FE

平成24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. **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。**
4. 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

5. 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) 答案用紙は光学式読取り装置で読み取った上で採点しますので，B又はHBの黒鉛筆で答案用紙の**マークの記入方法**のとおりマークしてください。マークの濃度がうすいなど，**マークの記入方法**のとおり正しくマークされていない場合は読み取れません。特にシャープペンシルを使用する際には，マークの濃度に十分注意してください。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) **受験番号欄**に**受験番号**を，**生年月日欄**に**受験票の生年月日**を記入及びマークしてください。答案用紙の**マークの記入方法**のとおり記入及び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。生年月日欄については，受験票の生年月日を訂正した場合でも，訂正前の生年月日を記入及びマークしてください。

〔問1，問3，問4，問6，問7，問9を選択した場合の例〕

選択欄					
問1	●	問8	●	問9	●
問2	(選)			問10	(選)
問3	●			問11	(選)
問4	●			問12	(選)
問5	(選)			問13	(選)
問6	●				
問7	●				

   (3) **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。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。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。
   (4) **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。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。

〔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	ソフトウェア	プロセスの排他制御
問 2	データベース	購買情報を管理する関係データベースの設計及び運用
問 3	ネットワーク	電子メールで用いる MIME 形式
問 4	情報セキュリティ	セキュリティ事故の対応
問 5	ソフトウェア設計	通信講座受講管理システム
問 6	IT サービスマネジメント	データ管理
問 7	経営戦略・企業と法務	在庫管理

●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	駅間の最短距離を求めるプログラム

●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）	くじの当選番号の確認
問 10	ソフトウェア開発（COBOL）	スポーツクラブの利用料金の計算
問 11	ソフトウェア開発（Java）	スレッドを利用したタイマ
問 12	ソフトウェア開発（アセンブラ）	多項式の計算
問 13	ソフトウェア開発（表計算）	最適配置問題

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。

E
後

〔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

注記　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

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，選択した問題については，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問以上マークした場合には，はじめの5問について採点します。

問1　プロセスの排他制御に関する次の記述を読んで，設問1～3に答えよ。

複数のプロセスが，共有するデータ（以下，共有データという）を書き換える処理を，並行して実行する場合がある。このようにプロセス間でデータを共有する方法の一つとして，プロセス間で共有するメモリ（以下，共有メモリという）に共有データを格納する方法がある。ここでは，CPU が一つで共有メモリをもつコンピュータ X 上で，二つのプロセス p1，p2 が共有メモリを使用して並行に処理を実行する場合を考える。

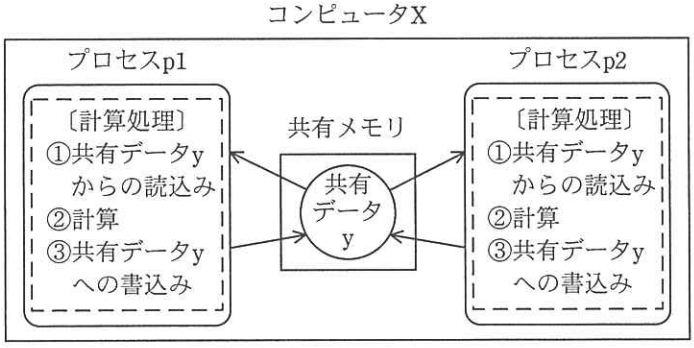
プロセス p1，p2 が共有データ y に対して計算処理をする場合を，図1に示す。ここで，各プロセスの計算処理は，次のとおりである。

〔計算処理〕

① 共有データ y の値を読み込む。

② 読み込んだ値を用いて計算する。

③ 計算結果を y に書き込む。



注記　“⟶”は共有データへのアクセス（読込み又は書込み）を表す。

図1　共有メモリによるデータの共有

設問1　図1に示すプロセス p1，p2 が共有データ y に対して次の処理を実行する場合を考える。

プロセス p1：y の値を 2 だけ増加させる。
プロセス p2：y の値を 1 だけ減少させる。

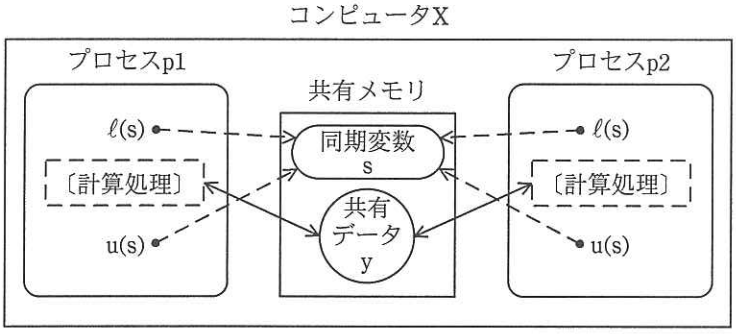
プロセスの実行前の共有データ y の値が 5 であり，y に対する排他制御をしないとき，プロセス p1，p2 が並行に 1 回だけ処理を実行した直後において，y が**取り得ない値**を，解答群の中から選べ。

解答群

ア　4　　イ　5　　ウ　6　　エ　7

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。

プロセスの排他制御の仕組みとして，共有データがいずれかのプロセスに確保されている状態（以下，確保状態という）又はどのプロセスにも確保されていない状態（以下，解放状態という）のいずれかの状態をもつ同期変数を使用する方法を考える。図1のプロセス p1，p2 の計算処理で使用する共有データ y に対して同期変数 s を用いて排他制御する場合を，図2に示す。



注記　“←→”は共有データへのアクセスを表す。
“•－－→”は同期変数 s への操作を表す。

図2　同期変数 s を用いた排他制御

図2において，同期変数の状態を変更する関数 ℓ と u があり，同期変数を引数で指定する。各プロセスは，共有データを排他制御して計算処理をする場合，関数 ℓ の呼出し，計算処理，関数 u の呼出しの順番で処理を実行する。このとき，共有データ y を排他制御するために，関数 ℓ と u の引数として同期変数 s を指定する。

関数 ℓ の操作内容は［ a ］処理であり，関数 u の操作内容は［ b ］処理である。ここで，“同期変数の状態を調べて，変更する処理”は中断のない処理として実行されるものとする。ただし，プロセスが待ちの状態になったら，CPU は別のプロセスに割り付けられるものとする。

解答群

ア　s の状態が解放状態ならば確保状態にし，確保状態ならば解放状態になるまで待ってから確保状態にする

イ　s の状態が解放状態ならば確保状態にし，確保状態ならば何もしない

ウ　s の状態が解放状態ならば確保状態になるまで待ち，確保状態ならば解放状態になるまで待つ

エ　s の状態が解放状態ならば確保状態になるまで待ち，確保状態ならば何もしない

オ　s の状態が確保状態ならば解放状態にし，解放状態ならば何もしない

カ　s の状態が確保状態ならば解放状態になるまで待ち，解放状態ならば何もしない

設問3　プロセス p1，p2 が使用する共有データが二つあり，共有データ y1 に対して同期変数 s1 を用いて排他制御し，共有データ y2 に対して同期変数 s2 を用いて排他制御する場合を考える。プロセス p1 が，y1 の確保，y2 の確保，y2 の解放，y1 の解放の順序で同期変数を操作するとき，プロセス p2 が y1，y2 の確保と解放を行う順序によってはデッドロックが発生する可能性がある。デッドロックが発生する可能性のあるプロセス p2 の操作の順序を，解答群の中から選べ。

解答群

ア　y1 の確保，y1 の解放，y2 の確保，y2 の解放

イ　y1 の確保，y2 の確保，y2 の解放，y1 の解放

ウ　y2 の確保，y1 の確保，y1 の解放，y2 の解放

エ　y2 の確保，y2 の解放，y1 の確保，y1 の解放

問2　購買情報を管理する関係データベースの設計及び運用に関する次の記述を読んで，設問1～4に答えよ。

B社では，購買業務の効率化を目的に，社内組織を変更して，これまでX事業部とY事業部に個別に存在していた購買部門を統合した。

これに伴い，それぞれの事業部が関係データベースで個別に管理していた購買データも統合した。統合後の表構成は図1のとおりであり，統合前からX事業部とY事業部ともに同じ表構成で管理していた。下線付きの項目は主キーを表す。

取引先表

取引先コード	社名	所在地	電話番号
XK001	情報商事株式会社	東京都文京区桜坂2-28	03-1111-2222

部表

部コード	部名
0001	X事業部営業部

商品表

商品コード	品目コード	商品名	定価
A0001	S01	エコ鉛筆黒	50

品目表

品目コード	品目名
S01	文具

発注表

発注コード	取引先コード	部コード	発注日	納品日
120001	XK001	0001	20120110	20120112

明細表

発注コード	商品コード	数量	購入額
120001	A0001	20	900
120001	A0027	10	1200

図1　表構成と統合後のデータの格納例

設問1　データを統合したときに実施した“名寄せ”と呼ばれる作業に関する次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

同じ情報が，表現が異なるデータとして社内に複数存在する場合がある。それぞれが，関連性のないデータベースで個別に管理されているのであれば，データの重複といった問題は発生しない。しかし，データベースの統合が必要になった場合，単純にデータを寄せ集めると，同じ情報を表すデータであるにもかかわらず別のデータとして格納されてしまい，［　a　］，［　b　］といった問題が発生する。

このような場合，同じ情報を表すデータを一つのデータにまとめる“名寄せ”と呼ばれる作業が必要となる。

B社では，名寄せを次の手順で実施した。

(1)　統合対象の表のデータを見比べて，表現は異なっていても同じ情報を表しているデータかどうかを識別するのに最も適した項目を特定する。

(2)　(1)で特定した項目で突合せを行い，同じ情報を表すデータを一つのデータにまとめる。

それぞれの事業部の取引先表から(1)の作業のために抽出した，名寄せが必要となる典型的なデータを，図2及び図3に示す。これらのデータから，突合せを行う項目は［　c　］が適切と判断できる。

X事業部の取引先表

取引先コード	社名	所在地	電話番号
XK001	情報商事株式会社	東京都文京区桜坂2-28	03-1111-2222
XK022	（株）情報商事	神奈川県横浜市大湊1-5	045-987-6543

図2　X事業部の取引先表から抽出したデータ

Y事業部の取引先表

取引先コード	社名	所在地	電話番号
0008	情報商事	新宿区青葉2丁目1番地	03-1234-5678
0105	情報商事（株）	文京区桜坂2丁目28番	03-1111-2222

図3　Y事業部の取引先表から抽出したデータ

a，bに関する解答群

ア　更新すべきデータが更新されない　　イ　削除すべきデータが残る

ウ　情報が漏えいする　　エ　挿入したはずのデータが消える

オ　トランザクションが遅延する

cに関する解答群

ア　社名　　イ　所在地

ウ　電話番号　　エ　取引先コード

設問2　統合された購買部門で分析したところ，同じ商品であっても，取引先によって購入単価が異なることが分かった。商品名“エコ鉛筆黒”について，取引先コードと平均購入単価を，金額が安い順に表示する。次のSQL文の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，商品名は一意に管理できているものとする。

```
SELECT 発注表.取引先コード,
       SUM(明細表.購入額) / SUM(明細表.数量) AS 平均購入単価
  FROM 発注表, 商品表, 明細表
  WHERE [          d          ]
  GROUP BY 発注表.取引先コード
  ORDER BY 平均購入単価
```

解答群

ア
```
発注表.発注コード = 明細表.発注コード AND
商品表.商品コード = 明細表.商品コード AND
商品表.商品名 = 'エコ鉛筆黒'
```

イ
```
発注表.発注コード = 明細表.発注コード AND
明細表.購入額 = (SELECT AVG(商品表.定価) FROM 商品表
                 WHERE 商品表.商品名 = 'エコ鉛筆黒')
```

ウ
```
発注表.発注コード = 明細表.発注コード AND
明細表.購入額 = (SELECT SUM(商品表.定価) FROM 商品表
                 WHERE 商品表.商品名 = 'エコ鉛筆黒')
```

エ
```
発注表.発注コード = 明細表.発注コード AND
明細表.商品コード = (SELECT COUNT(*) FROM 商品表
                     WHERE 商品表.商品名 = 'エコ鉛筆黒')
```

設問3　より安く商品を購入するために，購買部門の発注担当者は，商品の発注時に割引率を確認してから発注先を選定することにした。発注先選定時の検索処理に関する次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

商品コードごとの平均割引率は，［ e ］と［ f ］を結合すると求められる。加えて，取引先コードごとに集計したり，“以前は高かったが，最近は安くなった”といった傾向を把握したりしたい場合は，更に［ g ］を結合することで求められる。

解答群

ア　商品表	イ　取引先表	ウ　発注表
エ　品目表	オ　部表	カ　明細表

設問4　複数の取引先から文具を購入していたが，品ぞろえは同じだったので，1 社に一括発注することによって割引率を上げてもらうよう交渉することにした。文具について 10 回以上の発注実績がある取引先を対象に，購入金額が多い順に取引先コードと金額を表示する。次の SQL 文の［　　］に入れる正しい答えを，解答群の中から選べ。

```
SELECT 発注表.取引先コード, SUM(明細表.購入額) AS 合計購入金額
   FROM 発注表, 明細表, 商品表, 品目表
   WHERE 発注表.発注コード = 明細表.発注コード AND
         明細表.商品コード = 商品表.商品コード AND
         商品表.品目コード = 品目表.品目コード AND
         品目表.品目名 = '文具'
   GROUP BY 発注表.取引先コード
   [          h          ]
   ORDER BY 合計購入金額 DESC
```

解答群

ア　HAVING COUNT(*) >= 10

イ　HAVING COUNT(DISTINCT 発注表.発注コード) >= 10

ウ　HAVING COUNT(発注表.発注コード) >= 10

エ　HAVING MAX(明細表.購入額) >= 10

オ　HAVING MAX(明細表.数量) >= 10

カ　HAVING SUM(明細表.購入額) >= 10

キ　HAVING SUM(明細表.数量) >= 10

問3　電子メールで用いるMIME形式に関する次の記述を読んで，設問1，2に答えよ。

インターネットの電子メールは，規格上，US-ASCIIのような7ビット符号で書かれたテキストしか送信できない。そのため，UTF-8のような8ビット符号で書かれたテキストや，画像データなどのバイナリデータを電子メールで送信する際は，MIME（Multipurpose Internet Mail Extensions）と呼ばれる書式（以下，MIME形式という）に従ってメッセージを作成する。

MIME形式では，7ビット符号で書かれたテキスト以外のコンテンツを，base64やquoted-printableなどの方式で7ビット符号に変換（以下，エンコードという）する。ただし，エンコードを行うコンピュータのメモリ上では，7ビット符号を，先頭に0のビットを1ビット付加した8ビット（1バイト）として取り扱う。

base64によるエンコードでは，コンテンツを先頭から6ビットごとに区切り，各6ビットを，ビットパターンごとに定められた，US-ASCIIの図形文字1文字に変換する。

一方，quoted-printableによるエンコードでは，コンテンツをバイト列とみなし，US-ASCIIの制御文字又は図形文字“=”と一致するバイト，及び先頭ビットが1のバイトを，“=XX”（XXは2桁の16進数字列）の形の3文字のUS-ASCIIの図形文字列に置き換える。US-ASCIIの図形文字（“=”を除く）と一致するバイトは置き換えない。幾つかの例外があるが，ここでは考慮しなくてよいものとする。

例えば，UTF-8で書かれたテキスト“©△IPA△2012.”（“©”には16進数でC2A9の2バイトの符号が，他の文字にはUS-ASCIIと同じ1バイトの符号が，それぞれ割り当てられている）を［ a1 ］でエンコードすると“=C2=A9△IPA△2012.”になる。同じテキストを［ a2 ］でエンコードすると“wqkgSVBBIDIwMTIu”となり，いずれもUS-ASCIIに含まれる図形文字だけから成る7ビット符号のバイト列となる。ここで，“△”は空白（符号は16進数で20）を表すものとする。

エンコード後のデータ量に着目すると，大部分がUS-ASCIIに含まれる図形文字で構成されているテキストのエンコードには［ a3 ］が適しており，バイナリデータのエンコードには［ a4 ］が適している。

UTF-8 では，平仮名 1 文字に先頭ビットが 1 であるバイト三つから成る符号を割り当てているので，UTF-8 で書かれた 6 文字の平仮名から成るテキストを quoted-printable でエンコードすると，［ b ］文字の文字列となる。

設問 1　記述中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，a1 ～ a4 に入れる答えは，a に関する解答群の中から組合せとして正しいものを選ぶものとする。

a に関する解答群

	a1	a2	a3	a4
ア	base64	quoted-printable	base64	quoted-printable
イ	base64	quoted-printable	quoted-printable	base64
ウ	quoted-printable	base64	base64	quoted-printable
エ	quoted-printable	base64	quoted-printable	base64

b に関する解答群

ア　8　　イ　18　　ウ　24　　エ　54

設問2　MIME 形式を使用すると，1 通の電子メールに複数のコンテンツを格納することができる。1 通の電子メールに二つのコンテンツを格納した MIME 形式のメッセージは，図 1 のようになる。

```
通常の電子メールのヘッダ
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="delimiter"

--delimiter
コンテンツごとの属性を示すヘッダ

コンテンツ1
--delimiter
コンテンツごとの属性を示すヘッダ

コンテンツ2
--delimiter--
```

注記　網掛けは，その位置に格納される情報の種類を表している。

図 1　二つのコンテンツを格納した MIME 形式のメッセージ

boundary="*delimiter*"の *delimiter* には，格納したコンテンツの区切りを示す文字列を指定する。

delimiter の先頭に“--”を付けたものが一つのコンテンツの開始を示し，*delimiter* を“--”で囲ったものが最後のコンテンツの終了を示す。

delimiter には，コンテンツに含まれる行が，誤ってコンテンツの区切りと認識されることのないような文字列を，選択しなければならない。

“コンテンツごとの属性を示すヘッダ”には，テキストや画像などのコンテンツの種類や，エンコードの方式などを指定する。エンコードの方式には“base64”や“quoted-printable”のほか，コンテンツがエンコードされていない 7 ビット符号のテキストであることを示す“7bit”などが指定できる。

図 1 中の“コンテンツ 1”が 7 ビット符号のテキストであって，図 2 に示す 3 行をコンテンツに含むとき，*delimiter* として**ふさわしくない文字列**を，解答群の中から三つ選べ。

E
後

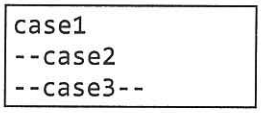


図２　“コンテンツ 1” に含まれる３行

解答群

ア　--case1	イ　--case2	ウ　--case3
エ　case1	オ　case2	カ　case3
キ　case1--	ク　case2--	ケ　case3--

問4　セキュリティ事故の対応に関する次の記述を読んで，設問1～4に答えよ。

自転車用品の中堅通信販売会社の D 社では，顧客からの注文を郵便及び電話で受け付けていた。顧客へのサービスの拡大を目的として，インターネットを利用した会員制のサービスを開始することとした。

会社の紹介だけを掲載していた従来の Web サイトを改修し，Web サイトでの会員情報の登録及び修正，会員に対する Web サイトでの商品の販売並びに会員向けのメールマガジン送付の登録を行う。

なお，Web サイトでの商品の販売における決済手段はクレジットカードだけとする。

改修後の Web サイトは，D 社の DMZ 上に設置された Web サーバと，社内 LAN 内に設置されたデータベースサーバで構成される。データベースサーバのデータは平文で保存しており，会員情報の登録及び修正，商品の販売並びに会員向けのメールマガジン送付の登録を行う際には，SSL によってネットワーク経路の暗号化を行う。

〔会員情報の登録〕

D 社では，Web サイトで取得する個人情報の利用目的を，注文の受付，決済，商品の配送，及びメールマガジンの送付に限定し，会員登録の希望者に対し，登録時に利用目的を通知して同意を得ることとした。同意を得た会員に対しては，会員情報として次の情報を登録してもらうこととした。

【全員に登録してもらう情報】

利用者 ID，パスワード，メールアドレス，メールマガジン送付の有無

【任意に登録してもらう情報】

性別，生年月日

【商品を購入する会員に登録してもらう情報】

氏名，配送先住所，電話番号，クレジットカードの発行会社名，
クレジットカード番号，クレジットカードの有効期限

〔セキュリティ事故の発生〕

複数の会員から，“D 社のサービスに登録したメールアドレス宛てに迷惑メールが大量に送られてくるようになった”との連絡がお客様相談窓口に入った。さらに，連絡があった会員のうち数名については，迷惑メールの宛先メールアドレスは D 社以外のサービスでは利用していないことが分かった。

この報告を受けて D 社の情報システム部の Y 部長は，会員情報が漏えいしている可能性があると判断した。また，会員情報として登録されているクレジットカード情報が漏えいしていることも考えられると判断した。そこで，情報システム部の Web サイト担当者 Z 氏に対し，Web サイトを停止して調査するよう指示した。

Z 氏の調査の結果，D 社の Web サイトにおいて，会員が利用者 ID とパスワードの入力を行うログインの処理に不備があり，外部から SQL インジェクション攻撃を受けていたことが判明した。

〔セキュリティ事故への対応〕

Z 氏は，一般的な Web サイトにおけるセキュリティ事故に関して考えられる対策と対応を調査し，今回のセキュリティ事故において会員と Web サイトに対して必要と思われる対策と対応を表 1 にまとめて，Y 部長に報告した。

表 1　セキュリティ事故の対策と対応の概要

対象	対策と対応
会員	①　全ての会員に対する謝罪 ②　事故の公表と被害状況の説明 ③　会員への事故対応の依頼
Web サイト	④　被害状況の把握及び原因の特定 ⑤　SQL インジェクション攻撃を防ぐための Web サイトの改修 ⑥　Web サイトへのアクセスの常時監視 ⑦　ネットワークを介した攻撃によるネットワークアクセス負荷上昇に対応するためのネットワーク回線の二重化

Y 部長は，表 1 の⑦は実施を見合わせ，Z 氏に①～⑥の実施を急がせるとともに，更なる情報セキュリティ対策の実施を指示した。

設問1　今回受けた SQL インジェクション攻撃に関する記述として適切な答えを，解答群の中から選べ。

解答群

ア　攻撃者が DNS に登録されたドメインの情報を改ざんすることによって，利用者をフィッシングサイトに誘導し，そこで入手した利用者 ID とパスワードを用いて，データベースを不正に操作した。

イ　攻撃者が，D 社の Web サイトの入力項目に対し，命令文を送り込むことによって，データベースを不正に操作した。

ウ　攻撃者が D 社のデータベースの管理ツールを入手し，管理ツール経由で直接 D 社のデータベースを不正に操作した。

エ　攻撃者がネットワーク上で情報の盗聴を行い，D 社のデータベースの管理者の ID とパスワードを入手し，データベースを不正に操作した。

設問2　表 1 中の③に関して，今回のセキュリティ事故の対応として適切な答えを，解答群の中から選べ。

解答群

ア　安易なパスワードの設定を防止するために，パスワードは英字，数字，記号が混在する 8 文字以上のものにするよう会員に依頼する。

イ　攻撃を受けた場合の被害を抑えるために，メールマガジン購読だけを利用する会員の会員情報を格納したデータベースと商品の購入を行う会員の会員情報を格納したデータベースとを分離し，商品の購入を行う会員だけには，利用者 ID 及びパスワードの変更を依頼する。

ウ　個人情報の目的外利用を避けるために，D 社が取得する個人情報の利用目的に事故の対応を追加し，同意を会員に依頼する。

エ　クレジットカード情報が漏えいしている場合の不正利用を防止するために，登録されたクレジットカードの停止及び番号変更の手続を会員に依頼する。

設問3　表1中の⑦に関して，Y部長が実施を見合わせた理由として適切な答えを，解答群の中から選べ。

解答群

ア　Webサーバの増設が必要となる。

イ　稼働中のサービスの停止が必要であり，事業への影響が大きい。

ウ　今回のSQLインジェクション攻撃を防ぐ対策にならない。

エ　セキュリティ事故発生時に攻撃者の侵入経路の特定に時間が掛かる。

設問4　Y部長は，Z氏に事故の再発防止のために更なる情報セキュリティ対策の実施を指示した。次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

〔SQLインジェクション攻撃への追加対策〕

SQLインジェクション攻撃は，システム開発の際にセキュリティを考慮した設計及び実装を行うことで回避できる。例えば，［　a　］ことで，アプリケーション開発時に脆弱性が作り込まれる可能性を減らすこととする。

〔会員情報に対するその他のセキュリティ対策〕

会員情報を格納したデータベースサーバへの不正アクセス対策として，［　b　］こととする。また，情報漏えいが発生した場合の原因の分析や犯人の追跡を行うための証拠の確保には，［　c　］を行うこととする。

a に関する解答群

ア　開発担当者と運用担当者の職務を分離する

イ　開発用の端末と通常利用の端末を分離する

ウ　瑕疵(かし)の発生に備えた保険に加入する

エ　機密保持に関する誓約書を作成する

オ　セキュアプログラミングのルールを作成する

カ　負荷分散装置を設置する

b に関する解答群

ア　Web サーバとデータベースサーバの時刻を同期させる

イ　会員情報を暗号化する

ウ　社内からのインターネット利用時にフィルタリングを実施する

エ　共有 ID を利用する

オ　データベースサーバを RAID 構成にする

c に関する解答群

ア　アクセスログやエラーログの保管

イ　外部記憶媒体の利用禁止を明文化

ウ　業界団体との連携によるセキュリティ事故情報の共有

エ　担当する業務に応じた情報セキュリティ教育の実施

オ　内部不正に対する罰則の強化

問5　通信講座を提供している企業の受講管理システムに関する次の記述を読んで，設問1，2に答えよ。

A社では資格を取得するための通信講座を提供している。資格には上級，中級，初級の3種類があり，各講座の修了判定で合格すると合格証が発行され，資格取得となる。上級資格向けの講座を受講するには，中級資格を取得している必要がある。A社通信講座の概要を表1に示す。

表1　A社通信講座の概要

資格区分	受講期間	受講条件
上級資格	12か月	中級資格を取得していること
中級資格	10か月	なし
初級資格	6か月	なし

〔通信講座運用の概要〕

A社では，受講者の受講状況や成績を，受講管理システムを使って管理し，修了判定を行っている。通信講座運用の概要を次に示す。

(1)　各講座は毎月初めに開始する。

(2)　テキストと課題が，受講期間中の毎月初めに受講者に到着するように送付される。

(3)　学習後の受講者から，課題に対する答案がA社に提出される。答案の提出期限は，受講者が課題を受け取った月の25日とする。

(4)　A社に答案が到着した日を提出日として，受講管理システムに入力する。

(5)　到着から3日以内に，答案を添削して100点満点で採点し，添削済み答案と模範解答を受講者に返送する。

(6)　点数と返送日を受講管理システムに入力する。

提出された答案の処理の流れを，図1に示す。

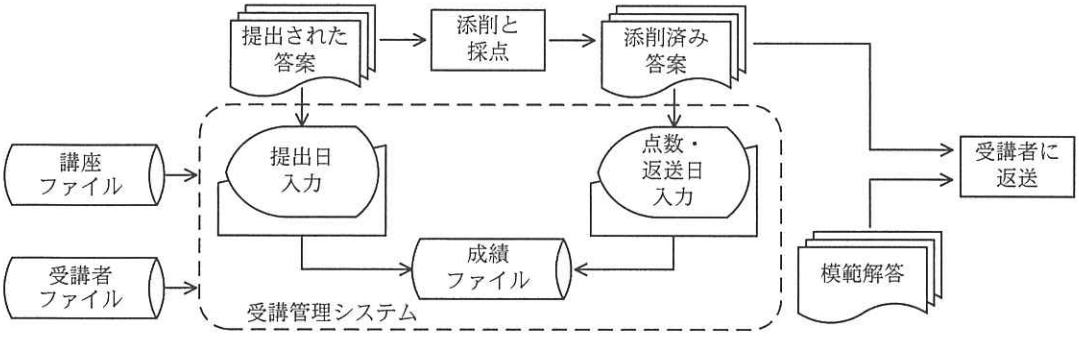


図１　提出された答案の処理の流れ

〔初級，中級資格向けの講座の修了判定処理の概要〕

(1)　毎月初めに，前月で受講期間が終了した受講者を対象に，答案の提出回数と平均点を算出し，修了判定処理を行う。平均点は，全ての答案の合計点を受講期間の月数で除算して求める。答案が未提出の場合，及び提出期限を過ぎて答案が到着した場合は，提出回数に含めず，点数は０点とする。

(2)　判定区分には，優秀，合格，不合格の３種類がある。

(3)　初級資格向けの講座では，4 回以上答案を提出し，かつ，平均点が 60 点以上の受講者を合格と判定する。

(4)　中級資格向けの講座では，7 回以上答案を提出し，かつ，平均点が 60 点以上の受講者を合格と判定する。

(5)　全ての答案を提出し，かつ，平均点が 90 点以上の受講者は優秀と判定する。

(6)　(3)，(4)の条件を満たさない受講者は，不合格と判定する。

(1)～(6)の処理は，成績ファイルと講座ファイルを修了判定プログラムに入力して行われる。

〔初級，中級資格向けの講座の修了判定後の処理の概要〕

(1)　合格者には合格証を発行する。

(2)　優秀者には優秀者用の合格証を発行する。優秀者は，修了判定した月の翌月 1 日（以下，起算日という）から 24 か月間，上位の講座（初級資格向けの講座の場合は中級資格向けの講座，中級資格向けの講座の場合は上級資格向けの講座）を半額で受講することができる。優秀者には優秀者用の合格証と併せて割引受講案内を送付する。

(3) 不合格者には，起算日から 12 か月間，同じ講座を半額で再受講することができる割引受講案内を送付する。

(1)～(3)の処理は，修了判定処理の結果に基づき，受講者ファイルと成績ファイルを参照して行われる。

受講者ファイル，講座ファイル，成績ファイルのレコード様式を，図 2 ～ 4 に示す。各ファイルは，全て索引順編成ファイルである。初級，中級資格向けの講座の修了判定プログラムの流れを図 5 に，主なモジュールの処理内容を表 2 に示す。

受講者コード	受講者名	住所	電話番号

注記　下線はキー項目を表す。

図 2　受講者ファイルのレコード様式

講座番号	講座名称	開始年月日	終了年月日	受講期間の月数	答案 1 の提出期限	…	答案 n の提出期限

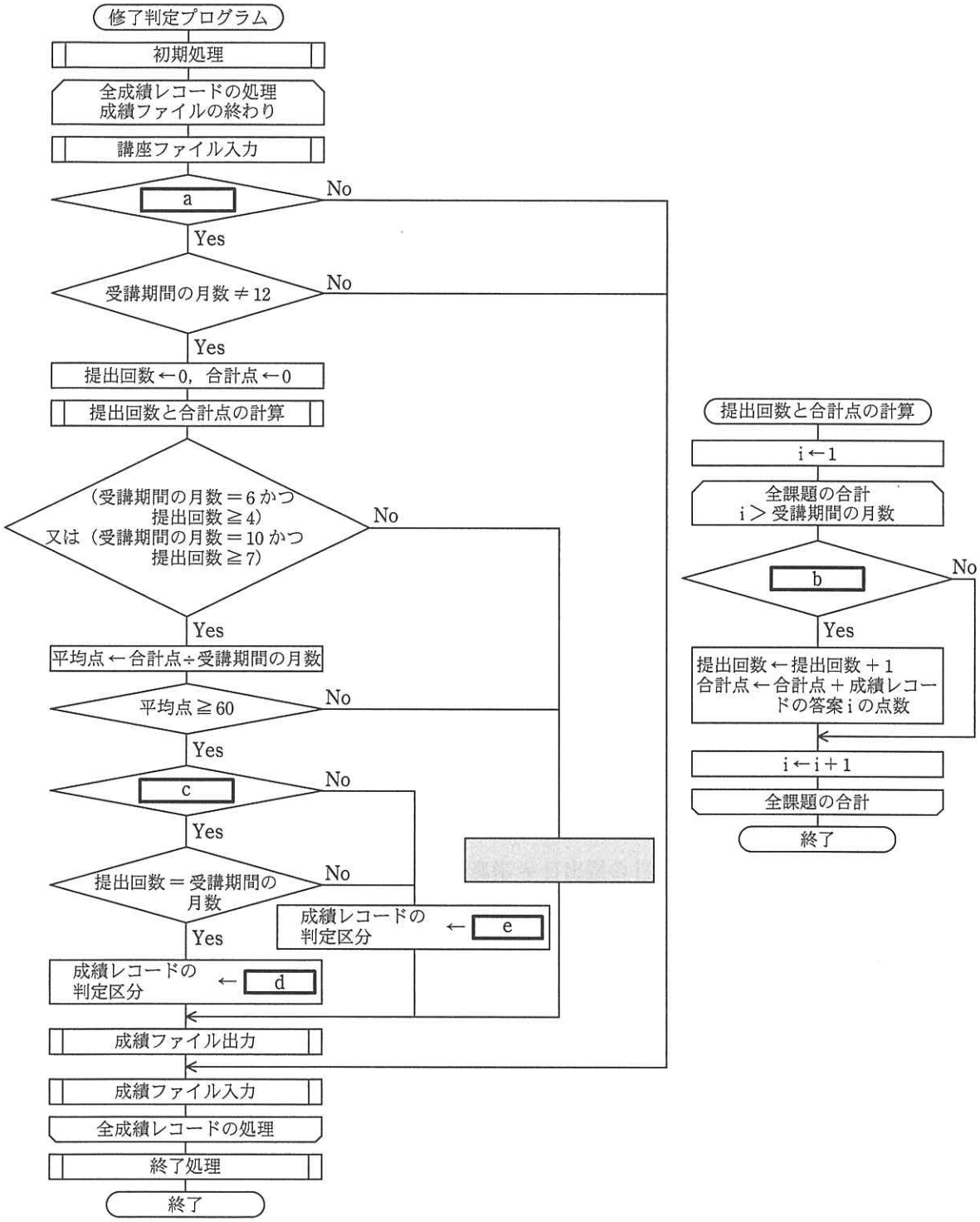
注記　下線はキー項目を表す。

図 3　講座ファイルのレコード様式

受講者コード	講座番号	判定区分	起算日	答案 1			…	答案 n		
				提出日	点数	返送日		提出日	点数	返送日

注記 1　下線はキー項目を表す。
注記 2　判定区分の初期値には空白が設定されている。
注記 3　提出日の初期値には提出期限の翌日の日付が設定されている。
注記 4　点数の初期値には 0 点が設定されている。

図 4　成績ファイルのレコード様式



注記　網掛けの部分は表示していない。

図５　初級，中級資格向けの講座の修了判定プログラムの流れ

表2　主なモジュールの処理内容

モジュール名	処理内容
初期処理	各ファイルを開く（成績ファイルは順次アクセスする）。 成績ファイルを読む（判定区分が空白以外のレコードは，読み飛ばす）。
講座ファイル入力	成績レコードの講座番号をキーとして，講座ファイルを読む。
成績ファイル出力	成績レコードを，成績ファイルに書き込む。
成績ファイル入力	成績ファイルを読む（判定区分が空白以外のレコードは，読み飛ばす）。
終了処理	各ファイルを閉じる。

設問1　図5中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，日付については早い方が小さい数として扱われる。

aに関する解答群

ア　講座レコードの開始年月日＜現在日付

イ　講座レコードの答案iの提出期限＜現在日付

ウ　講座レコードの終了年月日＜現在日付

エ　成績レコードの答案iの提出日＜現在日付

オ　成績レコードの答案iの返送日＜現在日付

bに関する解答群

ア　成績レコードの答案iの提出日＝講座レコードの答案iの提出期限

イ　成績レコードの答案iの提出日≠講座レコードの答案iの提出期限

ウ　成績レコードの答案iの提出日＞講座レコードの答案iの提出期限

エ　成績レコードの答案iの提出日≦講座レコードの答案iの提出期限

オ　成績レコードの答案iの提出日の前日＞講座レコードの答案iの提出期限

cに関する解答群

ア　平均点＝60　　イ　平均点＜60　　ウ　60＜平均点＜90

エ　平均点＞90　　オ　平均点≧90

d，eに関する解答群

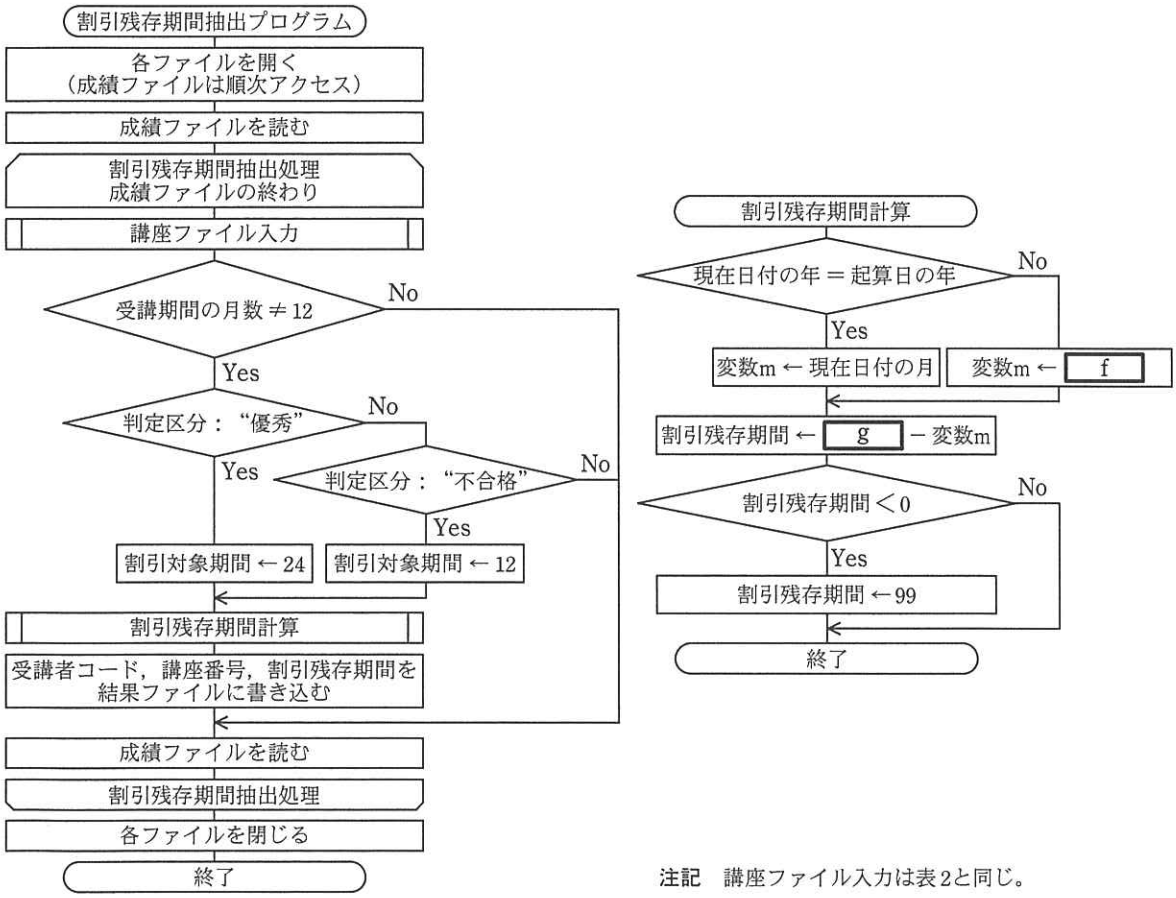
ア　“合格”　　イ　“不合格”　　ウ　“優秀”

設問2　割引を使用した受講の利用率を向上させるために，割引対象期間の残り月数（以下，割引残存期間という）が少なくなっている受講者に，ダイレクトメールを発送することになった。割引を使用した受講の利用状況を調べるために，割引対象受講者と割引残存期間を全て抽出し，結果ファイルに書き込む，割引残存期間抽出プログラムを作成する。割引対象期間を過ぎている場合，割引残存期間に99を入れる。結果ファイルのレコード様式を図6に，割引残存期間抽出プログラムの流れを図7に示す。［　　］に入れる正しい答えを，解答群の中から選べ。

受講者コード	講座番号	割引残存期間

注記　下線はキー項目を表す。

図6　結果ファイルのレコード様式



注記　講座ファイル入力は表2と同じ。

図7　割引残存期間抽出プログラムの流れ

fに関する解答群

ア　起算日の月＋(起算日の年－現在日付の年)×12

イ　起算日の月＋(現在日付の年－起算日の年)×12

ウ　現在日付の月＋(起算日の年－現在日付の年)×12

エ　現在日付の月＋(現在日付の年－起算日の年)×12

オ　現在日付の月－(現在日付の年－起算日の年)×12

gに関する解答群

ア　起算日の月

イ　起算日の月－割引対象期間

ウ　割引対象期間

エ　割引対象期間＋起算日の月

オ　割引対象期間－起算日の月

問6　データ管理に関する次の記述を読んで，設問1～3に答えよ。

大手の販売会社 X 社では，取引履歴などの大量のデータを一定期間保存するために，約 1,000 本の磁気テープ（以下，テープという）を使用している。テープは，電算センタ内のテープライブラリに保管してあり，システム部の運用課が管理している。テープには固有番号を付け，台帳で管理している。

テープは3か月ごとに棚卸しをして，台帳どおりの場所に実在していることを確認する。棚卸しの手続の概要は，次のとおりである。

(1)　今回の棚卸実施日に存在すべきテープの本数（今回の棚卸本数）を，次の式で求める。

今回の棚卸本数 ＝ 前回の棚卸本数 ＋ 追加した本数 － 廃棄した本数
＋ 搬入した本数 － 搬出した本数

式中の各項の概要は，次のとおりである。

①　追加した本数：新規に購入した本数。未使用テープ保管箱に保管する。

②　廃棄した本数：不要となって記録内容を消去して廃棄した本数。不要になったテープは，使用済みテープ保管箱に保管しておき，不定期にまとめて記録内容を消去する。消去が完了した時点で，廃棄した本数として計上する。

③　搬入した本数：社外の保管場所から受け入れた本数。例えば，外部保管業者に保管を委託してあったテープを取り寄せた場合などがこれに該当する。

④　搬出した本数：社外の保管場所へ受け渡した本数。例えば，外部保管業者にテープの保管を委託した場合などがこれに該当する。X 社内の部門からの貸出し要求に基づく貸出しは，これに該当しない。

(2)　棚卸作業では，テープの固有番号を基に，テープライブラリ内で実際にテープの有無を確認する。ただし，現在コンピュータ室で使用中又は社内部門に貸出中のものは，それぞれの現場に出向いて有無を確認する。

X社の前回の棚卸結果報告書を図1に示す。また，前回の棚卸実施後，今回の棚卸実施直前までのテープの移動状況を図2に示す。

宛先：システム部長　　報告日：2012年7月18日

磁気テープ棚卸結果報告書

(1) 棚卸実施日：2012年7月13日

(2) 移動記録：

前回の棚卸本数	1067		
（＋）追加した本数	20		
（－）廃棄した本数	15		
（＋）搬入した本数	12		
（－）搬出した本数	24		
今回の棚卸本数	1060	（合　計）	1060

(3) 実地確認済み本数：

所定テープラック内	1036
未使用テープ保管箱	12
使用済みテープ保管箱	8
社内部門（貸出中）	4

(4) 貸出先：調査部（2本），監査室（2本）。

棚卸実施者：ライブラリアン　○○ ○○，立会者　△△ △△

図1　前回の棚卸結果報告書

月-日	移動内容
07-17	外部保管業者に保管を委託するためテープ6本を出庫・配送
07-25	不要となったテープ5本を使用済みテープ保管箱に移動
08-15	使用済みテープ保管箱内の全テープについて記録内容を消去
08-28	調査部にテープ1本を貸出し
09-12	期末処理用のテープ20本を新規購入
09-21	監査室から貸出中のテープ2本を返却
10-02	不要となったテープ20本を使用済みテープ保管箱に移動
10-06	期末処理のため未使用テープ18本を使用
10-10	外部保管業者に保管を委託するためテープ15本を出庫・配送

図2　棚卸対象テープの移動状況

図1，図2を基に，今回の棚卸しを実施した。現場でテープの有無を確認した結果，調査部に貸し出した，機密情報を記録したテープ1本の所在が不明であった。それ以外のテープは，全て台帳どおりに実在していることが確認できた。これらの事実を基に，今回の棚卸結果報告書を図3のようにまとめた。

図3の棚卸結果報告書をシステム部長に提出した翌日に，所在不明であったテープ1本がシステム部のヘルプデスクで発見された。この貸出しの申請者である調査部員がテープの取扱いに不慣れであったため，部門ファイルサーバへの複写をヘルプデスクに電話で依頼した。複写した後，テープはヘルプデスクの施錠されたキャビネットに置かれたままとなっていた。

宛先：システム部長　　報告日：2012年10月17日

磁気テープ棚卸結果報告書

(1)　棚卸実施日：2012年10月12日

(2)　移動記録：前回の棚卸本数

（＋）追加した本数	20	所定テープラック内	1009
（－）廃棄した本数	a	未使用テープ保管箱	b
（＋）搬入した本数	0	使用済みテープ保管箱	c
（－）搬出した本数	21	社内部門（貸出中）	d
今回の棚卸本数	e	（合　計）	

(3)　実地確認済み本数：

(4)　貸出先：　　。

なお，このうちの１本は所在を確認中。確認後，別途ご報告いたします。

棚卸実施者：ライブラリアン　○○ ○○，立会者　□□ □□

注記　網掛けの部分は表示していない。

図３　今回の棚卸結果報告書

テープは，媒体管理手続に基づいて貸し出され，貸出中のテープは申請者が施錠保管することになっていた。また，テープに記録された機密情報の管理は，全社の情報管理手続に従い，承認された部門の社員だけが複写を許可されている。今回のケースでは，経理部と調査部が複写を許可されていた。

運用課長は，この経緯の詳細を文書にまとめてシステム部長に報告した。報告を受けたシステム部長は，運用課長に次のように指示した。

“このような事故が再発しないように，貸出し後のテープの取扱いについて規定を設けて，申請者に徹底すること。また，今回の件に限らず，図 1～3 を見る限り，返却の手続が十分に機能していないようだ。利用が終わったテープは速やかに返却することという今の規定だけでは不十分で，例えば，①返却予定日を定めて適切な管理を実施するなどの改善が必要ではないか。

今回の所在不明の件での最大の問題は，機密情報保護の点で②根本的な問題を含んでいることだ。貸出方式をやめて，例えば，電算センタで管理するサーバに特別のディレクトリを用意し，そこにテープのデータを複写して提供するなど，抜本的な対策を早急に検討してほしい。”

設問1　図3中の［　　］に入れる正しい答えを，解答群の中から選べ。

a～dに関する解答群

ア　0　　イ　2　　ウ　3　　エ　5　　オ　13

カ　14　　キ　20　　ク　25　　ケ　28

eに関する解答群

ア　1028　　イ　1034　　ウ　1039

エ　1046　　オ　1053　　カ　1054

設問2　本文中の下線①について，実施すべき管理の内容として適切な答えを，解答群の中から選べ。

解答群

ア　運用課が次回の棚卸日よりも後の日付で返却予定日を設定し，棚卸日に超過が発生しないようにする。

イ　運用課が返却予定日を管理し，超過した場合は申請者に再申請などの必要な処置をとらせる。

ウ　申請者に十分な余裕をもった返却予定日を設定させて，返却予定日の超過が発生しないようにする。

エ　申請者に返却予定日の管理を委ね，超過した場合は再申請などの必要な処置を自ら申請させる。

設問3　本文中の下線②について，根本的な問題とは何を指しているか，最も適切な答えを，解答群の中から選べ。

解答群

ア　調査部からの複写の依頼が，電子メールなどの文書ではなく電話で行われた。

イ　複写終了後に，申請者から運用課へ，複写の事後報告をしなかった。

ウ　複写終了後に，ヘルプデスク担当者から運用課へ，複写の事後報告をしなかった。

エ　ヘルプデスク担当者が機密情報を複写し，テープを保管していた。

問7　在庫管理に関する次の記述を読んで，設問1～4に答えよ。

日用品メーカであるZ社の工場では，在庫を次のように管理している。

〔Z社の工場での在庫管理〕

(1)　生産した製品は，品質検査が完了するまで検品中在庫として管理する。製品によっては品質検査に数日掛かるものもある。

(2)　品質検査の結果，良品と判定（以下，良品判定という）された場合は，その製品は良品在庫として管理する。

(3)　品質検査の結果，不良品と判定（以下，不良品判定という）された場合は，その製品は不良品在庫として管理する。

(4)　一度出荷した製品が返品された場合は，その製品は再度品質検査を行うので，検品中在庫として管理する。

(5)　良品在庫の製品が破損した場合は，不良品在庫として管理する。

(6)　不良品在庫の製品は，定期的に廃棄する。

(7)　良品在庫から製品を出荷する。

また，Z社では，1日の生産，品質検査，出荷，返品の受入れ，不良品の廃棄など全ての作業が完了した時点で，次の手順で翌日の出荷に備えた業務を行っている。

手順①　当日の生産数，良品判定数，不良品判定数，返品数，破損数，廃棄数，出荷数，検品中在庫数，良品在庫数及び不良品在庫数を，在庫管理台帳に記入する。また，翌日の出荷予定数を在庫管理台帳に記入する。

手順②　当日の良品在庫数が翌日の出荷予定数に満たない場合は，本社に連絡して出荷日の変更を依頼する。

手順③　当日の在庫数が基準を満たしていない場合は，翌日に一定数を生産する手配を行う。基準は，良品在庫数が翌日の出荷予定数の2倍以上であり，かつ，検品中在庫数と良品在庫数の合計が翌日の出荷予定数の3倍以上であることとする。

表1と表2は，10月20日の手順①の終了時点における，商品Aと商品Bの在庫管理台帳である。

表 1　10 月 20 日の手順①の終了時点における商品 A の在庫管理台帳

	18 日	19 日	20 日	21 日
生産数		1,800	1,800	
良品判定数		1,600	800	
不良品判定数		100	0	
返品数		50	100	
破損数		b	40	
廃棄数		180	0	
出荷数		1,700		1,500 [1)]
検品中在庫数	a	1,150	2,250	
良品在庫数	3,000	2,880	2,240	
不良品在庫数	180	120	c	

注記　網掛けの部分は表示していない。

注 [1)]　出荷予定数を表す。

表 2　10 月 20 日の手順①の終了時点における商品 B の在庫管理台帳

	18 日	19 日	20 日	21 日
生産数		150	0	
良品判定数		180	140	
不良品判定数		10	0	
返品数		20	10	
破損数		30	20	
廃棄数		0	0	
出荷数		160	130	120 [1)]
検品中在庫数	240	220	90	
良品在庫数	280	270	260	
不良品在庫数	30	70	90	

注記　網掛けの部分は表示していない。

注 [1)]　出荷予定数を表す。

設問1　表1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　800	イ　900	ウ　1,000	エ　1,100
オ　1,200	カ　1,300	キ　1,400	ク　1,500

bに関する解答群

ア　0	イ　20	ウ　40	エ　60
オ　200	カ　220	キ　240	ク　260

cに関する解答群

ア　60	イ　80	ウ　160	エ　180
オ　240	カ　260	キ　280	ク　360

設問2　表2の商品Bに関する記述として適切な答えを，解答群の中から選べ。

解答群

ア　19日は，生産数が出荷数を下回ったので，良品在庫数が減少した。

イ　19日は，生産数と破損数の合計が良品判定数と不良品判定数の合計を下回ったので，検品中在庫数が減少した。

ウ　19日も20日も廃棄数が0だったので，良品在庫数が減少した。

エ　20日は，不良品判定数と廃棄数は0であったが返品があったので，不良品在庫数が増加した。

オ　20日は，返品が少量あったものの生産がなかったので，品質検査を行ったことによって検品中在庫数が大きく減少した。

設問3　商品A，Bに関する10月20日の手順①の終了後の，翌日の出荷に備えた業務の組合せとして正しい答えを，解答群の中から選べ。

解答群

	商品A	商品B
ア	21日の生産を手配する	21日の生産を手配する
イ	21日の生産を手配する	なし
ウ	21日の生産を手配する	本社に連絡する
エ	なし	21日の生産を手配する
オ	なし	なし
カ	なし	本社に連絡する
キ	本社に連絡する	21日の生産を手配する
ク	本社に連絡する	なし
ケ	本社に連絡する	本社に連絡する

設問4　在庫削減の取組みに関する次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

Z社では余剰在庫が問題と捉えており，キャッシュフローと棚卸資産回転率の改善のために，在庫を削減したいと考えている。良品在庫と検品中在庫の削減は，［ d ］ことになる。そこで企画課は，在庫削減の施策として［ e ］ことを検討した。

dに関する解答群

ア　キャッシュフローと棚卸資産の両方を減少させる

イ　キャッシュフローと棚卸資産の両方を増加させる

ウ　キャッシュフローを減少させ，棚卸資産を増加させる

エ　キャッシュフローを増加させ，棚卸資産を減少させる

eに関する解答群

ア　手順②で良品在庫数が不足したときは，検品中在庫からも出荷できるように変更する

イ　手順③で翌日の生産の手配を判断する基準の倍率を下げる

ウ　手順③で良品在庫数と検品中在庫数が基準を満たさなかったときに手配する翌日の生産数を増加する

エ　廃棄の頻度を上げる

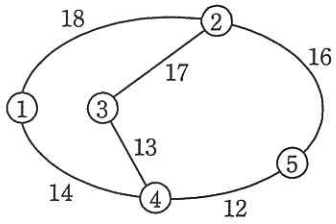
オ　返品があったときは，検品中在庫として管理するのではなく，良品在庫として管理する

カ　良品在庫の製品が破損したときは，不良品在庫として管理するのではなく，検品中在庫として管理する

次の問8は必須問題です。必ず解答してください。

問8　次のプログラムの説明及びプログラムを読んで，設問に答えよ。

鉄道の路線がある。駅数 N は 2 以上で，各駅には駅番号（1，2，…，N）が付いている。どの任意の2駅間にも，それらを結ぶ経路が一つ以上存在する。また，隣接する2駅間を直接結ぶ経路は一つだけ存在し，その距離が与えられている。図1に5駅からなる鉄道の路線例を示す。



注記　○は駅を表し，その中の数字は駅番号を表す。
また，経路に付した数字は2駅間の距離を表す。

図1　鉄道の路線例（1）

〔プログラムの説明〕

副プログラム CalcDist は，駅数 N 及び要素数 N×N の2次元配列 Dist を受け取り，プログラムに示すアルゴリズム（Warshall-Floyd 法）によって，配列 Dist の内容を更新しながら各駅間の最短距離を求めていく。実行が終わると，任意の2駅 i，j 間の最短距離が Dist[i][j] に求められている。どの 2 駅間についても，その最短距離は 999 より小さいものとする。配列 Dist の添字は1から始まる。

副プログラム CalcDist に渡す配列 Dist には，次の値を格納する。

なお，図1の路線情報を格納した配列 Dist の内容を，図2の初期値に示してある。

(1)　2駅 i，j が隣接しているとき，Dist[i][j] 及び Dist[j][i] には，その駅間距離を格納する。例えば，図1の駅①と②は隣接しているので，Dist[1][2] 及び Dist[2][1] には，18 を格納する。

(2)　2駅 i，j が隣接していないとき，Dist[i][j] 及び Dist[j][i] には，未確定を表す距離 999 を格納する。例えば，図 1 の駅①と③は隣接していないので，Dist[1][3] 及び Dist[3][1] には，999 を格納する。

(3)　各駅 i について，Dist[i][i] には 0 を格納する。

副プログラム CalcDist 中の Print(N, Dist) は，配列 Dist のその時点の内容を印字する副プログラムである。

〔プログラム〕

```
○副プログラム：CalcDist(整数型：N, 整数型：Dist[][])
○整数型：From, To, Via
• Print(N, Dist)
■ Via: 1, Via ≦ N, 1
  ■ From: 1, From ≦ N, 1
    ■ To: 1, To ≦ N, 1
      ▲ Dist[From][To] ＞ Dist[From][Via] ＋ Dist[Via][To]
        • Dist[From][To] ← Dist[From][Via] ＋ Dist[Via][To]
    ■
  ■
  • Print(N, Dist)
■
```

図1の路線情報を配列 Dist に格納して，CalcDist を実行した。配列 Dist の内容の変化を Print(N, Dist) で印字した結果は，図2のとおりであった。

初期値

```
 0  18 999  14 999
18   0  17 999  16
999  17   0  13 999
14 999  13   0  12
999  16 999  12   0
```

Via ＝ 1

```
 0  18 999  14 999
18   0  17  32  16
999  17   0  13 999
14  32  13   0  12
999  16 999  12   0
```

Via ＝ 2

```
 0  18 [a] 14  34
18   0  17  32  16
[a] 17   0  13 [b]
14  32  13   0  12
34  16 [b] 12   0
```

Via ＝ 3

```
 0  18  35  14  34
18   0  17  30  16
35  17   0  13  33
14  30  13   0  12
34  16  33  12   0
```

Via ＝ 4

```
 0  18  27  14  26
18   0  17  30  16
27  17   0  13  25
14  30  13   0  12
26  16  25  12   0
```

Via ＝ 5

```
 0  18  27  14  26
18   0  17  28  16
27  17   0  13  25
14  28  13   0  12
26  16  25  12   0
```

注記　“初期値”，“Via ＝ 1” などは，説明のために付加してある。

図2　配列 Dist の内容の変化を印字した結果

駅数 N に関して，副プログラム CalcDist の計算量のオーダは［ c ］，またメモリ使用量のオーダは［ d ］である。そこで，駅数 N が大きい場合に，計算量やメモリ使用量を減らすための方法を考える。

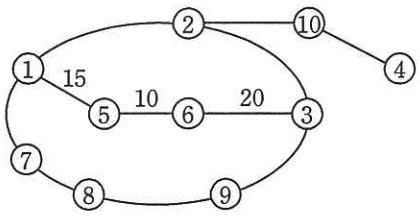
例えば，配列 Dist の対称性に着目し，プログラム中の［　　］の部分を，

```
■ From: 1, From ≦ N - 1, 1
  ■ To: [ e ]
    ▲ Dist[From][To] ＞ Dist[From][Via] + Dist[Via][To]
      • Dist[From][To] ← Dist[From][Via] + Dist[Via][To]
      • Dist[To][From] ← Dist[From][Via] + Dist[Via][To]
    ▼
  ■
■
```

とすれば，繰返し処理中の選択処理の実行回数を約半分に減らすことができる。

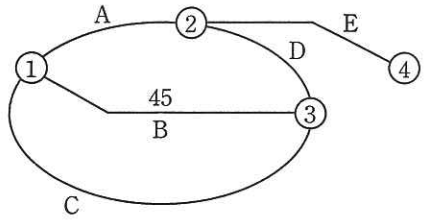
次に，少ないメモリ使用量で，任意の 2 駅 i，j 間の最短距離を求める方法を考える。

駅を，乗換駅（3 方向以上に隣接する駅がある），終端駅（1 方向だけに隣接する駅がある），中間駅（2 方向だけに隣接する駅がある）の 3 種類に分ける。乗換駅と終端駅を合わせて基幹駅と呼ぶ。基幹駅の数を K として，駅番号 1～K が基幹駅，駅番号 K+1～N が中間駅となるように駅番号を定める。ここで，K≧2 であるとする。図 3 の路線例では，駅①～③が乗換駅，駅④が終端駅，駅⑤～⑩が中間駅である。



注記　2 駅間の距離は一部だけを表示している。

図 3　鉄道の路線例（2）



注記　A～E は区間名を表す。

図 4　鉄道の路線例（2）の基幹駅と区間名

各駅間の最短距離は，次の方法で求める。

(1)　中間駅を全て除いて，基幹駅だけからなる路線を考え，二つの基幹駅を結ぶ経路ごとに一意の区間名を付ける。図 3 の路線例にこの操作を加えたものが図 4 である。例えば，図 3 の経路①－⑤－⑥－③は，図 4 の区間 B に対応する。

(2) 要素数 K×K の 2 次元配列 Dist を用意し，事前に配列 Dist に各基幹駅間の最短距離を求めておく。

(3) 全ての駅について，表 1 に示す形式の駅情報表を用意する（表 1 には，図 3 の駅情報の一部が示してある）。

中間駅の場合，Sec はその駅が属する区間の区間名，KL，KH はその駅が属する区間の両端の駅番号（KL ≦ KH とする），ToKL，ToKH はその駅から駅 KL，KH までの距離である。基幹駅の場合，その駅は自分自身の駅を両端とする距離 0 の区間に属すると考えて，表 1 の例のように中間駅と同様の情報を用意する。

これらの情報は，駅番号を添字として参照する。例えば，駅番号 5 の ToKL の値は ToKL[5] として参照する。

表 1　駅情報表の形式

駅番号	Sec	KL	ToKL	KH	ToKH
1	“1”	1	0	1	0
2	“2”	2	0	2	0
3	“3”	3	0	3	0
4	“4”	4	0	4	0
5	“B”	1	15	3	30
6	“B”	1	25	3	20
⋮	⋮	⋮	⋮	⋮	⋮

(4) 2 駅 i，j 間の最短距離を求める。一般に，両駅が属する区間が異なる場合，駅 i から駅 j へ行く経路は，

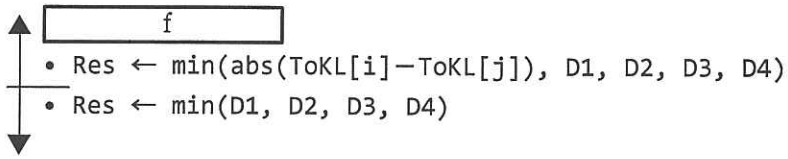
（駅 i）→（駅 i が属する区間のいずれか一端の基幹駅）

→（駅 j が属する区間のいずれか一端の基幹駅）→（駅 j）

となる。したがって，次の式で求める 4 通りの値 D1，D2，D3，D4 のうちの最小値が，2 駅 i，j 間の最短距離となる。

- D1 ← ToKL[i] + Dist[KL[i]][KL[j]] + ToKL[j]
- D2 ← ToKL[i] + Dist[KL[i]][KH[j]] + ToKH[j]
- D3 ← ToKH[i] + Dist[KH[i]][KL[j]] + ToKL[j]
- D4 ← ToKH[i] + Dist[KH[i]][KH[j]] + ToKH[j]

両駅が属する区間が同じ場合は，区間の端の基幹駅を経由せずに直接駅 i から駅 j へ行く経路がある。これも考慮すると，任意の2駅 i，j 間の最短距離 Res は次の処理で求められる。ここで，関数 min(式$_1$，式$_2$，…) は，式$_1$，式$_2$，… の値の最小値を返す関数であり，関数 abs(式) は，式の値の絶対値を返す関数である。



この方法なら，2次元配列 Dist の要素数を N×N から K×K に削減できる。ただし，表1に示す表が増えるので，例えば K=5 のとき，配列及び表が占めるメモリ使用量を削減できるのは N≧［ g ］の場合となる。ここで，表1に示す表は，1駅について配列 Dist の要素5個分のメモリを使用するものとする。

設問　本文中及び図2中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，b に関する解答群

ア　33　　イ　34　　ウ　35　　エ　999

c，d に関する解答群

ア　$O(N)$　　イ　$O(N \log N)$　　ウ　$O(N^2)$　　エ　$O(N^3)$

e に関する解答群

ア　1, To ≦ From, 1　　イ　1, To ≦ From + 1, 1

ウ　From, To ≦ N − 1, 1　　エ　From + 1, To ≦ N, 1

fに関する解答群

ア　(KL[i] ＝ KL[j]) and (KH[i] ＝ KH[j])

イ　(KL[i] ≠ KL[j]) or (KH[i] ≠ KH[j])

ウ　Sec[i] ＝ Sec[j]

エ　Sec[i] ≠ Sec[j]

gに関する解答群

ア　6　　イ　8　　ウ　9　　エ　14

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，選択した問題については，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問以上マークした場合には，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。

C

〔プログラムの説明〕

関数 check_winning_lot は，複数本のくじのくじ番号を当選番号と比較し，当たりを確認するプログラムである。

(1) くじ番号は，6桁の数字から成る文字列である。数字だけから成る文字列を数字列という。

(2) くじの当たりは1～6等の6等級から成り，図1に示すように各等級には一つ以上の当選番号がある。各等級での当選番号の桁数と個数は異なる。

等級	当選番号
1等	223692
2等	141421，314159
3等	下5桁　27182，00145
4等	下4桁　3301，1028
5等	下3桁　243，101，144
6等	下2桁　03，92

図1　当選番号の例

(3) 当選番号は6桁以下の数字列であり，くじ番号の最下位桁から当選番号の桁数だけ取り出した数字列が当選番号と一致したくじが当たりとなる。

なお，くじ番号によっては，複数の等級の当たりとなる場合もある。図1の例では，くじ番号“223692”のくじは，1等と6等の当たりとなる。

(4) 関数 check_winning_lot の引数は次のとおりである。ここで，引数の値に誤りはないものとする。

lots　　確認するくじのくじ番号表
win_list　当選番号表
winner　　当選本数表

(5) 確認するくじのくじ番号表 lots の要素 lots[i]には，くじ番号の数字列へのポインタが格納されている。最後の要素 lots[Nl]（Nl は，確認するくじの本数）には，空ポインタ定数（NULL）が格納されている。

(6) 当選番号表 win_list の要素 win_list[i]には，i+1 等の番号表へのポインタが格納されている。win_list[i]が指す i+1 等の番号表の要素 win_list[i][j]には，当選番号の数字列へのポインタが格納されている。最後の要素 win_list[i][Ni]（Ni は，i+1 等の当選番号の個数）には，図2に示すように空ポインタ定数が格納されている。

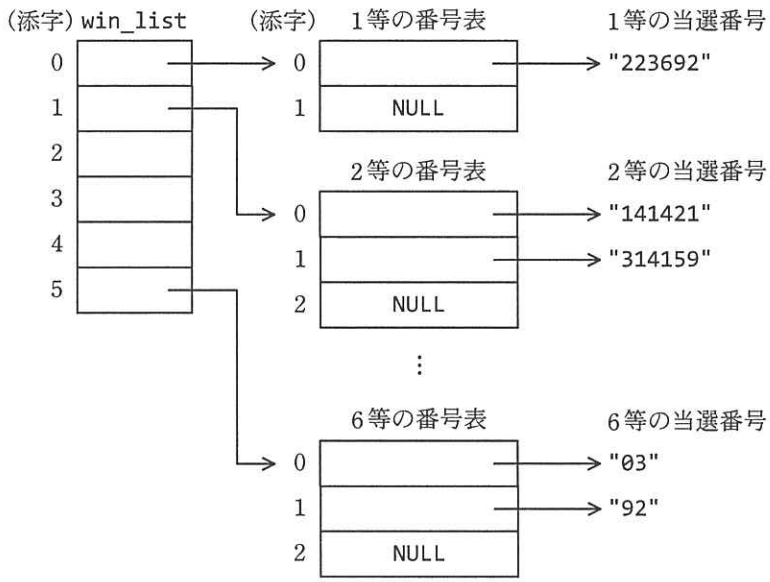


図2　当選番号表の例

(7) 当選本数表 winner の要素 winner[i]には，i+1 等の当たりのくじの本数が格納される。

(8) 関数 check_winning_lot で使用している関数 check_lot と strlen の仕様は次のとおりである。

```
int check_lot(char *lot, char *win);
```

機能：　くじ番号と当選番号を，最下位桁から1桁ずつ順に当選番号の桁数分だけ比較し，一致するかどうかを調べる。

引数：　lot　くじ番号

　　win　当選番号

返却値：確認結果（0：一致しない，1：一致する）

```
unsigned int strlen(const char *s);
```

機能：　文字列の長さを求める。

引数：　s　文字列

返却値：終端を示すナル文字に先行する文字の個数

〔プログラム〕

(行番号)

```
 1 #include <stdlib.h>
 2 #include <string.h>

 3 #define LOT_DNUM 6   /* くじの桁数 */
 4 #define LOT_GNUM 6   /* くじの等級数 */

 5 void check_winning_lot(char *[], char **[LOT_GNUM],
 6                         int [LOT_GNUM]);
 7 int check_lot(char *, char *);

 8 void check_winning_lot(char *lots[],
 9                         char **win_list[LOT_GNUM],
10                         int winner[LOT_GNUM]) {
11     int  i, j, k;

12     for (i = 0; i < LOT_GNUM; i++) {
13         winner[i] = 0;
14         for (j = 0; lots[j] != NULL; j++) {
15             for (k = 0;    a    ; k++) {
16                 winner[i] += check_lot(lots[j], win_list[i][k]);
17             }
18         }
19     }
20 }

21 int check_lot(char *lot, char *win) {
22     int  lenw, result = 1, i;

23     lenw = strlen(win);
24     win += lenw;
25     lot    b    ;
26     for (i = 0;    c    ; i++) {
27         if (*(--lot) != *(--win)) {
28             result = 0;
29         }
30     }
31     return result;
32 }
```

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　k < LOT_DNUM　　イ　k <= LOT_DNUM

ウ　win_list[i][k] == NULL　　エ　win_list[i][k] != NULL

オ　*win_list[i][k] == NULL　　カ　*win_list[i][k] != NULL

b に関する解答群

ア　= lenw　　イ　= LOT_DNUM

ウ　+= lenw　　エ　+= LOT_DNUM

c に関する解答群

ア　(i < lenw) && (result == 0)

イ　(i < lenw) && (result == 1)

ウ　(i < LOT_DNUM) && (result == 0)

エ　(i < LOT_DNUM) && (result == 1)

オ　result == 0

カ　result == 1

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。ただし，f1 と f2 に入れる答えは，f に関する解答群の中から組合せとして正しいものを選ぶものとする。

くじの当たりを増やすために，6 桁の当選番号に対して前後賞が用意されることになった。これに合わせて関数 check_winning_lot に，前後賞の当選結果を自動判定する処理を追加する。ここで，プログラム中の［ a ］～［ c ］には正しい答えが入っているものとする。

(1)　関数 check_winning_lot の引数を次のとおりに変更する。ここで，引数の値に誤りはないものとする。

lots　　確認するくじのくじ番号表

win_list　当選番号表

winner　　当選本数表

special　　前後賞の当選本数表

(2) 前後賞は，6桁の当選番号に対して用意される。例えば，当選番号“123456”に対する前後賞の番号は，“123455”と“123457”となる。当選番号“000000”に対する前後賞の番号は“999999”と“000001”，当選番号“999999”に対する前後賞の番号は“999998”と“000000”とする。

(3) 前後賞の当選結果の確認は，確認するくじのくじ番号を数値とみなし，それよりも1だけ少ない数値と多い数値を表す数字列を作成し，その数字列を当選番号と比較することによって行う。このとき，くじ番号“000000”よりも1だけ少ない数値を表す数字列は“999999”に，くじ番号“999999”よりも1だけ多い数値を表す数字列は“000000”にする。

(4) 前後賞の当選本数表の要素special[i]にはi+1等の前後賞の当たりのくじの本数が格納される。

(5) 処理の追加に対応するために，プログラムを表1のとおりに変更する。

表1　プログラムの変更内容

処置	変更内容
行番号5，6を変更	`void check_winning_lot(char *[], char **[LOT_GNUM],` `int [LOT_GNUM], int [LOT_GNUM]);`

行番号8～10を変更	`void check_winning_lot(char *lots[],`
	`char **win_list[LOT_GNUM],`
	`int winner[LOT_GNUM], int special[LOT_GNUM]) {`

行番号11と12の間に追加	`int pflg, nflg, m;` `char pnum[LOT_DNUM + 1], nnum[LOT_DNUM + 1];`
行番号13と14の間に追加	`special[i] = 0;`

行番号14と15の間に追加	`/* 前後賞の当選確認用数字列の作成 */`
	`pflg = nflg = 0;`
	`for (m = [ d ]; m >= 0; m--) {`
	`pnum[m] = nnum[m] = [ e ];`
	`if (pflg == 0) {`
	`if (pnum[m] == [ f1 ]) {`
	`pnum[m] = [ f2 ];`
	`} else {`
	`pnum[m]--;`
	`pflg = 1;`
	`}`

	}
	if (nflg == 0) {
	if (nnum[m] == [f2]) {
	nnum[m] = [f1];
	} else {
	nnum[m]++;
	nflg = 1;
	}
	}
	}
	pnum[LOT_DNUM] = nnum[LOT_DNUM] = '\0';

行番号 16 と 17 の間に追加	if (strlen(win_list[i][k]) == LOT_DNUM) {
	special[i] += check_lot(pnum, win_list[i][k]);
	special[i] += check_lot(nnum, win_list[i][k]);
	}

C

dに関する解答群

ア i	イ j	ウ LOT_DNUM
エ LOT_DNUM + 1	オ LOT_DNUM + i	カ LOT_DNUM + j
キ LOT_DNUM - 1	ク LOT_DNUM - i	ケ LOT_DNUM - j

eに関する解答群

ア *lots[j]	イ *lots[j] + m
ウ *lots[j] + m + 1	エ *(lots[j] + m)
オ *(lots[j] + m + 1)	

fに関する解答群

	f1	f2
ア	'0'	'1'
イ	'0'	'9'
ウ	'1'	'0'
エ	'1'	'9'
オ	'9'	'0'
カ	'9'	'1'

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〕

ある会員制スポーツクラブにおける利用料金の計算プログラムである。

このスポーツクラブは，会員が都合の良い時間に施設を利用できるようになっており，会員の利用実績を利用記録ファイルに格納する。利用料金は，1 か月に 10 時間まで利用できる基本料金と，10 時間を超過した時間に応じて課金する超過料金の合計であり，1 か月分をまとめて，翌月，会員に請求する。施設の利用実績がなかった会員にも，基本料金は請求する。

(1)　会員ファイルは，図 1 に示すレコード様式の索引ファイルであり，会員番号を主レコードキーとする。呼出し法は順呼出しとする。

会員番号 6 桁	会員種別 1 桁	個人情報			入会日 8 桁
		氏名 10 桁	住所 20 桁	電話番号 11 桁	

図 1　会員ファイルのレコード様式

①　会員番号は，会員ごとに割り当てられる一意の番号であり，000001 から始まる 6 桁の数字である。

②　会員種別には，表 1 に示す会員種別が格納される。利用料金は，会員種別によって決まる。

表1　料金表

名称	会員種別	基本料金 （月額）	超過料金 （時間当たり）	条件
一般会員	1	5,000 円	500 円	なし
ゴールド会員	2	5,000 円	400 円	5 年以上継続
シルバー会員	3	4,000 円	400 円	60 歳以上
学生会員	4	3,000 円	300 円	学生

③　入会日には，入会した年，月，日が，それぞれ 4 桁，2 桁，2 桁で格納される。

(2) 利用記録ファイルは，図2に示すレコード様式の順ファイルであり，1か月分の会員の利用実績を格納する。1レコードには，1人の会員の1回の利用に関する情報が格納される。

会員番号 6桁	利用日 8桁	開始時刻 4桁	終了時刻 4桁	利用時間（分） 4桁

図2　利用記録ファイルのレコード様式

① 利用日には，施設を利用した年，月，日が，それぞれ4桁，2桁，2桁で格納される。

② 開始時刻及び終了時刻には，時，分がそれぞれ2桁の24時間表記で格納される。施設の営業時間は8時から23時までである。

③ 利用時間には，施設の利用時間が分単位で格納される。

(3) 請求ファイルは，図3に示すレコード様式の順ファイルであり，会員ごとの月次請求金額を格納する。超過料金は1か月の利用時間の合計（分）を時間単位に切り上げて課金する。

会員番号 6桁	利用料金 6桁

図3　請求ファイルのレコード様式

(4) 作業用ファイルは，図4に示すレコード様式の索引ファイルであり，会員番号を主レコードキーとする。利用記録ファイルから会員ごとの利用時間の合計を分単位で求めるために使用する。呼出し法は乱呼出しとする。

会員番号 6桁	合計時間（分） 6桁

図4　作業用ファイルのレコード様式

〔プログラム〕

（行番号）

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD WORK-F.
 4  01 W-REC.
 5     02 W-NO            PIC X(6).
 6     02 W-MIN           PIC 9(6).
 7  FD MEM-F.
 8  01 M-REC.
 9     02 M-NO            PIC X(6).
10     02 M-CODE          PIC 9(1).
11     02 M-PERSON.
12       03 M-NAME           PIC X(10).
13       03 M-ADDR           PIC X(20).
14       03 M-TEL            PIC 9(11).
15     02 M-DATE          PIC 9(8).
16  FD TIME-F.
17  01 T-REC.
18     02 T-NO            PIC 9(6).
19     02 T-DATE          PIC 9(8).
20     02 T-STIME         PIC 9(4).
21     02 T-ETIME         PIC 9(4).
22     02 T-MIN           PIC 9(4).
23  FD BILL-F.
24  01 B-REC.
25     02 B-NO            PIC 9(6).
26     02 B-CHG           PIC 9(6).
27  WORKING-STORAGE SECTION.
28  77 TIME-FLAG       PIC X(1) VALUE SPACE.
29     88  TIME-EOF       VALUE "E".
30  77 MEM-FLAG        PIC X(1) VALUE SPACE.
31     88  MEM-EOF        VALUE "E".
32  77 EXT-H           PIC 9(4).
33  01 PRICE-VALUE.
34     02              PIC X(16) VALUE "5000500040003000".
35     02              PIC X(12) VALUE [      a      ].
36  01 PRICE-TABLE     REDEFINES PRICE-VALUE.
37     02 BASE-PRICE      OCCURS 4 PIC 9(4).
38     02 EXT-PRICE       OCCURS 4 PIC 9(3).
39  PROCEDURE DIVISION.
40  MAIN-PROC.
41      OPEN INPUT  MEM-F TIME-F
42           I-O    WORK-F
43           OUTPUT BILL-F.
44      PERFORM MKTIME-PROC.
45      PERFORM [      b      ].
46      CLOSE MEM-F TIME-F WORK-F BILL-F.
47      STOP RUN.
```

```
48  MKTIME-PROC.
49      PERFORM UNTIL TIME-EOF
50         READ TIME-F AT END     SET TIME-EOF TO TRUE
51                     NOT AT END PERFORM MKWORK-PROC
52         END-READ
53      END-PERFORM.
54  MKWORK-PROC.
55      MOVE T-NO TO W-NO.
56      READ WORK-F
57         INVALID KEY      MOVE T-NO TO W-NO
58                          [     c     ]
59                          WRITE W-REC END-WRITE
60         NOT INVALID KEY  ADD T-MIN TO W-MIN
61                          REWRITE W-REC END-REWRITE
62      END-READ.
63  MKBILL-PROC.
64      PERFORM UNTIL MEM-EOF
65         READ MEM-F
66            AT END     SET MEM-EOF TO TRUE
67            NOT AT END PERFORM WRBILL-PROC
68         END-READ
69      END-PERFORM.
70  WRBILL-PROC.
71      MOVE M-NO TO W-NO B-NO.
72      MOVE BASE-PRICE(M-CODE) TO B-CHG.
73      READ WORK-F
74         INVALID KEY CONTINUE
75         NOT INVALID KEY
76            IF W-MIN > 600 THEN
77               COMPUTE EXT-H = (W-MIN - 600 + 59) / 60
78               [       d       ]
79            END-IF
80      END-READ.
81      WRITE B-REC.
```

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　"300400400500"　　イ　"400400500500"

ウ　"500400400300"　　エ　"500500400300"

b に関する解答群

ア　MAIN-PROC　　イ　MKBILL-PROC

ウ　MKTIME-PROC　　エ　MKWORK-PROC

オ　WRBILL-PROC

c，d に関する解答群

ア　ADD EXT-PRICE(M-CODE) TO B-CHG

イ　ADD T-MIN TO W-MIN

ウ　COMPUTE B-CHG = B-CHG + EXT-PRICE(M-CODE) * EXT-H

エ　COMPUTE W-MIN = (T-MIN + 59) / 60 + 1

オ　MOVE EXT-PRICE(M-CODE) TO B-CHG

カ　MOVE T-MIN TO W-MIN

設問2　テストデータを用いて動作確認をしたところ，利用記録ファイルのレコード数の増加に伴って，処理時間が想定を大きく超えることが判明した。調査の結果，作業用ファイルの入出力に多くの時間を費やしていた。そこで，整列機能を使用して処理時間の短縮を図ることにし，表2の変更を行うことにした。表2のプログラムでは，図4に示すレコード様式の整列併合用ファイルを作業用ファイルとして使用する。表2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，表2の［　d　］には，設問1の正しい答えが入っているものとする。

COBOL

表2　プログラムの変更

処置	変更内容
行番号3を変更	SD WORK-F.
行番号31と32の間に追加	77 WORK-FLAG　　PIC X(1) VALUE SPACE. 　88　WORK-EOF　　VALUE "E". 77 TMP-NO　　PIC 9(6) VALUE ZERO. 77 TMP-MIN　　PIC 9(6) VALUE ZERO.
行番号39から81を変更	PROCEDURE DIVISION. MAIN-PROC. 　　OPEN INPUT　MEM-F TIME-F 　　　　OUTPUT BILL-F. 　　SORT WORK-F ASCENDING KEY W-NO 　　　INPUT　PROCEDURE IS MKTIME-PROC 　　　OUTPUT PROCEDURE IS MKBILL-PROC. 　　CLOSE MEM-F TIME-F BILL-F. 　　STOP RUN.

```
MKTIME-PROC.
    PERFORM UNTIL TIME-EOF
       READ TIME-F
          AT END     SET TIME-EOF TO TRUE
          NOT AT END PERFORM MKWORK-PROC
       END-READ
    END-PERFORM.
MKWORK-PROC.
    MOVE T-NO  TO W-NO.
    MOVE T-MIN TO W-MIN.
    [          e          ].
MKBILL-PROC.
    PERFORM UNTIL WORK-EOF
       RETURN WORK-F
          AT END     SET WORK-EOF TO TRUE
                     PERFORM WRBILL-PROC
          NOT AT END IF TMP-NO = W-NO OR ZERO THEN
                        [          f          ]
                     ELSE
                        PERFORM WRBILL-PROC
                        MOVE W-MIN TO TMP-MIN
                     END-IF
                     MOVE W-NO TO TMP-NO
       END-RETURN
    END-PERFORM.
WRBILL-PROC.
    PERFORM TEST AFTER
       UNTIL (M-NO = TMP-NO AND NOT WORK-EOF) OR MEM-EOF
       READ MEM-F
          AT END
             SET MEM-EOF TO TRUE
          NOT AT END
             MOVE M-NO TO B-NO
             MOVE BASE-PRICE(M-CODE) TO B-CHG
             IF [          g          ] THEN
                COMPUTE EXT-H = (TMP-MIN - 600 + 59) / 60
                [          d          ]
             END-IF
             WRITE B-REC
       END-READ
    END-PERFORM.
```

e, f に関する解答群

ア　ADD W-MIN TO TMP-MIN　　イ　MOVE ZERO TO TMP-MIN

ウ　PERFORM WRBILL-PROC　　エ　RELEASE W-REC

オ　WRITE W-REC

g に関する解答群

ア　(M-NO = TMP-NO OR TMP-MIN > 600) AND NOT MEM-EOF

イ　M-NO = TMP-NO

ウ　M-NO = TMP-NO AND TMP-MIN > 600

エ　M-NO = TMP-NO OR TMP-MIN > 600

オ　TMP-MIN > 600

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。
（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

スレッドを利用してタイマ処理を行うプログラムである。タイマ処理の基本機能は，秒単位で指定した時間（以下，遅延時間という）が経過したときに，インタフェースで指定したメソッドを呼び出すことである。次のクラス及びインタフェースから成る。

(1) クラス Timer は，タイマ処理の機能を定義し実装する。

① クラスメソッド createTimer：クラス Timer のインスタンス（以下，タイマという）を生成し，直ちにタイマ処理を開始する。引数で，タイマの名前，インタフェース TimerAction を実装するインスタンス及び遅延時間を指定する。

② メソッド getName：タイマの名前を返す。

③ メソッド cancel：タイマ処理をキャンセルする。

④ メソッド close：タイマ処理が終了するまで待機する。

⑤ 内部クラス Worker：タイマ処理を実装し，そのインスタンスは，スレッドとして実行される。実行スレッドは，スレッド名 Timer-*n*（*n* は 0 から始まるスレッド生成のシーケンス番号）が与えられ，次の処理を行う。

(a) 現在時刻に，タイマの生成時に指定された遅延時間を加えて，タイマの終了時刻を設定する。

(b) 実行開始時に，タイマの生成時に指定された TimerAction のインスタンスに開始の事象を通知する。

(c) 開始通知後，スレッドは終了時刻になるまで休止する。

(d) 終了時刻になったとき，終了の事象を TimerAction のインスタンスに通知し，タイマ処理の実行を終了する。

(e) タイマ処理がキャンセルされた場合は，終了時刻になるのを待たずにキャンセルの事象を TimerAction のインスタンスに通知し，タイマ処理の実行を終了する。

ここで，フィールド canceled は，複数スレッドから非同期にアクセスされるので volatile 修飾子を付けて宣言しておく。

(2) インタフェース TimerAction は，タイマ処理で事象が発生したときにタイマから事象の通知を受けるメソッドを定義する。タイマを利用するプログラムは，このインタフェースの全メソッドを実装しなければならない。各メソッドの引数は，事象が発生した Timer のインスタンス及び事象発生の時刻である。

① メソッド onStart：タイマ処理の開始時に呼び出される。

② メソッド onAlarm：タイマ生成時に指定した遅延時間が経過したときに呼び出される。

③ メソッド onCancel：タイマ処理がキャンセルされたときに呼び出される。

クラス TimerTest は，タイマ処理のテストプログラムである。メソッド main は，TimerTest のインスタンスを生成し，メソッド test を実行する。test は，二つのタイマを生成し，終了を待つ。メソッド onAlarm が，shortTimer を引数として呼び出された場合，longTimer のタイマ処理をキャンセルする。

このテストを実行したところ，図1の結果が得られた。プログラムの実行速度は十分に速いものとし，各メソッドの処理時間は秒単位の時刻の測定に影響がないものとする。


```
long timer: onStart at Mon Oct 01 21:01:40 JST 2012
short timer: onStart at Mon Oct 01 21:01:40 JST 2012
short timer: onAlarm at Mon Oct 01 21:01:42 JST 2012
long timer: onCancel at Mon Oct 01 21:01:42 JST 2012
```

図1　テストプログラムの実行結果

〔プログラム1〕

```
public interface TimerAction {
   public void onStart(Timer timer, long instant);
   public void onAlarm(Timer timer, long instant);
   public void onCancel(Timer timer, long instant);
}
```

〔プログラム2〕

```
public class Timer {
   private static int sequence;
   private String name;
   private Worker worker;
   private Thread thread;
   private TimerAction timerAction;

   private Timer(String name, TimerAction timerAction,
                 int delay) {
      this.name = name;
      this.timerAction = timerAction;
      worker = new Worker(delay);
      thread = new Thread(worker, getThreadSequenceName());
   }

   synchronized private static String getThreadSequenceName() {
      return "Timer-" + sequence++;
   }

   private void start() { thread.start(); }

   public static Timer createTimer(String name,
                                   TimerAction action, int delay) {
      Timer timer = new Timer(name, action, delay);
      timer.start();
      return timer;
   }

   public String getName() { return name; }

   public void cancel() { worker.cancel(); }

   public void close() throws InterruptedException {
      thread.join();
   }

   private class Worker implements Runnable {
      private final long endAt;
      private volatile boolean canceled;

      private Worker(int delay) {
```

```
            endAt = currentTime() + delay * 1000;
        }

        public void run() {
            timerAction.onStart(Timer.this, currentTime());
            long delta;
            while ( [ a ] ) {
                try {
                    Thread.sleep(delta);
                } catch (InterruptedException e) {
                    if (canceled) {
                        [ b ];
                    }
                }
            }
            if (canceled) {
                timerAction.onCancel(Timer.this, currentTime());
            } else {
                timerAction.onAlarm(Timer.this, currentTime());
            }
        }

        private void cancel() {
            canceled = true;
            Timer.this.thread.interrupt();
        }

        private long currentTime() {
            return System.currentTimeMillis();
        }
    }
}
```

〔プログラム3〕

```
import java.util.Date;

public class TimerTest [ c ] TimerAction {
   Timer longTimer, shortTimer;

   private void test() [ d ] InterruptedException {
      longTimer = Timer.createTimer("long timer", [ e ], 4);
      shortTimer = Timer.createTimer("short timer", [ e ], 2);
      shortTimer.close();
      longTimer.close();
   }

   private void log(String msg, Timer timer, long instant) {
      System.out.println(timer.getName() + ": " + msg
                         + " at " + new Date(instant));
   }

   public void onStart(Timer timer, long instant) {
      log("onStart", timer, instant);
   }

   public void onAlarm(Timer timer, long instant) {
      log("onAlarm", timer, instant);
      if (timer == shortTimer) {
         [ f ];
      }
   }

   public void onCancel(Timer timer, long instant) {
      log("onCancel", timer, instant);
   }

   public static void main(String[] args) [ d ]
                                    InterruptedException {
      new TimerTest().test();
   }
}
```

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　(delta = endAt + currentTime()) < 0

イ　(delta = endAt + currentTime()) > 0

ウ　(delta = endAt - currentTime()) < 0

エ　(delta = endAt - currentTime()) > 0

bに関する解答群

ア　break	イ　canceled = false	ウ　continue
エ　delta = 0	オ　endAt = 0	カ　return

c，dに関する解答群

ア　expands	イ　extends	ウ　implements
エ　subclasses	オ　throw	カ　throws

eに関する解答群

ア　longTimer	イ　new Runnable()	ウ　new Timer()
エ　new TimerAction()	オ　shortTimer	カ　this

fに関する解答群

ア　longTimer.cancel()	イ　onCancel(longTimer, instant)
ウ　onCancel(shortTimer, instant)	エ　onCancel(timer, instant)
オ　shortTimer.cancel()	カ　timer.cancel()

Java

設問2　クラス TimerTest に，メソッド log をオーバロードする次のメソッドを追加した。このメソッドは，タイマの名前の代わりにスレッド名を表示する。

```
private void log(String msg, long instant) {
   System.out.println(Thread.currentThread().getName()
           + ": " + msg + " at " + new Date(instant));
}
```

メソッド onAlarm 及び onCancel で，このオーバロードしたメソッド log を呼び出すように変更したところ，図2の実行結果が得られた。図2の3行目及び4行目の［　　］に入れる文字列の組合せとして正しい答えを，解答群の中から選べ。ここで，メソッド main を実行するスレッドの名前は main とする。また，［a］～［f］には正しい答えが入っているものとする。

```
long timer: onStart at Mon Oct 01 23:45:15 JST 2012
short timer: onStart at Mon Oct 01 23:45:15 JST 2012
[g1]: onAlarm at Mon Oct 01 23:45:17 JST 2012
[g2]: onCancel at Mon Oct 01 23:45:17 JST 2012
```

図2　TimerTest 変更後の実行結果

解答群

	g1	g2
ア	main	main
イ	main	Timer-0
ウ	main	Timer-1
エ	Timer-0	main
オ	Timer-0	Timer-1
カ	Timer-1	main
キ	Timer-1	Timer-0

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 1～4 に答えよ。

〔プログラム 1，2 の説明〕

与えられた値 x に対し，係数が全て 1 である n 次多項式 F(x, n)の値を求める副プログラム POLY である。

$$F(x, n) = x^n + x^{n-1} + \cdots + x + 1 \quad (n \geqq 1)$$

(1) POLY は，多項式の次数 n が GR1 に，x の値が GR2 に設定されて呼ばれ，計算結果を GR0 に設定して呼出し元に戻る。n は 1 ～ 32767 の整数，x は非負の整数とし，F(x, n)の値が 16 ビット符号なし 2 進整数の範囲に収まるように与えられる。POLY は，その中で副プログラム MULT を利用する。

(2) MULT は，16 ビット符号なし 2 進整数同士の乗算を行う副プログラムであり，被乗数と乗数がそれぞれ GR0，GR2 に設定されて呼ばれ，乗算結果を GR0 に設定して呼出し元に戻る。乗算結果などの桁あふれは発生しないものとする。

(3) 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 1〕

(行番号)

```
 1  POLY    START
 2          RPUSH
 3          LD      GR4,=0          ; 計算結果格納レジスタの初期化
 4  LP1     LD      GR3,GR1
 5                  a
 6          LD      GR0,=1          ; x^n の計算開始
 7  LP2     LD      GR3,GR3
 8          JZE     BRK             ; x^n の計算終了なら BRK へ
 9          CALL    MULT            ; (GR0) ← (GR0)×(GR2)
10          LAD     GR3,-1,GR3
11          JUMP    LP2
12  BRK     ADDL    GR4,GR0         ; 計算結果格納レジスタに x^n の値を加算
13          LAD     GR1,-1,GR1      ; n から 1 を減じ，低次の項の計算へ
14          JUMP    LP1
15  FIN     LD      GR0,GR4
16          RPOP
17          RET
18          END
```

〔プログラム 2〕

```
MULT    START               ; シフトによる乗算
        RPUSH
        LD      GR1,GR0
        LD      GR0,=0      ; 乗算結果の初期化
        LD      GR2,GR2
LP      [    b    ]
        LD      GR3,GR2
        AND     GR3,=#0001  ; 最下位ビットのチェック
        JZE     CONT
        ADDL    GR0,GR1
CONT    [    c    ]
        SRL     GR2,1
        JUMP    LP
FIN     RPOP
        RET
        END
```

設問 1　プログラム 1，2 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	JMI	BRK		イ	JMI	FIN		ウ	JNZ	BRK
エ	JNZ	FIN		オ	JZE	BRK		カ	JZE	FIN

b に関する解答群

ア	JMI	CONT		イ	JMI	FIN		ウ	JNZ	CONT
エ	JOV	CONT		オ	JPL	CONT		カ	JZE	FIN

c に関する解答群

ア	ADDL	GR0,GR2		イ	LD	GR2,GR1		ウ	OR	GR2,=#0001
エ	SLL	GR1,1		オ	SRA	GR1,1		カ	SRL	GR1,1

設問 2　プログラム 1 の POLY を使用して F(x, 4) を計算するとき，行番号 9 の CALL 命令が実行される回数として正しい答えを，解答群の中から選べ。

解答群

ア	4	イ	5	ウ	10
エ	11	オ	15	カ	16

設問3　関数F(x, n)は，次のように変形できる。

$$F(x,\ n) = x \times (x^{n-1} + x^{n-2} + \cdots + x + 1) + 1 \quad (n \geqq 1)$$

更に次のように，再帰的に表現することができる。

$$F(x,\ 0) = 1$$
$$F(x,\ n) = x \times F(x,\ n-1) + 1 \quad (n \geqq 1)$$

この再帰表現を実装する，プログラム3のPOLY2を作成した。プログラム3中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，POLY2はPOLYと同様に呼ばれる。

〔プログラム3〕

（行番号）

```
 1  POLY2   START
 2          RPUSH
 3          CALL    RSUB            ; 再帰処理の本体を呼ぶ
 4          RPOP
 5          RET                     ; 主プログラムへ戻る
 6  RSUB    LD      GR1,GR1         ; n=0 ?
 7          JNZ     CONT            ; n≠0 なら CONT へ
 8          LD      GR0,=1          ; GR0 ← F(x,0)の値(=1)
 9          RET
10  CONT    [                   ]
11          CALL    RSUB            ; F(x,n-1)を計算
12          CALL    MULT
13          ADDL    GR0,=1
14          RET
15          END
```

解答群

ア	LAD	GR1,-1,GR1	イ	LAD	GR1,1,GR1
ウ	LD	GR1,GR2	エ	LD	GR2,GR1
オ	POP	GR1	カ	PUSH	0,GR1

設問4　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。

プログラム3の POLY2 を使用して F(x, 4)を計算するとき，行番号 6 のラベル RSUB の行には［　d　］回制御が移る。また，行番号 12 の CALL 命令は［　e　］回実行される。

d，e に関する解答群

ア　1	イ　2	ウ　3
エ　4	オ　5	カ　6

問 13　次の表計算，ワークシート及びマクロの説明を読んで，設問 1～3 に答えよ。

〔表計算の説明〕

S 市では，急激な人口増加に伴い小学校の新設を検討している。そこで表計算ソフトを用いて候補地選定に関するシミュレーションを行うことにした。

(1) 図 1 のように，縦横が同じ長さの升目（以下，メッシュという）で S 市を分割する。また，候補地選定では，処理を簡略にするために，それぞれのメッシュ内に住んでいる全ての小学生（以下，児童という）は，メッシュの中心点から同じ小学校に通学するとみなす。

(2) S 市をメッシュで分割したときの左上を原点，水平方向右向きを x 軸の正の向き，垂直方向下向きを y 軸の正の向き，メッシュの 1 辺の長さを 1 とする 2 次元座標系を定義する。このとき，S 市を分割したメッシュの個数は 70 であった。

(3) 各メッシュを区別するために ID を設定する。左上の座標が (x, y) であるメッシュの ID を $x + 100y$ とする。ここで，全てのメッシュの左上の x 座標は 0 以上 100 未満の整数，y 座標は 0 以上の整数とする。

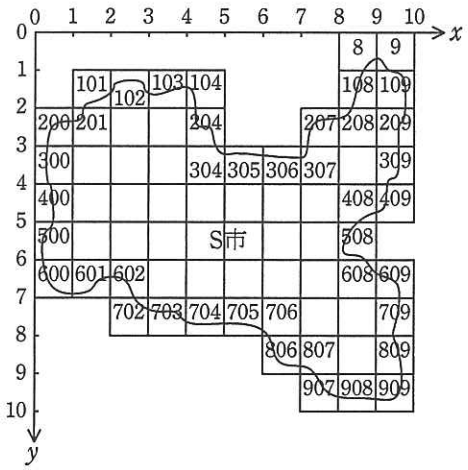


図 1　S 市をメッシュで分割した結果

〔ワークシート：児童数分布表〕

S 市の各メッシュの中心点の座標，メッシュ内に住んでいる児童数，小学校が設置可能なメッシュであるかどうかを記録する，図 2 のワークシート“児童数分布表”を作成した。

	A	B	C	D	E
1	ID	x	y	児童数（人）	設置可能
2	8	8.5	0.5	17	*
3	9	9.5	0.5	9	*
4	101	1.5	1.5	41	*
5	102	2.5	1.5	67	*
6	103	3.5	1.5	63	*
7	104	4.5	1.5	15	
⋮	⋮	⋮	⋮	⋮	⋮
71	909	9.5	9.5	41	

図２　ワークシート“児童数分布表”

(1)　セル A2～A71 には，メッシュの ID を入力する。

(2)　セル B2～B71 には，該当するメッシュの中心点の x 座標を求める式を，セル C2～C71 には，該当するメッシュの中心点の y 座標を求める式を入力する。

(3)　セル D2～D71 には，該当するメッシュ内に住んでいる児童数を入力する。

(4)　セル E2～E71 には，小学校が設置可能なメッシュのとき“＊”を入力する。

設問１　ワークシート“児童数分布表”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。

各メッシュの中心点の x 座標を求めるために，次の式をセル B2 に入力し，セル B3～B71 に複写する。

［ a ］

各メッシュの中心点の y 座標を求めるために，次の式をセル C2 に入力し，セル C3～C71 に複写する。

［ b ］

a，b に関する解答群

ア　A2／100＋0.5

イ　剰余(10, A2)／10＋0.5

ウ　剰余(100, A2)＋0.5

エ　剰余(A2, 10)／10＋0.5

オ　剰余(A2, 100)＋0.5

カ　整数部((A2＋0.5)／100)

キ　整数部(A2／100)＋0.5

ク　整数部(A2／100＋0.5)

〔ワークシート：小学校配置〕

S 市には現在，A 小学校，B 小学校，C 小学校の 3 小学校が設置されているが，既に定員を超過している状態である。そこで，定員超過を是正するために小学校を新設することになった。図 3 に示すように，新設小学校の座標を指定することで，各小学校に通学することになる児童数（以下，対象児童数という）が予測できるワークシート“小学校配置”を作成した。

	A	B	C	D	E	F	G
1	小学校名	A 小学校	B 小学校	C 小学校	新設小学校	全体	
2	x	6.327	3.292	3.002	5.527		
3	y	7.765	4.088	1.621	5.412		
4	メッシュの ID	706	403	103	505		
5	対象児童数	753	986	506	1096		
6	定員	1200	950	600	1000		
7	充足率	0.628	1.038	0.843	1.096		
8							
9	制約条件		×		×	×	
10							
11	ID	A 小学校までの距離	B 小学校までの距離	C 小学校までの距離	新設小学校までの距離	現通学小学校	新通学小学校
12	8	7.583	6.324	5.611	5.742	B 小学校	新設小学校
13	9	7.928	7.170	6.594	6.318	B 小学校	新設小学校
14	101	7.909	3.148	1.507	5.614	C 小学校	C 小学校
15	102	7.341	2.706	0.516	4.946	C 小学校	C 小学校
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
81	909	3.616	8.236	10.213	5.701	A 小学校	A 小学校

図 3　ワークシート“小学校配置”

(1)　セル B1 ～ E1 には，各小学校の名称を入力する。セル B2 ～ D3 には，既設の各小学校の x 座標及び y 座標を入力する。セル E2，E3 には，新設小学校の x 座標及び y 座標を入力する。

(2)　セル A12 ～ A81 には，ワークシート“児童数分布表”のセル A2 ～ A71 を複写する。

(3)　セル B12 ～ E81 には，メッシュの中心点から各小学校までの距離を求める式を入力する。座標 (x_1, y_1) と (x_2, y_2) の距離は次式で求められる。

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

(4) セル F12～F81 には，対応するメッシュ内に住んでいる児童が現在通学している小学校（現通学小学校という）の名称を入力する。3 小学校のうち最も距離が短い小学校に通学しているとは限らない。

(5) セル G12～G81 には，小学校を新設した場合，対応するメッシュ内に住んでいる児童が通学すべき小学校の名称を求める式を入力する。新設小学校までの距離が，現通学小学校までの距離よりも短いメッシュに住んでいる児童は，新設小学校へ転校することになる。

(6) セル B4～E4 には，各小学校が配置されているメッシュの ID を求める式を入力する。

(7) セル B5～E5 には，小学校をセル E2，E3 で示す場所に新設する場合の各小学校の対象児童数を求める式を入力する。

(8) セル B6～E6 には，各小学校の定員を入力する。

(9) セル B7～E7 には，各小学校の対象児童数を定員で割った値（充足率という）を求める式を入力する。

(10) セル B9～D9 には，制約条件として，各小学校の対象児童数が定員を超えていなければ空値に，定員を超えていれば“×”になる式を入力する。

(11) セル E9 には，制約条件として，次の条件を全て満たしていれば空値に，一つでも条件を満たさない場合は“×”になる式を入力する。

条件 1：小学校の座標が設置可能なメッシュに含まれている。

条件 2：小学校の対象児童数が定員を超えていない。

条件 3：小学校を配置するメッシュ内に既設小学校が存在しない。

(12) セル F9 には，既設 3 小学校及び新設小学校の全てが制約条件を満たしていれば空値に，そうでなければ“×”になる式を入力する。

設問2　ワークシート“小学校配置”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。

各メッシュの中心点から各小学校までの距離を算出するために，次の式をセルB12に入力し，セルB12～E81に複写する。

［　c　］

小学校の新設後，新たに通学することになる小学校（新通学小学校という）を求めるために，次の式をセルG12に入力し，セルG13～G81に複写する。

［　d　］

各小学校が配置されているメッシュのIDを算出するために，次の式をセルB4に入力し，セルC4～E4に複写する。

整数部(B2)＋整数部(B3)＊100

対象児童数を算出するために，次の式をセルB5に入力し，セルC5～E5に複写する。

［　e　］

充足率を算出するために，次の式をセルB7に入力し，セルC7～E7に複写する。

B5／B6

既設3小学校が制約条件を満たしているかどうかを判定するために，次の式をセルB9に入力し，セルC9，D9に複写する。

IF(B5≦B6, null, '×')

新設小学校が制約条件を満たしているかどうかを判定するために，次の式をセルE9に入力する。

IF(［　f　］, null, '×')

全ての小学校が制約条件を満たしているかどうかを判定するために，次の式をセルF9に入力する。

IF(論理積(B9＝null, C9＝null, D9＝null, E9＝null), null, '×')

cに関する解答群

ア　平方根((児童数分布表!$B2－B$2)^2＋(児童数分布表!$C2－B$3)^2)

イ　平方根((児童数分布表!$B2－B2)^2＋(児童数分布表!$C2－B3)^2)

ウ　平方根((児童数分布表!B2－B$2)^2＋(児童数分布表!C2－B$3)^2)

エ　平方根((児童数分布表!B2－B2)^2＋(児童数分布表!C2－B3)^2)

オ　(平方根(児童数分布表!$B2－B$2)＋平方根(児童数分布表!$C2－B$3))^2

カ　(平方根(児童数分布表!$B2－B2)＋平方根(児童数分布表!$C2－B3))^2

キ　(平方根(児童数分布表!B2－B$2)＋平方根(児童数分布表!C2－B$3))^2

ク　(平方根(児童数分布表!B2－B2)＋平方根(児童数分布表!C2－B3))^2

dに関する解答群

ア　IF(最小(B12～D12)≦E12, E$1, F12)

イ　IF(最小(B12～D12)≦E12, F12, E$1)

ウ　IF(照合一致(F12, B$1～D$1, 0)≦E12, E$1, F12)

エ　IF(照合一致(F12, B$1～D$1, 0)≦E12, F12, E$1)

オ　IF(照合検索(F12, B$1～D$1, B12～D12)≦E12, E$1, F12)

カ　IF(照合検索(F12, B$1～D$1, B12～D12)≦E12, F12, E$1)

キ　IF(水平照合(F12, B1～D12, 12, 0)≦E12, E$1, F12)

ク　IF(水平照合(F12, B1～D12, 12, 0)≦E12, F12, E$1)

eに関する解答群

ア　条件付合計($F12～$F81, ＝B1, 児童数分布表!$A2～$A71)

イ　条件付合計($F12～$F81, ＝B1, 児童数分布表!$D2～$D71)

ウ　条件付合計($F12～$F81, ≠B1, 児童数分布表!$A2～$A71)

エ　条件付合計($F12～$F81, ≠B1, 児童数分布表!$D2～$D71)

オ　条件付合計($G12～$G81, ＝B1, 児童数分布表!$A2～$A71)

カ　条件付合計($G12～$G81, ＝B1, 児童数分布表!$D2～$D71)

キ　条件付合計($G12～$G81, ≠B1, 児童数分布表!$A2～$A71)

ク　条件付合計($G12～$G81, ≠B1, 児童数分布表!$D2～$D71)

fに関する解答群

ア　論理積(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)＝'＊', E5≦E6, E4≠B4, E4≠C4, E4≠D4)

イ　論理積(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)＝'＊', E5≦E6, 否定(論理積(E4≠B4, E4≠C4, E4≠D4)))

ウ　論理積(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)≠'＊', E5≦E6, E4≠B4, E4≠C4, E4≠D4)

エ　論理積(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)≠'＊', E5≦E6, 否定(論理積(E4≠B4, E4≠C4, E4≠D4)))

オ　論理和(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)＝'＊', E5≦E6, 論理積(E4≠B4, E4≠C4, E4≠D4))

カ　論理和(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)＝'＊', E5≦E6, E4≠B4, E4≠C4, E4≠D4)

キ　論理和(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)≠'＊', E5≦E6, 論理積(E4≠B4, E4≠C4, E4≠D4))

ク　論理和(垂直照合(E4, 児童数分布表!$A2～$E71, 5, 0)≠'＊', E5≦E6, E4≠B4, E4≠C4, E4≠D4)

設問3　小学校の新設に適している場所の座標を探索することを考える。メッシュを探索するためのマクロ OptimalLocation をワークシート“小学校配置”に格納した。マクロ OptimalLocation を実行すると，新設小学校の座標が入るセル E2 及び E3 の値を，S 市を構成する各メッシュの中心点の座標で順次変化させていく。最終的に，全ての小学校が制約条件を満たし，かつ，各小学校の充足率のうちの最小値が最も大きくなる座標を，新設小学校の座標として表示する。マクロ OptimalLocation の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，全ての小学校が制約条件を満たすメッシュは一つ以上あるものとする。

〔マクロ：OptimalLocation〕

```
○マクロ：OptimalLocation
○数値型：MinI, MaxRatio, I, NumMesh
• NumMesh ← 70
• MaxRatio ← 0
• MinI ← 0
■ I: 1, I ≦ NumMesh, 1
  ▲ 相対(児童数分布表!A1, I, 4) = '*'
    • E2 ← 相対(児童数分布表!A1, I, 1)
    • E3 ← 相対(児童数分布表!A1, I, 2)
      ▲ g
      • h
      • MinI ← I
  ▼
■
• E2 ← 相対(児童数分布表!A1, MinI, 1)
• E3 ← 相対(児童数分布表!A1, MinI, 2)
```

表計算

gに関する解答群

ア　最小(B7～E7) ＜ MaxRatio

イ　最小(B7～E7) ＞ MaxRatio

ウ　論理積(最小(B7～E7) ＜ MaxRatio, F9 ＝ null)

エ　論理積(最小(B7～E7) ＜ MaxRatio, F9 ≠ null)

オ　論理積(最小(B7～E7) ＞ MaxRatio, F9 ＝ null)

カ　論理積(最小(B7～E7) ＞ MaxRatio, F9 ≠ null)

キ　論理和(最小(B7～E7) ＜ MaxRatio, F9 ＝ null)

ク　論理和(最小(B7～E7) ＜ MaxRatio, F9 ≠ null)

ケ　論理和(最小(B7～E7) ＞ MaxRatio, F9 ＝ null)

コ　論理和(最小(B7～E7) ＞ MaxRatio, F9 ≠ null)

hに関する解答群

ア　MaxRatio ← E7

イ　MaxRatio ← 最小(B7～D7)

ウ　MaxRatio ← 最小(B7～E7)

エ　MaxRatio ← 最大(B7～D7)

オ　MaxRatio ← 最大(B7～E7)

カ　MaxRatio ← 最大(最小(B7～D7), E7)

■ Java プログラムで使用する API の説明

java.lang

public class Thread

クラス Thread は，Java 仮想計算機のスレッドを表す。

コンストラクタ

public Thread(Runnable target, String name)

スレッドオブジェクトを生成する。引数で実行する Runnable のオブジェクト及びスレッドに与える名前を指定する。

引数：target — スレッド実行開始時に呼び出すメソッド run をもつ Runnable オブジェクト
name — スレッドに与える名前

メソッド

public static Thread currentThread()

現在実行中のスレッドのインスタンスを得る。

戻り値：現在実行中のスレッドのインスタンス

public static void sleep(long millis)

現在実行中のスレッドを引数で指定された時間だけ実行を一時的に中断する。

引数：millis — 実行を中断する時間（ミリ秒）

public final String getName()

このスレッドの名前を得る。

戻り値：スレッドの名前

public void interrupt()

このスレッドに割り込む。このスレッドがメソッド sleep や join などの呼出しによって実行がブロックされている場合は，このスレッドは，例外 InterruptedException を受け取る。

public final void join()

このスレッドの実行終了を待つ。実行終了を待っている間に現在のスレッドが割り込まれた場合は，例外 InterruptedException が発生する。

java.util

public class Date

クラス Date は，特定の時刻をミリ秒単位で表す。時刻は，1970 年 1 月 1 日午前 0 時（世界標準時）からの相対時間で表す。

コンストラクタ

public Date(long date)

引数で与えられた時刻を表すインスタンスを生成する。

引数 : date — 時刻（1970 年 1 月 1 日午前 0 時（世界標準時）からのミリ秒単位の相対時間）

メソッド

public String toString()

この Date オブジェクトが表している時刻を次の形式の文字列に変換する。

dow mon dd hh:mm:ss zzz yyyy

ここで,

(1) dow — 曜日の省略形（Sun, Mon, Tue, Wed, Thu, Fri, Sat）

(2) mon — 月名の省略形（Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec）

(3) dd — 日

(4) hh:mm:ss — 時間（時:分:秒）

(5) zzz — タイムゾーンの省略形（日本標準時は JST）

(6) yyyy — 年（西暦）

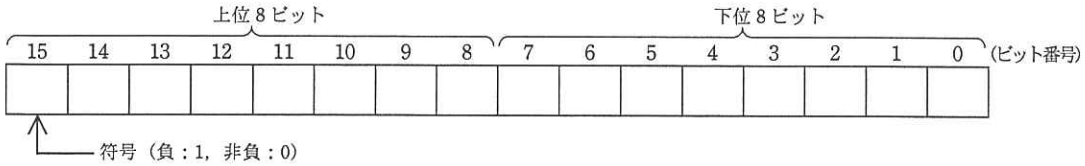
戻り値 : この Date インスタンスの文字列表現

■アセンブラ言語の仕様

1. システム COMET Ⅱ 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)	0	0
(r) ＞ (実効アドレス)		
(r1) ＝ (r2)	0	1
(r) ＝ (実効アドレス)		
(r1) ＜ (r2)	1	0
(r) ＜ (実効アドレス)		

(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（基本情報技術者試験用）

表計算ソフトの機能，用語などは，原則として次による。

なお，ワークシートの保存，読出し，印刷，罫線作成やグラフ作成など，ここで示す以外の機能などを使用するときには，問題文中に示す。

1. ワークシート

(1) 列と行とで構成される升目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，10,000 行とする。

(2) ワークシートの列と行のそれぞれの位置は，列番号と行番号で表す。列番号は，最左端列の列番号を A とし，A，B，…，Z，AA，AB，…，AZ，BA，BB，…，BZ，…，IU，IV と表す。行番号は，最上端行の行番号を 1 とし，1，2，…，10000 と表す。

(3) 複数のワークシートを利用することができる。このとき，各ワークシートには一意のワークシート名を付けて，他のワークシートと区別する。

2. セルとセル範囲

(1) ワークシートを構成する各升をセルという。その位置は列番号と行番号で表し，それをセル番地という。

［例］列 A 行 1 にあるセルのセル番地は，A1 と表す。

(2) ワークシート内のある長方形の領域に含まれる全てのセルの集まりを扱う場合，長方形の左上端と右下端のセル番地及び“～”を用いて，“左上端のセル番地～右下端のセル番地”と表す。これを，セル範囲という。

［例］左上端のセル番地が A1 で，右下端のセル番地が B3 のセル範囲は，A1～B3 と表す。

(3) 他のワークシートのセル番地又はセル範囲を指定する場合には，ワークシート名と“!”を用い，それぞれ“ワークシート名！セル番地”又は“ワークシート名！セル 範囲”と表す。

［例］ワークシート“シート1”のセル範囲 B5 ～ G10 を，別のワークシートから指定する場合には，シート1!B5～G10 と表す。

3. 値と式

(1) セルは値をもち，その値はセル番地によって参照できる。値には，数値，文字列，論理値及び空値がある。

(2) 文字列は一重引用符“ ’ ”で囲って表す。

［例］文字列“ A ”，“ BC ”は，それぞれ ’A’，’BC’ と表す。

(3) 論理値の真を true，偽を false と表す。

(4) 空値を null と表し，空値をもつセルを空白セルという。セルの初期状態は，空白セルとする。

(5) セルには，式を入力することができる。セルは，式を評価した結果の値をもつ。

(6) 式は，定数，セル番地，演算子，括弧及び関数から構成される。定数は，数値，文字列，論理値又は空値を表す表記とする。式中のセル番地は，その番地のセルの値を参照する。

(7) 式には，算術式，文字式及び論理式がある。評価の結果が数値となる式を算術式，文字列となる式を文字式，論理値となる式を論理式という。

(8) セルに式を入力すると，式は直ちに評価される。式が参照するセルの値が変化したときには，直ちに，適切に再評価される。

4. **演算子**

(1) 単項演算子は，正符号“＋”及び負符号 “－”とする。

(2) 算術演算子は，加算“＋”，減算“－”，乗算“＊”，除算“／”及びべき乗“＾”とする。

(3) 比較演算子は，より大きい“＞”，より小さい“＜”，以上“≧”，以下“≦”，等しい“＝”及び等しくない“≠”とする。

(4) 括弧は丸括弧“(”及び“)”を使う。

(5) 式中に複数の演算及び括弧があるときの計算の順序は，次表の優先順位に従う。

演算の種類	演算子	優先順位
括弧	( )	高
べき乗演算	＾	↑
単項演算	＋，－	↑
乗除演算	＊，／	↑
加減演算	＋，－	↑
比較演算	＞，＜，≧，≦，＝，≠	低

5. **セルの複写**

(1) セルの値又は式を，他のセルに複写することができる。

(2) セルを複写する場合で，複写元のセル中にセル番地を含む式が入力されているとき，複写元と複写先のセル番地の差を維持するように，式中のセル番地を変化させるセルの参照方法を相対参照という。この場合，複写先のセルとの列番号の差及び行番号の差を，複写元のセルに入力された式中の各セル番地に加算した式が，複写先のセルに入る。

［例］セル A6 に式 A1 ＋ 5 が入力されているとき，このセルをセル B8 に複写すると，セル B8 には式 B3 ＋ 5 が入る。

(3) セルを複写する場合で，複写元のセル中にセル番地を含む式が入力されているとき，そのセル番地の列番号と行番号の両方又は片方を変化させないセルの参照方法を絶対参照という。絶対参照を適用する列番号と行番号の両方又は片方の直前には“$”を付ける。

［例］セル B1 に式 A1 ＋ $A2 ＋ A$5 が入力されているとき，このセルをセル C4 に複写す

ると，セル C4 には式 A1 + $A5 + B$5 が入る。

(4) セルを複写する場合で，複写元のセル中に，他のワークシートを参照する式が入力されているとき，その参照するワークシートのワークシート名は複写先でも変わらない。

［例］ワークシート“シート2”のセル A6 に式 シート1!A1 が入力されているとき，このセルをワークシート“シート3”のセル B8 に複写すると，セル B8 には式 シート1!B3 が入る。

6. 関数

式には次の表で定義する関数を利用することができる。

書式	解　　説
合計 (セル範囲[1)])	セル範囲に含まれる数値の合計を返す。 ［例］合計 (A1 ～ B5) は，　セル範囲 A1～B5 に含まれる数値の合計を返す。
平均 (セル範囲[1)])	セル範囲に含まれる数値の平均を返す。
標本標準偏差 (セル 範囲[1)])	セル範囲に含まれる数値を標本として計算した標準偏差を返す。
母標準偏差 (セル範囲[1)])	セル範囲に含まれる数値を母集団として計算した標準偏差を返す。
最大 (セル範囲[1)])	セル範囲に含まれる数値の最大値を返す。
最小 (セル範囲[1)])	セル範囲に含まれる数値の最小値を返す。
IF (論理式 , 式1, 式2)	論理式の値が true のとき式 1 の値を，false のとき式 2 の値を返す。 ［例］IF (B3 ＞ A4,' 北海道 ',C4) は，セル B3 の値がセル A4 の値より大きいとき文字列“北海道”を，それ以外のときセル C4 の値を返す。
個数 (セル範囲)	セル範囲に含まれるセルのうち，空白セルでないセルの個数を返す。
条件付個数 (セル範囲 , 検索条件の記述)	セル範囲に含まれるセルのうち，検索条件の記述で指定された条件を満たすセルの個数を返す。検索条件の記述は比較演算子と式の組で記述し，セル範囲に含まれる各セルと式の値を，指定した比較演算子によって評価する。 ［例1］条件付個数 (H5 ～ L9, ＞ A1) は，セル範囲 H5 ～ L9 のセルのうち，セル A1 の値より大きな値をもつセルの個数を返す。 ［例2］条件付個数 (H5 ～ L9, ＝'A4') は，セル範囲 H5 ～ L9 のセルのうち，文字列 “A4 ” をもつセルの個数を返す。
整数部 (算術式)	算術式の値以下で最大の整数を返す。 ［例1］整数部 (3.9) は，3 を返す。 ［例2］整数部 (−3.9) は，−4 を返す。
剰余 (算術式1, 算術式2)	算術式1 の値を被除数，算術式2 の値を除数として除算を行ったときの剰余を返す。関数“剰余”と“整数部”は，剰余 (x,y) ＝ x − y ＊ 整数部 (x ／ y) という関係を満たす。 ［例1］剰余 (10,3) は，1 を返す。 ［例2］剰余 (− 10,3) は，2 を返す。
平方根 (算術式)	算術式の値の非負の平方根を返す。算術式の値は，非負の数値でなければならない。
論理積 (論理式1, 論理式2, …) [2)]	論理式1, 論理式2, … の値が全て true のとき，true を返す。それ以外のとき false を返す。
論理和 (論理式1, 論理式2, …) [2)]	論理式1, 論理式2, … の値のうち，少なくとも一つが true のとき，true を返す。それ以外のとき false を返す。
否定 (論理式)	論理式の値が true のとき false を，false のとき true を返す。

切上げ（算術式，桁位置） 四捨五入（算術式，桁位置） 切捨て(算術式，桁位置)	算術式の値を指定した桁位置で，関数“切上げ”は切り上げた値を，関数“四捨五入”は四捨五入した値を，関数“切捨て”は切り捨てた値を返す。ここで，桁位置は小数第1位の桁を0とし，右方向を正として数えたときの位置とする。 ［例1］切上げ(－314.159,2) は，－314.16 を返す。 ［例2］切上げ(314.159,－2) は，400 を返す。 ［例3］切上げ(314.159,0) は，315 を返す。
結合(式1,式2,…)[2]	式1，式2，… のそれぞれの値を文字列として扱い，それらを引数の順につないでできる一つの文字列を返す。 ［例］結合('北海道','九州',123,456) は，文字列“北海道九州123456”を返す。
順位（算術式，セル範囲[1]，順序の指定）	セル範囲の中での算術式の値の順位を，順序の指定が0の場合は昇順で，1の場合は降順で数えて，その順位を返す。ここで，セル範囲の中に同じ値がある場合，それらを同順とし，次の順位は同順の個数だけ加算した順位とする。
乱数（ ）	0 以上 1 未満の一様乱数 （実数値） を返す。
表引き(セル範囲，行の位置，列の位置)	セル範囲の左上端から行と列をそれぞれ1，2，… と数え，セル範囲に含まれる行の位置と列の位置で指定した場所にあるセルの値を返す。 ［例］表引き(A3 ～ H11,2,5) は，セル E4 の値を返す。
垂直照合(式,セル範囲,列の位置,検索の指定)	セル範囲の左端列を上から下に走査し，検索の指定によって指定される条件を満たすセルが現れる最初の行を探す。その行に対して，セル範囲の左端列から列を1，2，… と数え，セル範囲に含まれる列の位置で指定した列にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき，左端列は上から順に昇順に整列されている必要がある。 ［例］垂直照合(15,A2 ～ E10,5,0) は，セル範囲の左端列をセル A2，A3，…，A10 と探す。このとき，セル A6 で 15 を最初に見つけたとすると，左端列 A から数えて5列目の列 E 中で，セル A6 と同じ行にあるセル E6 の値を返す。
水平照合(式,セル範囲,行の位置,検索の指定)	セル範囲の上端行を左から右に走査し，検索の指定によって指定される条件を満たすセルが現れる最初の列を探す。その列に対して，セル範囲の上端行から行を1，2，… と数え，セル範囲に含まれる行の位置で指定した行にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき，上端行は左から順に昇順に整列されている必要がある。 ［例］水平照合(15,A2 ～ G6,5,1) は，セル範囲の上端行をセル A2，B2，…，G2 と探す。このとき，15 以下の最大値をセル D2 で最初に見つけたとすると，上端行2から数えて5行目の行6中で，セル D2 と同じ列にあるセル D6 の値を返す。
照合検索（式，検索のセル範囲，抽出のセル範囲）	1行又は1列を対象とする同じ大きさの検索のセル範囲と抽出のセル範囲に対して，検索のセル範囲を左端又は上端から走査し，式の値と一致する最初のセルを探す。見つかったセルの検索のセル範囲の中での位置と，抽出のセル範囲の中での位置が同じセルの値を返す。 ［例］照合検索(15,A1 ～ A8,C6 ～ C13) は，セル範囲 A1 ～ A8 をセル A1，A2，… と探す。このとき，セル A5 で 15 を最初に見つけたとすると，セル範囲 C6 ～ C13 の上端から数えて5番目のセル C10 の値を返す。

照合一致(式,セル範囲,検索の指定)	1行又は1列を対象とするセル範囲に対して，セル範囲の左端又は上端から走査し，検索の指定によって指定される条件を満たす最初のセルを探す。見つかったセルの位置を，セル範囲の左端又は上端から 1，2，… と数えた値とし，その値を返す。 ・検索の指定が 0 の場合の条件：式の値と一致する値を検索する。 ・検索の指定が 1 の場合の条件：式の値以下の最大値を検索する。このとき，セル範囲は左端又は上端から順に昇順に整列されている必要がある。 ・検索の指定が −1 の場合の条件：式の値以上の最小値を検索する。このとき，セル範囲は左端又は上端から順に降順に整列されている必要がある。 ［例］照合一致(15,B2 ～ B12,−1) は，セル範囲 B2 ～ B12 をセル B2，B3，…と探す。このとき，15 以上の最小値をセルB9 で最初に見つけたとすると，セル B2 から数えた値 8 を返す。
条件付合計(検索のセル範囲,検索条件の記述,合計のセル範囲[1)])	行数及び列数が共に同じ検索のセル範囲と合計のセル範囲に対して，検索と合計を行う。検索のセル範囲に含まれるセルのうち，検索条件の記述で指定される条件を満たすセルを全て探す。検索条件の記述を満たした各セルについての左上端からの位置と，合計のセル範囲中で同じ位置にある各セルの値を合計して返す。 検索条件の記述は比較演算子と式の組で記述し，検索のセル範囲に含まれる各セルと式の値を，指定した比較演算子によって評価する。 ［例1］条件付合計 (A1 ～ B8,＞ E1,C2 ～ D9) は，検索のセル範囲である A1 ～ B8 のうち，セル E1 の値より大きな値をもつ全てのセルを探す。このとき，セル A2，B4，B7 が見つかったとすると，合計のセル範囲である C2 ～ D9 の左上端からの位置が同じであるセル C3，D5，D8 の値を合計して返す。 ［例2］条件付合計 (A1 ～ B8,＝ 160,C2 ～ D9) は，検索のセル範囲である A1 ～ B8 のうち，160 と一致する値をもつ全てのセルを探す。このとき，セル A2，B4，B7 が見つかったとすると，合計のセル範囲である C2 ～ D9 の左上端からの位置が同じであるセル C3，D5，D8 の値を合計して返す。

注[1)] 引数として渡したセル範囲の中で，数値以外の値は処理の対象としない。

[2)] 引数として渡すことができる式の個数は，1 以上である。

7. マクロ

(1) ワークシートとマクロ

ワークシートには複数のマクロを格納することができる。

マクロは一意のマクロ名を付けて宣言する。マクロの実行は，表計算ソフトのマクロの実行機能を使って行う。

［例］○**マクロ**：Pro

例は，マクロ Pro の宣言である。

(2) 変数とセル変数

変数の型には，数値型，文字列型及び論理型があり，変数は宣言することで使用できる。変数名にセル番地を使用することはできない。

［例］○**数値型**：row，col

例は，数値型の変数 row，col の宣言である。

セルを変数として使用でき，これをセル変数という。セル変数は，宣言せずに使用できる。

セル変数の表現方法には，絶対表現と相対表現とがある。

セル変数の絶対表現は，セル番地で表す。

セル変数の相対表現は，次の書式で表す。

書式	解　　説
相対(セル変数，行の位置，列の位置)	セル変数で指定したセルを基準のセルとする。そのセルの行番号と列番号の位置を 0 とし，下又は右方向を正として数え，行の位置と列の位置で指定した数と一致する場所にあるセルを表す変数である。

［例1］**相対(B5, 2, 3)** は，セル E7 を表す変数である。

［例2］**相対(B5, −2, −1)** は，セル A3 を表す変数である。

(3)　配列

数値型，文字列型又は論理型の配列は宣言することで使用できる。添字を“[”及び“]”で囲み，添字が複数ある場合はコンマで区切る。添字は 0 から始まる。

なお，数値型及び文字列型の変数及び配列の要素には，空値を格納することができる。

［例］**○文字列型：table[100, 200]**

例は，100 × 200 個 の文字列型の要素をもつ 2 次元配列 table の宣言である。

(4)　宣言，注釈及び処理

宣言，注釈及び処理の記述は，“共通に使用される擬似言語の記述形式”に従う。

処理の記述中に式又は関数を使用する場合，その記述中に変数，セル変数又は配列の要素が使用できる。

［例］**○数値型：row**

```
row：0, row ＜ 5, 1
・相対(B5, row, 0)　←　順位(相対(C5, row, 0), G5～G9, 0)
```

例は，セル C5，C6，…，C9 の各値に対して，セル範囲 G5 ～ G9 の中での順位を調べ，その順位をセル B5，B6，…，B9 に順に代入する。

〔 メ モ 用 紙 〕

6. 退室可能時間に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

7. **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。
8. 問題冊子の余白などは，適宜利用して構いません。
9. Java プログラムで使用する API の説明，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。
10. 試験時間中，机上に置けるものは，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，黒鉛筆及びシャープペンシル（B又はHB），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，ポケットティッシュ，目薬
    これら以外は机上に置けません。使用もできません。
11. 試験終了後，この問題冊子は持ち帰ることができます。
12. 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。
13. 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。