0)

各アルバイトの割当日数の合計を算出するために，セルB34には次の式を入力し，セルC34～P34に複写する。

合計(B3～B33)

各日の合計割当人数を算出するために，セルQ3には次の式を入力し，セルQ4～Q33に複写する。

合計(B3～P3)

d～fに関する解答群

ア　B$3	イ　B$36	ウ　B$38
エ　B3	オ　B36	カ　B38
キ　合計(B$3～B3)	ク　合計(B$3～$B3)	

g，hに関する解答群

ア　勤務可能調査!B3＝0

イ　勤務可能調査!B3＝1

ウ　合計($B3～$P3)≦5

エ　合計($B3～$P3)≧5

オ　合計(勤務可能調査!$B3～$P3)≦5

カ　合計(勤務可能調査!$B3～$P3)≧5

キ　順位(AB3,$AB3～$AP3,0)≦5

ク　順位(AB3,$AB3～$AP3,1)≦5

ケ　順位(B3,$B3～$P3,0)≦5

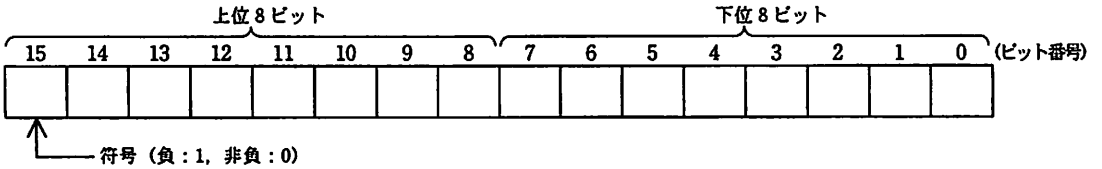
コ　順位(B3,$B3～$P3,1)≦5

■アセンブラ言語の仕様

1. システムCOMETⅡの仕様

1.1 ハードウェアの仕様

(1) 1語は16ビットで，そのビット構成は，次のとおりである。



(2) 主記憶の容量は65536語で，そのアドレスは0～65535番地である。

(3) 数値は，16ビットの2進数で表現する。負数は，2の補数で表現する。

(4) 制御方式は逐次制御で，命令語は1語長又は2語長である。

(5) レジスタとして，GR（16ビット），SP（16ビット），PR（16ビット），FR（3ビット）の4種類がある。

GR（汎用レジスタ，General Register）は，GR0～GR7の8個があり，算術，論理，比較，シフトなどの演算に用いる。このうち，GR1～GR7のレジスタは，指標レジスタ（index register）としてアドレスの修飾にも用いる。

SP（スタックポインタ，Stack Pointer）は，スタックの最上段のアドレスを保持している。

PR（プログラムレジスタ，Program Register）は，次に実行すべき命令語の先頭アドレスを保持している。

FR（フラグレジスタ，Flag Register）は，OF（Overflow Flag），SF（Sign Flag），ZF（Zero Flag）と呼ぶ3個のビットからなり，演算命令などの実行によって次の値が設定される。これらの値は，条件付き分岐命令で参照される。

OF	算術演算命令の場合は，演算結果が−32768～32767に収まらなくなったとき1になり，それ以外のとき0になる。論理演算命令の場合は，演算結果が0～65535に収まらなくなったとき1になり，それ以外のとき0になる。
SF	演算結果の符号が負（ビット番号15が1）のとき1，それ以外のとき0になる。
ZF	演算結果が零（全部のビットが0）のとき1，それ以外のとき0になる。

(6) 論理加算又は論理減算は，被演算データを符号のない数値とみなして，加算又は減算する。

1.2 命令

命令の形式及びその機能を示す。ここで，一つの命令コードに対し2種類のオペランドがある場合，上段はレジスタ間の命令，下段はレジスタと主記憶間の命令を表す。

命　令	書き方		命令の説明	FRの設定
	命令コード	オペランド		

(1) ロード，ストア，ロードアドレス命令

ロード LoaD	LD	r1,r2	r1 ← (r2)	○*1
		r,adr [,x]	r ← (実効アドレス)	
ストア STore	ST	r,adr [,x]	実効アドレス ← (r)	−
ロードアドレス Load ADdress	LAD	r,adr [,x]	r ← 実効アドレス	

(2) 算術，論理演算命令

算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	
論理積 AND	AND	r1,r2	r1 ← (r1) AND (r2)	○*1
		r,adr [,x]	r ← (r) AND (実効アドレス)	
論理和 OR	OR	r1,r2	r1 ← (r1) OR (r2)	
		r,adr [,x]	r ← (r) OR (実効アドレス)	
排他的論理和 eXclusive OR	XOR	r1,r2	r1 ← (r1) XOR (r2)	
		r,adr [,x]	r ← (r) XOR (実効アドレス)	

(3) 比較演算命令

算術比較 ComPare Arithmetic	CPA	r1,r2	(r1) と (r2)，又は (r) と (実効アドレス) の算術比較又は論理比較を行い，比較結果によって，FR に次の値を設定する。	○*1
		r,adr [,x]		
論理比較 ComPare Logical	CPL	r1,r2		
		r,adr [,x]		

比較結果	FR の値 SF	FR の値 ZF
(r1) ＞ (r2) (r) ＞ (実効アドレス)	0	0
(r1) ＝ (r2) (r) ＝ (実効アドレス)	0	1
(r1) ＜ (r2) (r) ＜ (実効アドレス)	1	0

(4) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	r,adr [,x]	符号を除き (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には，左シフトのときは 0，右シフトのときは符号と同じものが入る。	○*2
算術右シフト Shift Right Arithmetic	SRA	r,adr [,x]		
論理左シフト Shift Left Logical	SLL	r,adr [,x]	符号を含み (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には 0 が入る。	
論理右シフト Shift Right Logical	SRL	r,adr [,x]		

(5) 分岐命令

正分岐 Jump on PLus	JPL	adr [,x]	FR の値によって，実効アドレスに分岐する。分岐しないときは，次の命令に進む。	—
負分岐 Jump on MInus	JMI	adr [,x]		
非零分岐 Jump on Non Zero	JNZ	adr [,x]		
零分岐 Jump on ZEro	JZE	adr [,x]		
オーバフロー分岐 Jump on OVerflow	JOV	adr [,x]		
無条件分岐 unconditional JUMP	JUMP	adr [,x]	無条件に実効アドレスに分岐する。	

命令	分岐するときの FR の値 OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

(6) スタック操作命令

プッシュ PUSH	PUSH adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← 実効アドレス	—
ポップ POP	POP r	r ← ((SP)), SP ← (SP) $+_L$ 1	

(7) コール，リターン命令

コール CALL subroutine	CALL adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← (PR), PR ← 実効アドレス	—
リターン RETurn from subroutine	RET	PR ← ((SP)), SP ← (SP) $+_L$ 1	

(8) その他

スーパバイザコール SuperVisor Call	SVC adr [,x]	実効アドレスを引数として割出しを行う。実行後の GR と FR は不定となる。	—
ノーオペレーション No OPeration	NOP	何もしない。	

(注) r, r1, r2　いずれも GR を示す。指定できる GR は GR0 ～ GR7
adr　アドレスを示す。指定できる値の範囲は 0 ～ 65535
x　指標レジスタとして用いる GR を示す。指定できる GR は GR1 ～ GR7
[　]　[　] 内の指定は省略できることを示す。
(　)　(　) 内のレジスタ又はアドレスに格納されている内容を示す。
実効アドレス　adr と x の内容との論理加算値又はその値が示す番地
←　演算結果を，左辺のレジスタ又はアドレスに格納することを示す。
$+_L$, $-_L$　論理加算，論理減算を示す。
FR の設定　○　：設定されることを示す。
○*1：設定されることを示す。ただし，OF には 0 が設定される。
○*2：設定されることを示す。ただし，OF にはレジスタから最後に送り出されたビットの値が設定される。
－　：実行前の値が保持されることを示す。

1.3 文字の符号表

(1) JIS X 0201 ラテン文字・片仮名用 8 ビット符号で規定する文字の符号表を使用する。
(2) 右に符号表の一部を示す。1 文字は 8 ビットからなり，上位 4 ビットを列で，下位 4 ビットを行で示す。例えば，間隔，4，H，¥ のビット構成は，16 進表示で，それぞれ 20，34，48，5C である。16 進表示で，ビット構成が 21 ～ 7E（及び表では省略している A1 ～ DF）に対応する文字を図形文字という。図形文字は，表示（印刷）装置で，文字として表示（印字）できる。
(3) この表にない文字とそのビット構成が必要な場合は，問題中で与える。

行＼列	02	03	04	05	06	07
0	間隔	0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[	k	{
12	,	<	L	¥	l	|
13	-	=	M	]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

2. アセンブラ言語 CASLⅡの仕様

2.1 言語の仕様

(1) CASLⅡは，COMETⅡのためのアセンブラ言語である。
(2) プログラムは，命令行及び注釈行からなる。
(3) 1命令は1命令行で記述し，次の行へ継続できない。
(4) 命令行及び注釈行は，次に示す記述の形式で，行の1文字目から記述する。

行の種類		記述の形式
命令行	オペランドあり	[ラベル] {空白} {命令コード} {空白} {オペランド} [{空白} [コメント]]
	オペランドなし	[ラベル] {空白} {命令コード} [{空白} [{;} [コメント]]]
注釈行		[空白] {;} [コメント]

(注) [ ] [ ] 内の指定が省略できることを示す。
{ } { } 内の指定が必須であることを示す。
ラベル その命令の（先頭の語の）アドレスを他の命令やプログラムから参照するための名前である。長さは 1～8 文字で，先頭の文字は英大文字でなければならない。以降の文字は，英大文字又は数字のいずれでもよい。なお，予約語である GR0～GR7 は，使用できない。
空白 1文字以上の間隔文字の列である。
命令コード 命令ごとに記述の形式が定義されている。
オペランド 命令ごとに記述の形式が定義されている。
コメント 覚え書きなどの任意の情報であり，処理系で許す任意の文字を書くことができる。

2.2 命令の種類

命令は，4種類のアセンブラ命令（START，END，DS，DC），4種類のマクロ命令（IN，OUT，RPUSH，RPOP）及び機械語命令（COMETⅡの命令）からなる。その仕様を次に示す。

命令の種類	ラベル	命令コード	オペランド	機能
アセンブラ命令	ラベル	START	[実行開始番地]	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
		END		プログラムの終わりを明示
	[ラベル]	DS	語数	領域を確保
	[ラベル]	DC	定数 [,定数] …	定数を定義
マクロ命令	[ラベル]	IN	入力領域,入力文字長領域	入力装置から文字データを入力
	[ラベル]	OUT	出力領域,出力文字長領域	出力装置へ文字データを出力
	[ラベル]	RPUSH		GR の内容をスタックに格納
	[ラベル]	RPOP		スタックの内容を GR に格納
機械語命令	[ラベル]	（「1.2 命令」を参照）		

2.3 アセンブラ命令

アセンブラ命令は，アセンブラの制御などを行う。

(1) START [実行開始番地]

START 命令は，プログラムの先頭を定義する。
実行開始番地は，そのプログラム内で定義されたラベルで指定する。指定がある場合はその番地から，省略した場合は START 命令の次の命令から，実行を開始する。
また，この命令につけられたラベルは，他のプログラムから入口名として参照できる。

(2) | END | |

END 命令は，プログラムの終わりを定義する。

(3) | DS | 語数 |

DS 命令は，指定した語数の領域を確保する。

語数は，10 進定数（≧0）で指定する。語数を 0 とした場合，領域は確保しないが，ラベルは有効である。

(4) | DC | 定数［,定数］… |

DC 命令は，定数で指定したデータを（連続する）語に格納する。

定数には，10 進定数，16 進定数，文字定数，アドレス定数の 4 種類がある。

定数の種類	書き方	命令の説明
10 進定数	n	n で指定した 10 進数値を，1 語の 2 進数データとして格納する。ただし，n が−32768～32767 の範囲にないときは，その下位 16 ビットを格納する。
16 進定数	#h	h は 4 けたの 16 進数（16 進数字は 0～9，A～F）とする。h で指定した 16 進数値を 1 語の 2 進数データとして格納する（0000≦h≦FFFF）。
文字定数	'文字列'	文字列の文字数（＞0）分の連続する領域を確保し，最初の文字は第 1 語の下位 8 ビットに，2 番目の文字は第 2 語の下位 8 ビットに，…と順次文字データとして格納する。各語の上位 8 ビットには 0 のビットが入る。 文字列には，間隔及び任意の図形文字を書くことができる。ただし，アポストロフィ（'）は 2 個続けて書く。
アドレス定数	ラベル	ラベルに対応するアドレスを 1 語の 2 進数データとして格納する。

2.4 マクロ命令

マクロ命令は，あらかじめ定義された命令群とオペランドの情報によって，目的の機能を果たす命令群を生成する（語数は不定）。

(1) | IN | 入力領域,入力文字長領域 |

IN 命令は，あらかじめ割り当てた入力装置から，1 レコードの文字データを読み込む。

入力領域は，256 語長の作業域のラベルであり，この領域の先頭から，1 文字を 1 語に対応させて順次入力される。レコードの区切り符号（キーボード入力の復帰符号など）は，格納しない。格納の形式は，DC 命令の文字定数と同じである。入力データが 256 文字に満たない場合，入力領域の残りの部分は実行前のデータを保持する。入力データが 256 文字を超える場合，以降の文字は無視される。

入力文字長領域は，1 語長の領域のラベルであり，入力された文字の長さ（≧0）が 2 進数で格納される。ファイルの終わり（end of file）を検出した場合は，−1 が格納される。

IN 命令を実行すると，GR の内容は保存されるが，FR の内容は不定となる。

(2) | OUT | 出力領域,出力文字長領域 |

OUT 命令は，あらかじめ割り当てた出力装置に，文字データを，1 レコードとして書き出す。

出力領域は，出力しようとするデータが 1 文字 1 語で格納されている領域のラベルである。格納の形式は，DC 命令の文字定数と同じであるが，上位 8 ビットは，OS が無視するので 0 でなくてもよい。

出力文字長領域は，1 語長の領域のラベルであり，出力しようとする文字の長さ（≧0）を 2 進数で格納しておく。

OUT 命令を実行すると，GR の内容は保存されるが，FR の内容は不定となる。

(3) | RPUSH |

RPUSH 命令は，GR の内容を，GR1，GR2，…，GR7 の順序でスタックに格納する。

(4) | RPOP |

RPOP 命令は，スタックの内容を順次取り出し，GR7，GR6，…，GR1 の順序で GR に格納する。

2.5 機械語命令

機械語命令のオペランドは，次の形式で記述する。

r, r1, r2	GR は，記号 GR0 ～ GR7 で指定する。
x	指標レジスタとして用いる GR は，記号 GR1 ～ GR7 で指定する。
adr	アドレスは，10 進定数，16 進定数，アドレス定数又はリテラルで指定する。 リテラルは，一つの 10 進定数，16 進定数又は文字定数の前に等号（=）を付けて記述する。CASL Ⅱ は，等号の後の定数をオペランドとする DC 命令を生成し，そのアドレスを adr の値とする。

2.6 その他

(1) アセンブラによって生成される命令語や領域の相対位置は，アセンブラ言語での記述順序とする。ただし，リテラルから生成される DC 命令は，END 命令の直前にまとめて配置される。

(2) 生成された命令語，領域は，主記憶上で連続した領域を占める。

3. プログラム実行の手引

3.1 OS

プログラムの実行に関して，次の取決めがある。

(1) アセンブラは，未定義ラベル（オペランド欄に記述されたラベルのうち，そのプログラム内で定義されていないラベル）を，他のプログラムの入口名（START 命令のラベル）と解釈する。この場合，アセンブラはアドレスの決定を保留し，その決定を OS に任せる。OS は，実行に先立って他のプログラムの入口名との連係処理を行いアドレスを決定する（プログラムの連係）。

(2) プログラムは，OS によって起動される。プログラムがロードされる主記憶の領域は不定とするが，プログラム中のラベルに対応するアドレス値は，OS によって実アドレスに補正されるものとする。

(3) プログラムの起動時に，OS はプログラム用に十分な容量のスタック領域を確保し，その最後のアドレスに 1 を加算した値を SP に設定する。

(4) OS は，CALL 命令でプログラムに制御を渡す。プログラムを終了し OS に制御を戻すときは，RET 命令を使用する。

(5) IN 命令に対応する入力装置，OUT 命令に対応する出力装置の割当ては，プログラムの実行に先立って利用者が行う。

(6) OS は，入出力装置や媒体による入出力手続の違いを吸収し，システムでの標準の形式及び手続（異常処理を含む）で入出力を行う。したがって，IN，OUT 命令では，入出力装置の違いを意識する必要はない。

3.2 未定義事項

プログラムの実行等に関し，この仕様で定義しない事項は，処理系によるものとする。

表計算ソフトの機能・用語

表計算ソフトの機能，用語などは，原則として次による。

1. ワークシート

表計算ソフトの作業領域をワークシートという。ワークシートの大きさは256列（列Aから列Z，列AAから列AZ，さらに列BAから列BZと続き，列IVまで続く），10,000行（行1から行10,000まで）とする。

2. セル

(1) ワークシートを縦・横に分割したときの一つのます目をセルという。列A行1のセルはA1と表す。

(2) 長方形の形をしたセルの集まりを範囲として指定することができる。範囲の指定はA1～B3のように表す。

(3) 範囲に名前を付けることができる。範囲名は[ ]を用いて，“セルA1～B3に[金額]と名前を付ける”などと表す。

(4) データが入力されていないセルを，空白セルという。

3. セルへの入力

(1) セルに数値，文字列，計算式を入力できる。

(2) セルを保護すると，そのセルへの入力を不可能にすることができる。セルの保護を解除すると，そのセルへの入力が再び可能になる。

(3) セルA1に数値5を入力するときは，“セルA1に5を入力”と表す。

(4) セルB2に，文字列ABCを入力するときは，“セルB2に'ABC'を入力”と表す。

(5) セルC3に，セルA1とセルB2の和を求める計算式を入力するときは，“セルC3に計算式A1+B2を入力”などと表す。

4. セルの内容の表示

(1) セルに数値を入力すると，右詰めで表示される。

(2) セルに文字列を入力すると，左詰めで表示される。

(3) セルに計算式を入力すると，計算結果が数値ならば右詰めで，文字列ならば左詰めで表示される。

(4) セルの内容の表示については，左詰め，中央揃え，右詰めに変更できる。

5. 計算式

(1) 計算式には，数学で用いられる数式が利用できる。

(2) 計算式で使用する算術演算子は，“+”（加算），“−”（減算），“*”（乗算），“／”（除算）及び“^”（べき算）とする。

(3) 算術演算子による計算の優先順位は，数学での優先順位と同じである。

6. 再計算

(1) セルに計算式を入力すると，直ちに計算結果を表示する。

(2) セルの数値が変化すると，そのセルを参照しているセルも自動的に再計算される。この再計算はA1，A2，A3，…，B1，B2，B3，…の順に1回だけ行われる。

7. 関数

(1) 計算式には次の表で定義する関数を利用することができる。

関数名と使用例	解　　説
合計(A1～A5)	セルA1からセルA5までの範囲のすべての数値の合計を求める。
平均(B2～F2)	セルB2からセルF2までの範囲のすべての数値の平均を求める。
平方根(I6)	セルI6の値（正の数値でなければならない）の正の平方根を求める。
標準偏差(D5～D19)	セルD5からセルD19までの範囲のすべての数値の標準偏差を求める。
最大(C3～E7)	セルC3からセルE7までの範囲のすべての数値のうちの最大値を求める。
最小([得点])	[得点]と名前を付けた範囲のすべての数値のうちの最小値を求める。
IF(B3＞A4，'北海道'，'九州')	第1引数に指定された論理式が真（成立する）ならば第2引数が，偽（成立しない）ならば第3引数が求める値となる。左の例では，セルB3がA4より大きければ文字列'北海道'が，それ以外の場合には文字列'九州'が求める値となる。論理式中では，比較演算子として，＝，≠，＞，＜，≦，≧を利用することができる。第2引数，第3引数に，更にIF関数を利用して，IF関数を入れ子にすることができる。
個数(G1～G5)	セルG1からG5までの範囲のうち，空白セルでないセルの個数を求める。
条件付個数(H5～H9，'＞25')	第1引数に指定された範囲のうち，第2引数に指定された条件を満たすセルの個数を求める。左の例では，セルH5からH9までの範囲のうち，値として25より大きな数値を格納しているセルの個数を求める。
整数部(A3)	セルA3の値（数値でなければならない）を超えない最大の整数を求める。 例えば， 整数部(3.9)＝3 整数部(－3.9)＝－4 となる。
剰余(C4，D4)	セルC4の値を被除数，D4の値を除数とし，被除数を除数で割ったときの剰余を求める。剰余の値は常に除数と同じ符号をもつ。“剰余”関数と“整数部”関数は，次の関係を満たしている。 剰余(x，y)＝x－y＊整数部(x／y)
論理積(論理式1，論理式2，…)	引数として指定された論理式がすべて真であれば，真を返す。引数のうち一つでも偽のものがあれば，偽を返す。引数として指定できる論理式の数は任意である。
論理和(論理式1，論理式2，…)	引数として指定された論理式がすべて偽であれば，偽を返す。引数のうち一つでも真のものがあれば，真を返す。引数として指定できる論理式の数は任意である。
否定(論理式)	引数として指定された論理式が真であれば偽を，偽であれば真を返す。
注　“合計”，“平均”，“標準偏差”，“最大”，“最小”は，引数で指定された範囲のセルのうち，値として数値以外を格納しているものは無視する。	

(2) 関数の引数には，セルを用いた計算式，範囲，範囲名，論理式を指定することができる。

8. セルの複写

(1) セルに入力された数値，文字列，計算式を他のセルに複写することができる。

(2) セルに入力された計算式が他のセルを参照している場合は，複写先のセルでは相対的にセルが自動的に変更される。例えば，セルA6に合計(A1～A5)を入力した場合，セルA6をセルB7に複写すると，セルB7の計算式は合計(B2～B6)となる。

9. 絶対参照

(1) 計算式を複写しても参照したセルが変わらない参照を絶対参照といい，記号$を用いて$A$1などと表す。例えば，セルB1に計算式$A$1+5を入力した場合，セルB1をセルC4に複写してもセルC4の計算式はA1+5のままである。

(2) 絶対参照は行と列の一方だけについても指定可能であり，$A1，A$1などと表す。例えば，セルD2に計算式$C1−3を入力した場合，セルD2をセルE3に複写すると，セルE3の計算式は$C2−3となる。また，セルG3に計算式F$2−3を入力した場合，セルG3をH4に複写すると，セルH4の計算式はG$2−3となる。

10. マクロ

(1) ワークシートには幾つかのマクロを保存できる。マクロはマクロP，マクロQなどと表す。

(2) マクロについては“マクロPを実行するとワークシートを保存する。”，“セルA1からセルA10までを昇順に並べ替える手続をマクロQに登録する。”，“マクロR：数値を入力。”，“C列のデータがその数値以下のものを抽出する。”などと記述する。

11. その他

ワークシートの“保存”，“読出し”，“印刷”や，罫線機能，グラフ化機能など市販されている多くの表計算ソフトに備わっている機能は使用できるものとする。

〔 メ モ 用 紙 〕

7. 途中で退室する場合には，手を挙げて監督員に合図し，答案用紙が回収されてから静かに退室してください。

退室可能時間	13:40 ～ 15:20

8. 問題に関する質問にはお答えできません。文意どおり解釈してください。
9. 問題冊子の余白などは，適宜利用して構いません。
10. アセンブラ言語の仕様及び表計算ソフトの機能・用語は，この冊子の末尾を参照してください。
11. 試験時間中，机上に置けるもの及び使用できるものは，次のものに限ります。
なお，会場での貸出しは行っていません。
受験票，黒鉛筆又はシャープペンシル，鉛筆削り，消しゴム，定規，時計（アラームなど時計以外の機能は使用不可），ハンカチ，ティッシュ
これら以外は机上に置けません。使用もできません。
12. 試験終了後，この問題冊子は持ち帰ることができます。
13. 答案用紙は，いかなる場合でも提出してください。回収時に提出しない場合は，採点されません。
14. 試験時間中にトイレへ行きたくなったり，気分が悪くなったりした場合は，手を挙げて監督員に合図してください。