FE

平成25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2	問3	問4	問5	問6	問7
●	(選)	●	●	(選)	●	●
問8	問9	問10	問11	問12	問13	
●	●	(選)	(選)	(選)	(選)	

   (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) (オ) (カ) (キ) (ク) (ケ) (コ)

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	論理演算と加算器
問 2	データベース	選手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	インターネットプロトコルのアドレス表記
問 4	情報セキュリティ	VPN（Virtual Private Network）
問 5	ソフトウェア設計	ソフトウェアのテスト設計
問 6	プロジェクトマネジメント	プロジェクトの実績管理
問 7	システム戦略	販売管理システムの見直しを伴う業務改善

●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	文字列の圧縮

●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）	辞書順での文字列の比較
問 10	ソフトウェア開発（COBOL）	テニスコートの予約
問 11	ソフトウェア開発（Java）	木構造の生成
問 12	ソフトウェア開発（アセンブラ）	数字列の時間と数値の秒との変換
問 13	ソフトウェア開発（表計算）	受講学生のグループ分け

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，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　論理演算と加算器に関する次の記述を読んで，設問1～4に答えよ。

真を1，偽を0として，主要な論理演算の真理値表を，表1に示す。

表1　主要な論理演算の真理値表

A	B	A AND B	A OR B	A XOR B	A NAND B	A NOR B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	1	0
1	1	1	1	0	0	0

A	NOT A
0	1
1	0

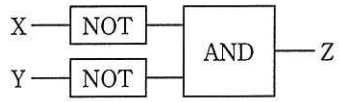
設問1　AND，OR，XOR，NOT の各論理演算を行う論理回路を用いて，NAND と NOR の論理演算を行う論理回路を作成した。次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，X，Y は 1 ビットの入力，Z は 1 ビットの出力とする。

(1)　NAND の論理回路は［　a　］である。

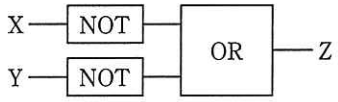
(2)　NOR の論理回路は［　b　］である。

a，bに関する解答群

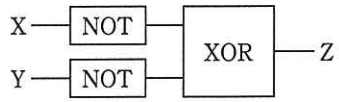
ア



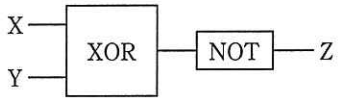
イ



ウ



エ



設問2　各1ビットの入力X，Yを加算して，その結果を各1ビットのZと桁上がりCに出力する“半加算器”の真理値表を表2に，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。

$$\begin{array}{rcc} & & X \\ + & & Y \\ \hline & C & Z \end{array}$$

表2　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

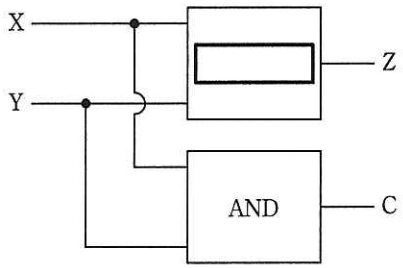


図1　半加算器の論理回路

解答群

ア　AND　　イ　NAND　　ウ　NOR　　エ　OR

オ　XOR

設問3　各1ビットの入力X，Yと，下位桁からの1ビットの桁上がり C_{in} を加算して，その結果を各1ビットのZと桁上がりのCに出力する“全加算器”の真理値表を表3に，論理回路を図2に示す。図2中の［　　］に入れる正しい答えを，解答群の中から選べ。

$$\begin{array}{rcc} & X & \\ & Y & \\ + & C_{in} & \\ \hline & C & Z \end{array}$$

表3　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	β
1	0	0	0	1	β
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	

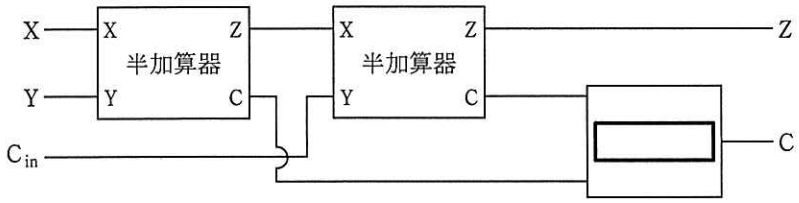


図2　全加算器の論理回路

解答群

ア　AND　　イ　NAND　　ウ　NOR　　エ　OR

設問4　2の補数表現による4ビットの符号付き2進整数を加算する加算器を図3に示す。加算器は，2進整数 $A_4A_3A_2A_1$ と $B_4B_3B_2B_1$ を加算して，結果 $S_4S_3S_2S_1$ を出力する。添字は桁の位置を示しており，値が大きいほど上位の桁を表す。

$$\begin{array}{cccccc} & A_4 & A_3 & A_2 & A_1 \\ + & B_4 & B_3 & B_2 & B_1 \\ \hline & S_4 & S_3 & S_2 & S_1 \end{array}$$

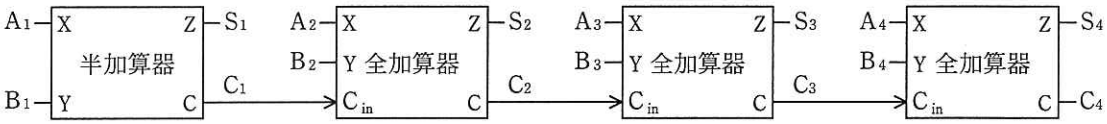
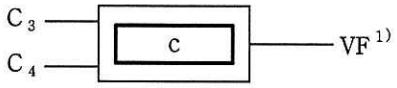


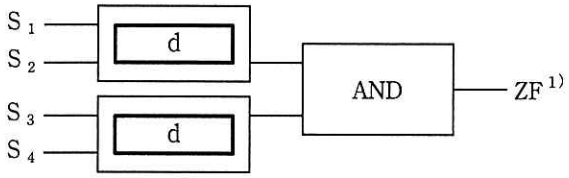
図3　加算器の論理回路

最上位ビットの加算において，A_4，B_4，C_3 の値が表3の全加算器の真理値表のそれぞれ X，Y，C_{in} の値の β 部分の組合せになるとき，桁あふれが生じる。これを検出するための論理回路を図4に，S_1 ～ S_4 が全て0となる場合を検出する論理回路を図5に示す。図4中と図5中の［　　］に入れる正しい答えを，解答群の中から選べ。



注 1)　桁あふれが生じたとき VF の値は 1，
それ以外のとき VF の値は 0

図 4　桁あふれ検出の論理回路



注 1)　S_1 ～ S_4 が全て 0 のとき ZF の値は 1，それ以外のとき ZF の値は 0

図 5　ゼロ検出の論理回路

c，d に関する解答群

ア　AND　　イ　NAND　　ウ　NOR　　エ　OR

オ　XOR

問2　選手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。

ある少年野球リーグの事務局では，登録選手の氏名や成績などの個人情報を管理するために，関係データベースを構築することにした。このリーグには，近隣の8チームが参加している。

まず，リーグに所属するチームと登録選手の情報を管理するために，図1に示すチーム表と選手表を設計した。下線付きの項目は，主キーを表す。

チーム表

チーム番号	チーム名	代表者氏名	代表者住所	代表者電話番号
01	巣鴨キッズ	情報太郎	東京都豊島区○○○2-28	03-1111-2222
⋮	⋮	⋮	⋮	⋮

選手表

選手番号	氏名	住所	電話番号	チーム番号	登録日	抹消日
0001	巣鴨一郎	東京都豊島区□□1-2	03-8888-9999	01	20080401	null
⋮	⋮	⋮	⋮	⋮	⋮	⋮

図1　チーム表と選手表のデータ格納例

設問1　チームの対戦成績を管理する表を設計する。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。

このリーグでは，毎年4月から翌年の3月までを1シーズンとし，試合は各チームが他のチームの全てと1回だけ対戦する総当たり方式で行う。

このデータベースでは，各チームの対戦成績や，勝利投手，敗戦投手などの情報を管理する。チーム成績は勝点によって順位付けする。勝点は，勝利チームに3点，敗戦チームに0点，引分けの場合は両チームに1点ずつを付与する。

最初，図1に示すチーム表に，必要な項目を追加することを考えたが，総当たりで対戦することから，表に繰返し項目が発生することになる。これを改善するために，［　a　］した。また，管理する情報の性質上，チーム表や選手表は更新しながら継続的に使用するが，対戦成績はシーズンごとに表を作成して管理

したい。さらに，勝利投手や敗戦投手といった，個々の試合に関する情報を管理するには，別の表にした方が扱いやすいと判断して，図2に示す日程表と結果表を作成することにした。ここで，1シーズンで作成される結果表のレコード件数は，［ b ］件になる。

日程表

試合番号	試合日	対戦チーム1	対戦チーム2
001	20130413	01	02
002	20130414	03	04
⋮	⋮	⋮	⋮

結果表

試合番号	チーム番号	得点	勝点	勝利投手	敗戦投手
001	01	2	0	null	0038
001	02	5	3	0021	null
002	03	2	1	null	null
002	04	2	1	null	null
⋮	⋮	⋮	⋮	⋮	⋮

図2　日程表と結果表のデータ格納例

aに関する解答群

ア　インデックスを設定　　イ　第1正規化　　ウ　第2正規化

エ　第3正規化　　オ　セキュリティを強化

bに関する解答群

ア　48　　イ　56　　ウ　64　　エ　98　　オ　112　　カ　128

設問2　勝点が多いチームから降順にチーム番号，チーム名，勝点，総得点を表示する。このとき，勝点が等しい場合は，総得点の降順に表示する。次の SQL 文の［　　］に入れる正しい答えを，解答群の中から選べ。

```
SELECT チーム表.チーム番号，チーム表.チーム名，
       SUM(結果表.勝点) AS 勝点，SUM(結果表.得点) AS 総得点
  FROM チーム表，結果表
  WHERE チーム表.チーム番号 = 結果表.チーム番号
        [　　c　　]
```

解答群

ア　GROUP BY チーム表.チーム番号，チーム表.チーム名
　　ORDER BY 勝点 ASC，総得点 ASC

イ　GROUP BY チーム表.チーム番号，チーム表.チーム名
　　ORDER BY 勝点 DESC，総得点 DESC

ウ　ORDER BY 勝点，総得点 ASC

エ　ORDER BY 勝点，総得点 DESC

設問3　選手個人の打撃成績を管理するために，図3に示す打席表と打撃表を作成した。ホームランを打った数（以下，ホームラン数という）が多い選手から降順に選手番号，選手名，ホームラン数を表示する。次の SQL 文の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，d1 と d2 に入れる答えは，d 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。

```
SELECT 打席表.打者，選手表.氏名，[　　d1　　] AS 集計数
  FROM 打席表，選手表
  WHERE 打席表.打者 = 選手表.選手番号 AND
        打席表.打撃結果 = (SELECT 打撃表.打撃結果 FROM 打撃表
                           WHERE 打撃表.名称 = 'ホームラン')
  GROUP BY 打席表.打者，選手表.氏名
  ORDER BY 集計数 [　　d2　　]
```

打席表

試合番号	打席番号[1]	回数	表裏[2]	打者	投手	打撃結果	打点
001	001	01	1	0002	0021	002	0
001	002	01	1	0004	0021	003	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
001	008	01	2	0075	0038	002	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

注[1] 打席番号には，試合番号で示す試合における通番を格納する。例えば，1回表の攻撃が5人で終わった場合，1回裏の攻撃は打席番号006から始まる。

注[2] 表裏には，表の攻撃の場合に1を，裏の攻撃の場合に2を格納する。

打撃表

打撃結果	名称
001	三振
002	ヒット
003	ホームラン
⋮	⋮

図3 打席表と打撃表のデータ格納例

dに関する解答群

	d1	d2
ア	COUNT(*)	DESC
イ	MAX(打席表.打者)	ASC
ウ	MIN(打席表.投手)	ASC
エ	SUM(打席表.打点)	DESC

設問4　このリーグでは，チーム表や選手表は更新しながら継続的に使用する。対戦成績と打撃成績はシーズンごとに表を作成するが，過去の情報も参照できるように，シーズン終了後も蓄積しておく。

リーグに所属する選手情報の管理について，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。

新しい選手の情報は，選手表に追加すればよい。リーグを離れる選手の情報は，蓄積されている情報の参照を考慮して，削除せずに残しておいた方がよい。

ある選手がシーズン途中に別のチームへ移籍する場合，選手表のチーム番号を更新すると，例えば，SQL 文を用いて当該シーズンにおける［　e　］の集計はできなくなる。移籍前の情報は，抹消日を格納した上でそのまま残して，当該選手に新しい選手番号を割り振って登録する方法もあるが，その場合は，SQL 文を用いて［　f　］の集計ができなくなる。そこで，このリーグでは，選手の移籍にも柔軟に対処できるように，選手の情報を図4に示すとおり所属選手表と選手表で管理するように変更した。ただし，移籍して元のチームに戻ることはないものとする。

所属選手表

チーム番号	選手番号	登録日	抹消日
01	0001	20080401	20100831
⋮	⋮	⋮	⋮
03	0001	20100901	null

選手表

選手番号	氏名	住所	電話番号	登録日	抹消日
0001	巣鴨一郎	東京都豊島区□□1-2	03-8888-9999	20080401	null
⋮	⋮	⋮	⋮	⋮	⋮

図4　変更後の表構成とデータ格納例

解答群

ア　選手ごとのホームラン数　　イ　チームごとの勝点

ウ　チームごとの総得点　　エ　チームごとのホームラン数

オ　リーグの総得点　　カ　リーグのホームラン数

問3 インターネットプロトコルのアドレス表記に関する記述を読んで，設問 1，2 に答えよ。

インターネットプロトコル（IP）として現時点で広く使われているのは，IP バージョン 4（以下，IPv4 という）である。IPv4 では，そのアドレスを表現するのに 32 ビットを使用している。単純に 32 ビットのアドレス空間を考えると，2^{32} 個（約 43 億個）のアドレスが使用できるが，世界の人口が約 70 億人であるので一人ひとりに 1 個ずつのアドレスを割り当てられない。今日，インターネットに接続できる機器の数が爆発的に増えていることを考えると，IPv4 のアドレスを全て使い切ってしまうのは時間の問題である。これは，“IPv4 アドレスの枯渇問題”として知られている。

そこで，IP バージョン 6（以下，IPv6 という）が開発された。IPv6 では，アドレスを 128 ビットで表す。単純に 128 ビットのアドレス空間を考えると，2^{128} 個（約 3.4×10^{38} 個）のアドレスが使用でき，世界中の人に割り当てても一人当たり約 [a] 個使える計算になる。

IPv4 では，32 ビットのアドレスを表現するのに 8 ビットごとに区切り，各 8 ビットの値を 10 進数で表し，区切りにドット（.）を使用する表記方法が用いられている。IPv6 では，128 ビットを 16 ビットごとに区切り，各 16 ビット（以下，16 ビットセクションという）を 16 進数で表し，区切りにコロン（:）を使用する。また，次の規則を使用してアドレスを表現する文字列を圧縮できる。

(1) 各 16 ビットセクションの先行する `0` を省略する。例えば，`0012` は `12` になる。ただし，16 ビットセクションが `0000` のときは，`0` とする。

(2) `0` の 16 ビットセクションが連続する場合は，連続する 2 個のコロン（`::`）で表す。例えば，`2001:0db8:0000:0000:0000:ff00:0042:8329` は `2001:db8::ff00:42:8329` と表す。ただし，`::` は 1 か所にだけ使用できる。

IPv6 も IPv4 と同様に，アドレスはネットワークを表す部分とホストを表す部分から成る。IPv6 では，ネットワークの部分が何ビットで表されるかを示すのにプレフィックス長が用いられる。プレフィックス長は，先頭からのビット数を 10 進数で表したものである。アドレスとプレフィックス長は，スラッシュ（`/`）で区切る。例えば，`2001:db8:abcd:12::/64` では，プレフィックス長は 64 であり，先頭から 64 ビットがネットワークアドレス部であることを表している。

IPv6 を導入することによって IPv4 アドレスの枯渇問題は回避できるが，インターネットにつながる世界中のシステムや機器が一斉に IPv6 に変更されるということは期待できず，長い時間を掛けて移行が行われると考えられる。そこで，IPv4 から IPv6 への移行期間中に二つのバージョンが共存する手段として様々な技術が開発されている。そのうちの一つがトンネル方式であり，送信元と宛先の間に異なる IP バージョンが存在するとき，元の IP バージョンのパケットを異なる IP バージョンのパケットにカプセル化し，異なる IP バージョンのネットワークを通過させる方式である。トンネル方式を実現する方法は幾つかあり，6 to 4 と呼ばれる技術もその一つである。

6 to 4 では，IPv4 のグローバルアドレスが必要である。6 to 4 で使用する IPv6 アドレスの最初の 16 ビットは，必ず `2002` である。続いて IPv4 のグローバルアドレスをそのまま付けた［ b ］ビットが IPv6 アドレスのプレフィックスとなる。例えば，IPv4 グローバルアドレスが `203.0.113.128` の場合，IPv6 アドレスのプレフィックスは，［ c ］/［ b ］となる。

設問1　説明文中の［　　］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　4.9×10^{27}　　イ　4.9×10^{28}　　ウ　4.9×10^{29}

エ　1.4×10^{117}　　オ　1.4×10^{118}　　カ　1.4×10^{119}

bに関する解答群

ア　32　　イ　40　　ウ　48

エ　56　　オ　64　　カ　72

キ　80　　ク　88　　ケ　96

cに関する解答群

ア　2002::cb00:7180　　イ　2002:0:0:cb00:7180::

ウ　2002:203:0:113:128::　　エ　2002:cb:0:71:80::

オ　2002:cb00:7180::　　カ　203:0:113:128::

キ　cb00:7180::

設問2　表1は，IPv6アドレスの基本形と，それを最も短く表現する圧縮形の対応を示したものである。表1中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，基本形は全16ビットセクションを4桁の16進数で表現したもの（先行する0を省略しない）である。

表1　IPv6アドレスの基本形と圧縮形の対応

基本形	圧縮形
d	2001:db8::2:1
2001:0db8:0000:0000:cd30:0000:0000:0000	e
0000:0000:0000:0000:0000:0000:0000:0001	f

dに関する解答群

ア　2001:0db8:0000:0000:0000:0000:0000:0021

イ　2001:0db8:0000:0000:0000:0000:0000:2:01

ウ　2001:0db8:0000:0000:0000:0000:0002:0001

エ　2001:0db8:0000:0000:0000:0002:0001

オ　2001:0db8:0000:0000:0002:0001

カ　2001:0db8:0000:0000:0002:0001:0000

キ　2001:0db8:0000:0000:0002:0001:0000:0000

eに関する解答群

ア　2001:0db8::cd30::　　イ　2001:0db8::cd30:0:0:0

ウ　2001:0db8:0:0:cd30　　エ　2001:0db8:0:0:cd30::

オ　2001:db8::cd30::　　カ　2001:db8::cd30:0:0:0

キ　2001:db8:0:0:cd30　　ク　2001:db8:0:0:cd30::

fに関する解答群

ア　::0:1　　イ　::0001　　ウ　::1

エ　:0::1　　オ　0::1　　カ　0:0::1

キ　0000::1

問4　VPN（Virtual Private Network）に関する記述を読んで，設問1～3に答えよ。

A社は，関東のN事業所で利用している営業支援システムを，関西のM事業所でも利用することにした。営業支援システムのサーバはN事業所のコンピュータセンタに設置されている。M事業所でN事業所の営業支援システムを利用するために，システム部が中心となってIPsecを利用したVPNの導入を検討し，報告書を作成した。

〔報告書の内容（抜粋）〕

(1)　ネットワーク構成

M事業所からN事業所の営業支援システムに接続するためのネットワーク構成を図1に示す。VPNの実現には，VPNルータを利用する。

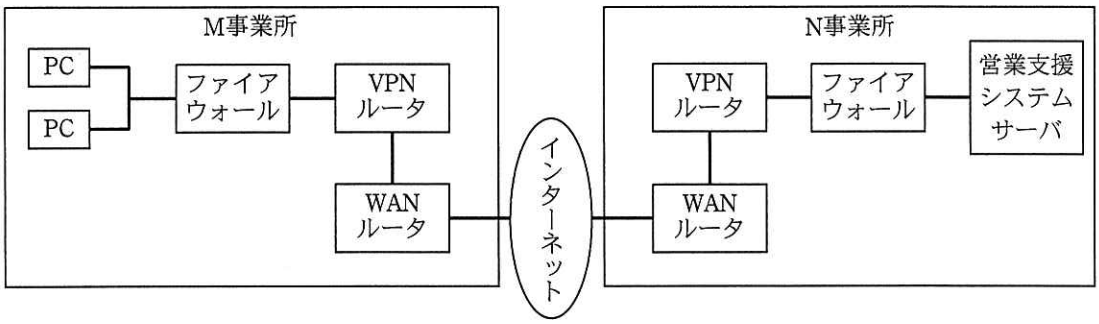


図1　ネットワーク構成

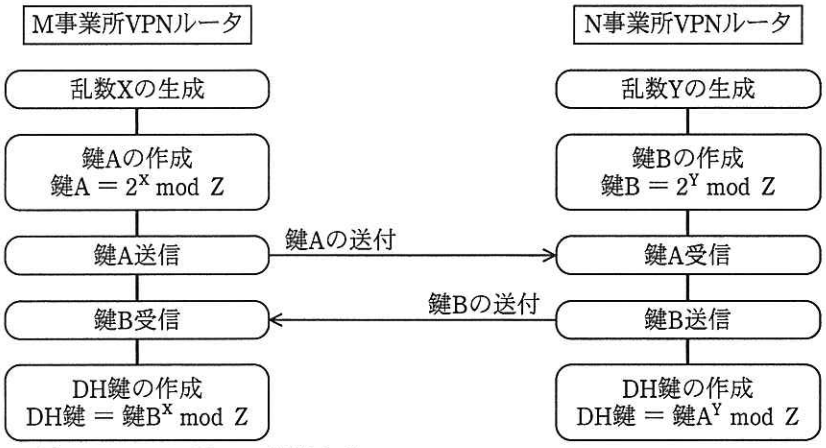
(2)　IPsecの説明

IPsecは，暗号技術を用いてインターネットでデータを安全に送受信するための規格である。IPsecには，暗号化に利用する鍵を安全に交換する仕組みや，相手のVPNルータを認証する仕組みがある。

(3) IPsecの要素技術の説明

① 暗号化に利用する鍵を安全に交換する仕組み

IPsecでは，VPNルータ間で暗号化に利用する鍵を，安全に交換する仕組みの一つとして，Diffie-Hellman鍵交換法（以下，DH法という）を利用している。DH法の例を図2に示す。DH法で作成された鍵（以下，DH鍵という）を暗号化に利用する。



注記1　X，Yは正の整数とする。

注記2　2^Xは，2のX乗を示す。

注記3　P mod Qは，PのQによる剰余を示す。

注記4　Zは，M事業所VPNルータ，N事業所VPNルータに事前に設定された素数である。

図2　DH法の例

② 相手の VPN ルータを認証する仕組み

IPsec では，データ受信側の VPN ルータがデータ送信側の VPN ルータを認証する仕組みの一つとして，RSA アルゴリズムを用いたディジタル署名を利用している。その仕組みを図 3 に示す。

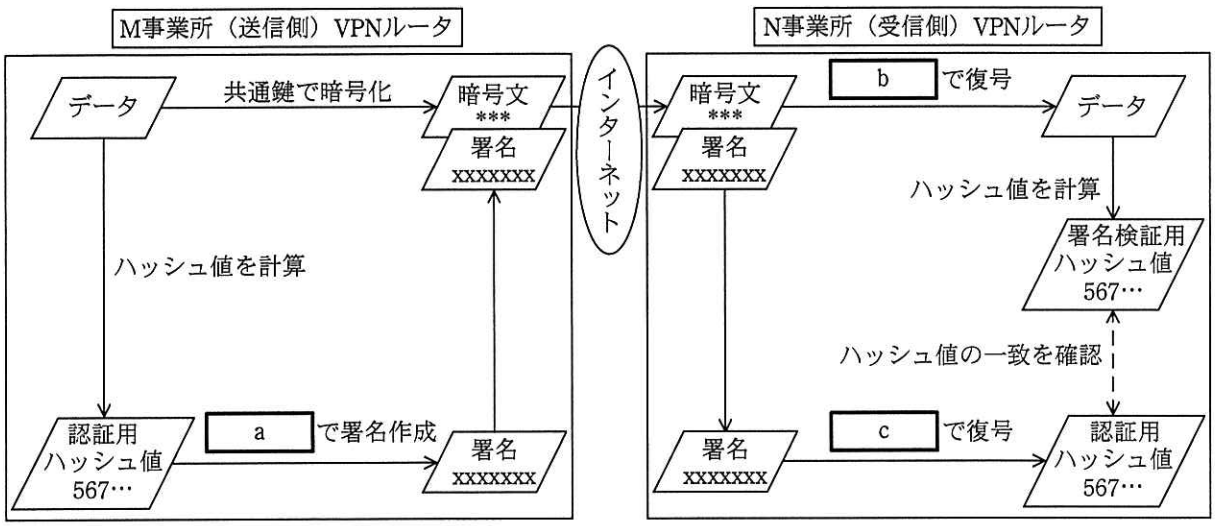


図 3　相手の VPN ルータを認証する仕組み

(4) IPsec を利用した VPN の導入効果

IPsec は，［ d ］層でセキュリティを実現するプロトコルであるので，アプリケーションを変更せずに通信のセキュリティを担保できる。そして，パケットを暗号化することによって［ e ］を行い，RSA アルゴリズムを用いたディジタル署名を利用することによって［ f ］及び改ざんの検知を行っている。

設問1　図2でZ＝11，X＝7，Y＝5の場合，DH鍵として正しい値を，解答群の中から選べ。

解答群

ア　2　　イ　5　　ウ　7　　エ　10　　オ　13

設問2　図3中の［　　］に入れる適切な答えを，解答群の中から選べ。

a～cに関する解答群

ア　共通鍵　　イ　受信側の公開鍵　　ウ　受信側の秘密鍵

エ　送信側の公開鍵　　オ　送信側の秘密鍵

設問3　(4)のIPsecを利用したVPNの導入効果に関する記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

dに関する解答群

ア　アプリケーション　　イ　データリンク　　ウ　トランスポート

エ　ネットワーク

e，fに関する解答群

ア　DoS攻撃の対策　　イ　ウイルス感染の検知

ウ　セキュリティホールの修正　　エ　送受信するデータの圧縮

オ　盗聴の対策　　カ　なりすましの検知

問5　ソフトウェアのテスト設計に関する次の記述を読んで，設問1～3に答えよ。

システムインテグレータの N 社は，開発したプログラムに対するバグの摘出漏れの削減を目的として，テストの方法を見直している。

〔N社のテスト方法に関する説明〕

N社では主にホワイトボックス法の一つである制御フローテストで，開発したプログラムのテストを実施している。

制御フローテストは，プログラムを構成する最小単位である命令，経路，判定条件に着目し，テスト計画時に定めたカバレッジ基準を満たすテストケース，テストデータを作成して，開発したプログラムの動作を確認するテスト方法である。

カバレッジ基準としては，テストにおいて全ての命令文を1回は実行する命令網羅，全ての分岐について分岐後の全ての経路を1回は実行する判定条件網羅（以下，分岐網羅という）などがある。

N社は，カバレッジ基準として分岐網羅を採用している。

〔N社が採用している分岐網羅の判定条件に関する説明〕

分岐の判定条件には，一つの条件だけを評価する単独条件と，二つ以上の単独条件を and 又は or で組み合わせて評価する複数条件がある。単独条件と複数条件の例を次に示す。

例　$\underbrace{\underbrace{(a>b)}_{\text{単独条件}}\ \text{and}\ \underbrace{(a<c)}_{\text{単独条件}}}_{\text{複数条件}}$

ここで，プログラムの実行時に，複数条件については短絡評価を行うものとする。短絡評価とは，複数条件を構成する単独条件を左から右へ向かって順に評価し，複数条件の結果が確定したら，残りの単独条件を評価しない方法である。例えば，二つの単独条件を and で組み合わせた複数条件の場合，一つ目の単独条件を評価した結果が偽ならば，複数条件は二つ目の単独条件に関係なく必ず偽になるので，二つ目の単独条件を評価しない。

設問1　N 社が採用している分岐網羅の判定条件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。

図 1 はテスト対象のプログラムの例，表 1 はこのプログラムのテストケースの例である。N 社が採用している分岐網羅の判定条件に従って，このテストケースを用いて，図 1 のプログラムをテストしたとき，テストケース①では［ a ］結果となり，テストケース②では［ b ］結果となる。

○プログラム(整数型：x，整数型：a，整数型：b，整数型：c，整数型：d)

```
x > 10
 • func1()
(a < 10) or (b < 20)
 • func2()
 • func3()
(c > 10) and (d > 10)
 • func4()
 • exit        /* 繰返しを抜ける */
 • func5()
 • func6()
```

図 1　テスト対象のプログラムの例

表 1　テストケースの例

	テストデータ				
変数	x	a	b	c	d
テストケース①	11	9	19	10	10
テストケース②	11	10	20	11	11

a，bに関する解答群

ア　b ＜ 20 が評価されない

イ　b ＜ 20 とc ＞ 10 が評価されない

ウ　b ＜ 20 とd ＞ 10 が評価されない

エ　c ＞ 10 が評価されない

オ　c ＞ 10 とd ＞ 10 が評価されない

カ　d ＞ 10 が評価されない

キ　全ての単独条件が評価される

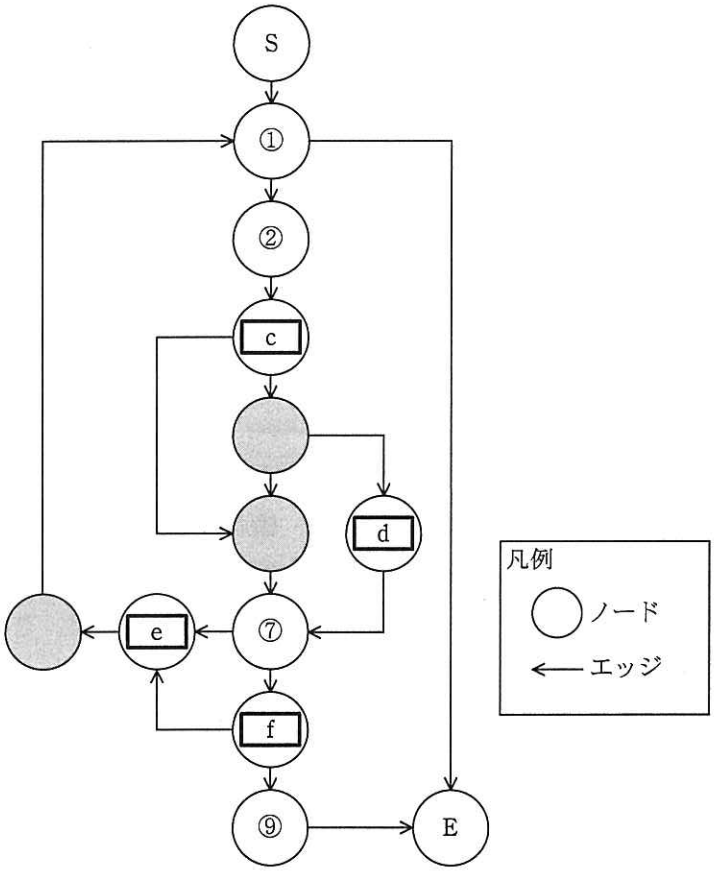
設問2　プログラムの制御構造は，制御フローグラフで記述することができる。制御フローグラフは，処理を逐次実行する命令，繰返し命令，分岐命令に分け，それぞれを処理ブロック（以下，ノードという）として処理の実行順に有向線分（以下，エッジという）で結んだグラフである。ここで，複数条件は，それぞれの単独条件に分解して制御フローグラフに置き換える。

図2は，図1のテスト対象のプログラムの例にノード番号①～⑪を付与したものであり，図3は，それに対応する制御フローグラフである。図3のノード番号は，図2中のノード番号に対応する。図3のノードSとノードEは，それぞれプログラムの入口と出口を表す特別なノードであり，テスト対象のプログラムの例には対応する処理はない。図3の制御フローグラフ中の［　　］に入れる適切なノード番号を，解答群の中から選べ。

○プログラム(整数型：x，整数型：a，整数型：b，整数型：c，整数型：d)

```
① x ＞ 10
② • func1()
③ (a ＜ 10) or ④ (b ＜ 20)
  ⑤ • func2()
  ⑥ • func3()
⑦ (c ＞ 10) and ⑧ (d ＞ 10)
  ⑨ • func4()
    • exit        /* 繰返しを抜ける */
  ⑩ • func5()
⑪ • func6()
```

図2　図1にノード番号を付与したプログラムの例



注記　網掛けの部分は表示していない。

図３　図２のプログラムの例に対応する制御フローグラフ

c～fに関する解答群

ア　③　　イ　④　　ウ　⑤　　エ　⑥　　オ　⑧

カ　⑩　　キ　⑪

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。

図1のプログラムのテストにおいて，N社が採用している分岐網羅の場合，最低限必要となるテストケースの数は［　g　］である。

これに対して，制御フローグラフから経路を抽出してテストケースを作成する方法がある。制御フローグラフの全てのエッジとノードを網羅する，最小の経路の数（S）は，次の式で求められる。

S＝エッジの数 － ノードの数 ＋2

抽出した経路に対応した S 個のテストケースについてテストを行うことによって，分岐網羅以上の高いカバレッジを保証することができる。

図3の制御フローグラフから，Sを求めると［　h　］となる。N社は，バグの摘出漏れの削減を目的として，制御フローグラフに基づくテストケースでプログラムのテストをすることとした。

g，hに関する解答群

ア　2　　イ　3　　ウ　4　　エ　5　　オ　6

カ　7

問6　プロジェクトの実績管理に関する次の記述を読んで，設問1，2に答えよ。

製造会社のA社は，業務改革に伴い，新システムを開発している。

〔プロジェクトの概要〕

(1) 新システムは，三つのサブシステムから成り，総数 120 本のプログラムで構成されている。プログラムには，複数のサブシステムで使用される共通プログラムとサブシステム固有のプログラム（以下，固有プログラムという）の2種類がある。プログラムの本数の内訳は，共通プログラムが30本，固有プログラムが90本である。

(2) 新システムをプロジェクトのメンバ10名で分担して開発する。

(3) プログラム 1 本当たりに掛かる工数は，過去のプロジェクト実績を基に算出している。内部設計に掛かるプログラム1本当たりの工数（以下，内部設計の標準工数という）は，共通プログラム，固有プログラムともに8人時／本である。また，プログラミングに掛かるプログラム1本当たりの工数（以下，プログラミングの標準工数という）は，共通プログラム，固有プログラムともに10人時／本である。

(4) 外部設計工程の終了時に見積もった内部設計工程とプログラミング工程の作業計画を表1に示す。表中の計画本数及び計画工数は，各週で作成するプログラムの本数及び作成に掛かる作業の工数の計画値である。

表1　内部設計工程とプログラミング工程の作業計画

工程		第1週	第2週	第3週	第4週	第5週	第6週	合計
内部設計	線表	←―――	―――	―――	―――	―――	―――→	
	計画本数	50	50	20				120
	計画工数	400	400	160				960
プログラミング	線表			←―――	―――	―――	―――→	
	計画本数			24	40	40	16	120
	計画工数			240	400	400	160	1,200

注記　計画工数の単位は人時である。

(5) 内部設計での 1 人時当たりに作成できるプログラム本数を内部設計の生産性といい，プログラミングでの 1 人時当たりに作成できるプログラム本数をプログラミングの生産性という。

(6) 共通プログラムの内部設計を先に行い，完了した後に固有プログラムの内部設計を行う。同様に，共通プログラムのプログラミングが完了した後に，固有プログラムのプログラミングを行う。

(7) 内部設計及びプログラミングにおける生産性は，計画値及び実績値ともにメンバ全員等しいものとする。ここで，各メンバの 1 日の作業時間の上限は 8 時間であり，1 週間の作業日数は 5 日である。

設問 1 内部設計工程の途中段階における進捗状況の分析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。

内部設計工程の開始から 2 週間が経過した時点での各週の実績値を表 2 に示す。

表 2 内部設計工程の開始から 2 週間が経過した時点での実績値

	第 1 週	第 2 週
実績本数	40	50
実績工数	380	400

注記 1 実績工数の単位は人時である。
注記 2 共通プログラムと固有プログラムのそれぞれの生産性は，第 1 週と第 2 週では変わらなかった。

プログラム総本数に対する内部設計が完了した本数の比率を，内部設計の進捗率（%）という。内部設計工程の開始から 2 週間が経過した時点で，実績値に基づいた進捗率が計画値に基づいた進捗率を約［ a ］ポイント下回っており，進捗にやや遅れが生じている。また，内部設計の生産性の実績値は，［ b ］が計画値を下回っており，全体としても内部設計の生産性の実績値が計画値を下回っている。

しかし，残りのプログラムを，表 2 が示す実績値と同じ生産性で内部設計ができると仮定した場合，残り工数の予測値は［ c ］人時となり，進捗にやや遅れがあるものの，内部設計をこのまま進めても第 3 週末までに終えられると判

断した。

aに関する解答群

ア 4　　イ 8　　ウ 11　　エ 15　　オ 20

bに関する解答群

ア 共通プログラム

イ 共通プログラム及び固有プログラム

ウ 固有プログラム

cに関する解答群

ア 240　　イ 260　　ウ 300　　エ 320

設問2 プログラミング工程の計画変更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。

内部設計の生産性の実績値が計画値を下回っていた原因は，今回のプロジェクトが過去のプロジェクトと比べて，予定した内部設計を上回るレベルまで詳細に内部設計が行われていたためである。このため，プログラミングの生産性では，実績値が計画値を上回ることが期待できる。

そこで，プログラミングの標準工数を見直し，共通プログラムを8人時／本に，固有プログラムを9人時／本に変更することにした。また，1本のプログラムを複数メンバで作ることも可能とした。変更後の標準工数を用いてプログラミング工程の総工数を再計算すると，当初の総工数よりも［ d ］人時の工数削減が見込める。この工数削減の見込み値を基に，プログラミング工程の各週の作業計画を変更することにした。

プログラミング期間を短縮するだけであれば，表3に示す作業計画で，メンバ全員に対して，第4～6週の日々の上限時間までプログラミングに専念させればよい。しかし今回は，テストの開始を当初計画よりも前倒ししたいと考え，第4～6週の期間にプログラミングとテストを並行して実施することにした。また，

第3週のプログラミングの計画本数は10本とした。

ここで，内部設計工程の開始から2週間経過した時点での予測どおり，内部設計工程は第3週で終了すると仮定し，第4週及び第5週のいずれの週も週末の時点において仕掛中のプログラムはないようにする。

表3　プログラミング期間を短縮する作業計画

週	第3週	第4週	第5週	第6週
計画本数	10	e		

テストを並行して実施する作業計画では，第4～6週の各週のプログラミング工数ができるだけ均等になるようにして，余った時間でテストを実施する。表4は，それを考慮してプログラミングの本数を配分したものであり，第4～6週の各週のプログラミング工数の差は最大 f 人時である。

表4　テストを並行して実施する作業計画

週	第3週	第4週	第5週	第6週
計画本数	10	38	36	36

dに関する解答群

ア　90　　イ　120　　ウ　150　　エ　180

eに関する解答群

ア	44	44	22	イ	44	45	21
ウ	45	44	21	エ	45	45	20
オ	45	46	19	カ	46	45	19
キ	46	44	20				

fに関する解答群

ア　2　　イ　6　　ウ　10　　エ　16　　オ　18

問7　販売管理システムの見直しを伴う業務改善に関する次の記述を読んで，設問 1～3 に答えよ。

A社は美容用品（以下，商品という）の卸売業者であり，地方都市Xを中心に5か所の営業所をもっている。メーカから仕入れた商品を，理容店や美容室など（以下，顧客という）に販売している。A 社は 10,000 種類以上の商品を取り扱っており，商品は，各営業所の倉庫に保管されている。在庫数は販売管理システムで共通管理されているが，受発注処理や在庫管理は営業所ごとに行っている。各営業所には，営業担当と事務担当がいる。

〔業務の現状〕

(1)　注文は，営業担当が顧客を訪問して受ける場合と，電話やファクシミリで受ける場合がある。営業担当は注文を受けると，口頭で事務担当に注文内容を連絡する。

(2)　営業担当は，在庫があれば，注文を受けた翌日の午前中までに顧客に商品を届ける。営業担当が顧客に商品を届けるのは，市場の生の声や顧客のニーズを聴くためである。

(3)　事務担当は，販売管理システムを使い，次の手順で受発注処理を行っている。

①　営業担当から注文内容の連絡を受けると，販売管理システムに注文内容を入力し，自営業所の在庫を確認する。

②　自営業所に必要な在庫数がない場合，販売管理システムで他営業所の在庫を確認する。

③　他営業所に在庫がある場合は，他営業所から商品を移動してもらうように電話で手配する。他営業所では発送作業を行い，自営業所では受取り作業を行う。商品の移動は当日中に行われる。

④　他営業所の在庫を移動しても必要数に満たない場合は，自営業所で販売管理システムから発注書を出力して，ファクシミリでメーカに発注する。メーカからの納品には数日掛かる。

(4)　事務担当は，在庫をもつ商品の在庫管理を定量発注方式で行っている。定量発注方式とは，在庫数があらかじめ設定した数量（以下，発注点という）まで下がったときに，一定数量（以下，発注量という）を発注して在庫を補充する方法である。

各営業所で，販売管理システムに商品ごとの発注点と発注量を設定している。

(5) 事務担当は，商品が各営業所の倉庫から出庫される時点で販売管理システムに出庫入力し，販売管理システムは在庫数から出庫数を減らす処理を行う。

これまで，在庫をもつ商品と在庫をもたない商品を営業所ごとに決めて，在庫数の削減に努めてきた。しかし，更なる在庫数削減のために，各営業所の倉庫をなくして，在庫を集約して一元管理するための物流倉庫を新たに設置することを決定した。物流倉庫には，物流担当を置く。

これに併せて，A 社では営業所の業務の効率化を図ることを決定した。そこで，業務の現状分析から問題点の洗出しを行い，それらを解決するための改善案を取りまとめた。

業務の効率化に当たり，営業担当が顧客に商品を届ける方法は従来どおりとする。

〔問題点〕

(1) 営業担当から事務担当に口頭で注文内容を伝えているので，連絡ミスが発生している。

(2) 販売管理システムの受発注処理と在庫数を減らす処理のタイミングに差があるので，販売管理システムの在庫数を見ても，顧客に納入可能な在庫数を正確に把握できない。

(3) 取扱い商品の増加とともに，事務担当が受発注処理や在庫管理に費やす時間が増えている。他営業所の在庫確認や，営業所間での商品の発送作業と受取り作業にも手間が掛かっている。

〔改善案〕

(1) 営業担当は，注文内容を電子情報にして物流担当に伝える。物流担当は注文内容を確認し，当日中に営業所に商品が到着するように発送する。

(2) 顧客からの注文を受けたら速やかに，在庫数を減らす処理を行う。

(3) メーカへの発注は営業所から行わず，物流担当がまとめて行う。

設問1　改善案(1)～(3)を実現するために，販売管理システムの改修の要求事項を設定した。要求事項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。

(1)　［　a　］を伝えて即時に販売管理システムに反映するために，営業担当が販売管理システムと連携した携帯用の注文入力端末で注文入力できるようにする。

(2)　注文入力処理によって，受注処理と在庫数を減らす処理を行うようにする。

(3)　［　b　］できるように，物流担当が発注点と発注量の設定を行えるようにする。

aに関する解答群

ア　営業所に市場の生の声　　イ　営業所に注文内容

ウ　顧客に在庫数　　エ　物流担当に注文内容

bに関する解答群

ア　営業所に商品発送　　イ　顧客に一括発送

ウ　メーカから商品発送　　エ　メーカに一括発注

設問2　改善案を実施することで得られる効果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。

(1)　各営業所の在庫をなくし，物流倉庫に集約することによって，［　c　］の［　d　］。

(2)　営業担当が入力した注文内容で受注処理を行うことによって，注文連絡ミスによる無駄な作業がなくなる。

cに関する解答群

ア　営業担当　　イ　顧客

ウ　事務担当　　エ　販売管理システム

dに関する解答群

ア　顧客を訪問する回数が削減される

イ　受発注処理や在庫管理に費やされる時間が削減される

ウ　全ての商品の発注点と発注量が同じになる

エ　ニーズを効率よく聴くことができる

設問3　A社では，業務改善と販売管理システムの改修に先立ち，商品Kと商品Lを対象にして，物流倉庫に集約して一元管理することによる在庫数の削減効果を評価することにした。削減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。ここで，［　e　］は小数第1位を四捨五入するものとする。

在庫数は，受注数量に合わせてもつ在庫数（以下，回転在庫数という）と，欠品を起こさないようにもつ在庫数（以下，安全在庫数という）の和で表される。

全体の受注数量に対して各拠点の受注数量が占める割合（以下，需要比率という）が異なるN か所の在庫拠点を，1拠点に集約したときの総在庫数を推定するための算出式は，次のとおりである。

$$\text{総在庫数} = \sum_{i=1}^{N} \text{拠点 i の回転在庫数} + \frac{\sum_{i=1}^{N} \text{拠点 i の安全在庫数}}{\sum_{i=1}^{N} \sqrt{\text{拠点 i の需要比率}}}$$

表1は，物流倉庫に在庫を集約する前の，A社の商品Kと商品Lの各営業所の在庫数及び需要比率である。

表１　商品Ｋと商品Ｌの各営業所の在庫数及び需要比率

	商品Ｋ			商品Ｌ		
	在庫数		需要比率	在庫数		需要比率
	回転在庫数	安全在庫数		回転在庫数	安全在庫数	
営業所Ｐ	25	100	0.2	90	445	0.9
営業所Ｑ	30	150	0.3	5	25	0.1
営業所Ｒ	20	100	0.2	0	0	0.0
営業所Ｓ	10	70	0.2	0	0	0.0
営業所Ｔ	10	50	0.1	0	0	0.0
合計			1.0	95	470	1.0

注記　網掛けの部分は表示していない。

在庫拠点を１拠点に集約したときの総在庫数の算出式と表１から，物流倉庫に在庫を集約して一元管理した場合，商品Ｋの総在庫数は［　e　］，商品Ｌの総在庫数は 465 となる。ここで，$\sqrt{0.1} = 0.32$，$\sqrt{0.2} = 0.45$，$\sqrt{0.3} = 0.55$，$\sqrt{0.9} = 0.95$ を用いる。

在庫を集約して一元管理をしたときの在庫数の削減効果は，［　f　］。

eに関する解答群

ア　283　　イ　307　　ウ　465　　エ　565

fに関する解答群

ア　需要比率が各営業所で均一に近い商品の場合に高い

イ　需要比率が特定の営業所に偏在している商品の場合に高い

ウ　需要比率が偏在している商品と均一に近い商品とでは，どちらが高いかは一概には言えない

エ　どの商品も，各営業所の需要比率に関係なく同じである

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1，2に答えよ。

〔プログラムの説明〕

英字A～Zから構成される文字列を圧縮する副プログラムCompress，及び圧縮された文字列を復元する副プログラムDecompressである。

〔圧縮処理の説明〕

副プログラムCompressでは，配列Plaindataに格納されている圧縮前の文字列を受け取り，圧縮後の文字列を配列Compresseddataに格納する。図1に示す文字列“ABCDEFABCDABCDEF”を例として，圧縮処理の内容を説明する。

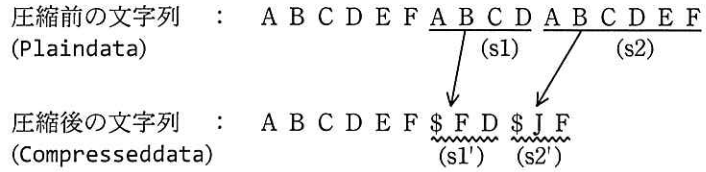


図1　文字列の圧縮例

(1)　図1の圧縮前の文字列“ABCDEFABCD…”のように，“ABCD”という同じ文字の並びが複数回出現する場合，二つ目以降の“ABCD”を圧縮列に置き換えることによって，文字列の長さを短くする。図1では，Plaindataに格納されている圧縮前の文字並び(s1)，(s2)を圧縮列(s1')，(s2')に置き換えてCompresseddataに格納している。

(2)　圧縮列は，制御記号，距離，文字数の三つの部分から成る。

①　制御記号は，“$”であり，圧縮列の先頭を表す。

②　距離と文字数は，圧縮前の文字列において，“この場所の文字並びは，何文字前（距離）の文字から始まる何文字の長さ（文字数）の文字並びと同じである”という内容を意味する。距離と文字数の最大値は26とし，1，2，…，26をA，B，…，Zで表す。図1の例では，圧縮前の文字並び(s1)は，(s1)の先頭文字の

6文字前から始まる長さ4の文字並び“ＡＢＣＤ”と一致するので，圧縮列(s1')は“$ＦＤ”となる。また，圧縮前の文字並び(s2)は，(s2)の先頭文字の10文字前から始まる長さ6の文字並び“ＡＢＣＤＥＦ”と一致するので，圧縮列(s2')は“$ＪＦ”となる。

(3) 圧縮処理は，配列Plaindataに格納されている圧縮前の文字列中で，圧縮する文字並びを検索する検索処理と，検出した文字並びを圧縮列に置き換えて配列Compresseddataに格納する置換処理から成る。

(4) 圧縮する文字並びの検索処理の内容を次に示す。

① 圧縮処理の対象となる文字並びを圧縮文字並び，圧縮文字並びの先頭位置を圧縮文字位置という。圧縮列の文字数は3なので，一致する文字数が4以上の圧縮文字並びの場合に圧縮列に置き換える。したがって，最初の圧縮文字位置は，図2に示すように圧縮前の文字列の先頭から5文字目となる。この文字位置から圧縮文字位置を文字列の後方に向かって移動させながら検索処理②，③を行う。

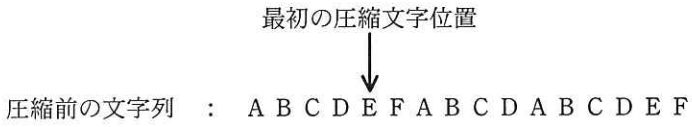


図2 最初の圧縮文字位置

② 圧縮文字並びの比較対象とする文字並びを比較文字並び，比較文字並びの先頭位置を比較文字位置という。最初の比較文字位置は，図3に示すように圧縮文字位置の4文字前とする。

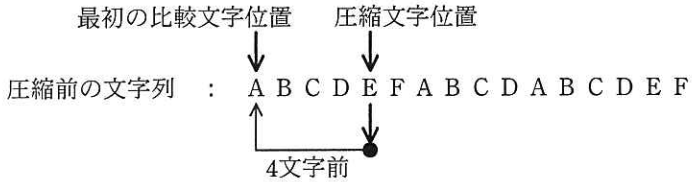


図3 最初の比較文字位置

③　比較文字位置を文字列の先頭に向かって，圧縮文字位置から最大 26 文字前まで移動させながら，次の(a)，(b)を行う。

(a)　圧縮文字位置と比較文字位置から文字列の後方に向かって，圧縮文字並びと比較文字並びが何文字一致するかを調べる。

図 4 は，圧縮文字位置が 7 文字目のときに，比較文字位置を文字列の先頭に向かって p_1，p_2，p_3 と移動しながら，一致する文字を検索している様子を示している。比較文字位置 p_1 と p_2 では，1 文字目が一致していない。比較文字位置 p_3 では 1 文字目が一致し，一致する文字数は 4 となる。

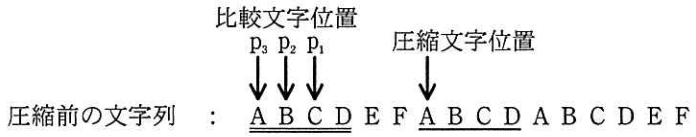


図 4　文字列の比較

(b)　圧縮する文字並びは，それよりも前にある比較文字並びの中で，一致する文字数が 4 以上で最も多い比較文字並びとする。圧縮する文字並びの対応例を図 5 に示す。図 5 では，圧縮文字位置から始まる圧縮文字並び“A B C D …”と，それよりも前方に出現する比較文字並びの中で，一致する文字数が 4 以上の比較文字並びは，4 文字前の(s1)“A B C D”と 10 文字前の(s0)“A B C D E F”である。したがって，一致する文字数が多い(s0)に対応させて，(s2)“A B C D E F”を圧縮列に置き換えることになる。

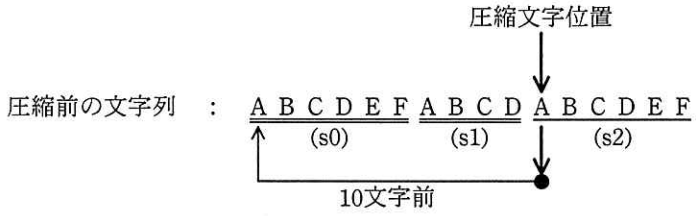


図 5　圧縮する文字並びの対応例

(5) (4)の検索処理の結果に対する置換処理の内容を次に示す。

① 一致する文字数が3以下の場合には，圧縮列に置き換えないで，圧縮文字位置の文字を配列Compresseddataに格納する。

② 一致する文字数が4以上の場合には，その文字数が最も多い比較文字並びに対応する圧縮列を配列Compresseddataに格納する。

〔副プログラムCompressの引数の仕様〕

Compressの引数の仕様は，次のとおりである。各配列の添字は，0から始まる。

引数名	データ型	入力／出力	意味
Plaindata[]	文字型	入力	圧縮前の文字列が格納されている1次元配列
Plength	整数型	入力	圧縮前の文字列の長さ（1以上）
Compresseddata[]	文字型	出力	圧縮後の文字列が格納される1次元配列
Clength	整数型	出力	圧縮後の文字列の長さ

〔復元処理の説明〕

副プログラムDecompressでは，配列Compresseddataに格納されている圧縮された文字列を受け取り，復元後の文字列を配列Plaindataに格納する。復元処理の内容を次に示す。

(1) 配列Compresseddataの先頭から圧縮された文字列を順に調べる。

(2) 文字が制御記号でなければ，その文字をそのまま配列Plaindataに格納する。

(3) 文字が制御記号ならば，圧縮列の距離，文字数から，圧縮前の文字並びを復元して配列Plaindataに格納する。

〔副プログラムDecompressの引数の仕様〕

Decompressの引数の仕様は，次のとおりである。各配列の添字は，0から始まる。

引数名	データ型	入力／出力	意味
Compresseddata[]	文字型	入力	復元前の文字列が格納されている1次元配列
Clength	整数型	入力	復元前の文字列の長さ（1以上）
Plaindata[]	文字型	出力	復元後の文字列が格納される1次元配列
Plength	整数型	出力	復元後の文字列の長さ

副プログラム Compress と Decompress で使用している関数 IntToAlphabet と AlphabetToInt の仕様は，次のとおりである。

〔関数 IntToAlphabet の仕様〕

整数 1 ～ 26 を順に英字 A ～ Z に変換する。IntToAlphabet の引数と返却値の仕様は，次のとおりである。

引数／返却値	データ型	意味
引数	整数型	整数1～26の値
返却値	文字型	引数に対応した英字

〔関数 AlphabetToInt の仕様〕

英字 A ～ Z を順に整数 1 ～ 26 に変換する。AlphabetToInt の引数と返却値の仕様は，次のとおりである。

引数／返却値	データ型	意味
引数	文字型	英字A～Zの文字
返却値	整数型	引数に対応した整数の値

〔プログラム 1〕

```
○副プログラム：Compress(文字型：Plaindata[]，整数型：Plength，
                         文字型：Compresseddata[]，整数型：Clength)
○文字型：Esym
○整数型：Pindex，Cindex
○整数型：Maxfitnum，Maxdistance，Distance，Fitnum
```

```
Plength ≧ 1
  • Esym ← "$"              /* 制御記号の設定 */
  • Pindex ← 0              /* 圧縮前の文字列の文字位置を初期化 */
  • Cindex ← 0              /* 圧縮後の文字列の文字位置を初期化 */
  (Pindex < Plength) and (Pindex < 4)
    • Compresseddata[Cindex] ← Plaindata[Pindex]
    • Cindex ← Cindex + 1
    • Pindex ← Pindex + 1

  Pindex < Plength
    • Maxfitnum ← −1
    • Maxdistance ← −1
    • Distance ← 4
    (Distance ≦ 26) and ( [      a      ] )
      • Fitnum ← 0
      /* 同じ文字並びの文字数を調べる */
      (Fitnum < Distance) and ((Pindex + Fitnum) < Plength)
        /* 文字が一致するかどうかを調べる */
        Plaindata[Pindex + Fitnum] ≠ [      b      ]
          • break    /* 最も内側の繰返し処理を終了する */
        • Fitnum ← Fitnum + 1
      /* 文字数が 4 以上，かつ，最も多いかどうかを調べる */
      (Fitnum ≧ 4) and (Maxfitnum < Fitnum)
        • Maxfitnum ← Fitnum
        • Maxdistance ← Distance
      • [      c      ]
    Maxfitnum = −1
      • Compresseddata[Cindex] ← Plaindata[Pindex]
      • Cindex ← Cindex + 1
      • Pindex ← Pindex + 1
    ---
      • Compresseddata[Cindex] ← Esym
      • Compresseddata[Cindex + 1] ← IntToAlphabet(Maxdistance)
      • Compresseddata[Cindex + 2] ← IntToAlphabet(Maxfitnum)
      • Cindex ← Cindex + 3
      • [      d      ]

  • Clength ← Cindex
```

〔プログラム2〕

```
○副プログラム：Decompress(文字型：Compresseddata[]，整数型：Clength，
                           文字型：Plaindata[]，整数型：Plength)
○文字型：Esym
○整数型：Pindex，Cindex
○整数型：Num，Fitcnt，Start
▲ Clength ≧ 1
  • Esym ← "$"
  • Cindex ← 0
  • Pindex ← 0
  ■ Cindex ＜ Clength
    ▲ Compresseddata[Cindex] ≠ Esym
      • Plaindata[Pindex] ← Compresseddata[Cindex]
      • Pindex ← Pindex + 1                          ←―― α
      • Cindex ← Cindex + 1
    ―
      • Num ← AlphabetToInt(Compresseddata[Cindex + 2])
      • Start ← AlphabetToInt(Compresseddata[Cindex + 1])
      ■ Fitcnt: 0，Fitcnt ＜ Num，1
        • Plaindata[Pindex + Fitcnt] ← [      e      ]
      ■
      • Pindex ← Pindex + Num
      • Cindex ← Cindex + 3
    ▼
  ■
  • Plength ← Pindex
▼
```

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　Pindex − Distance ≧ 0　　イ　Pindex − Plength ≧ 0

ウ　Plength − Distance ≧ 0　　エ　Plength − Pindex ≧ 0

bに関する解答群

ア　Plaindata[Pindex + Distance]

イ　Plaindata[Pindex + Distance + Fitnum]

ウ　Plaindata[Pindex − Distance]

エ　Plaindata[Pindex − Distance + Fitnum]

cに関する解答群

ア　Cindex ← Cindex + 1　　イ　Distance ← Distance + 1

ウ　Fitnum ← Fitnum + 1　　エ　Pindex ← Pindex + 1

オ　Plength ← Plength + 1

dに関する解答群

ア　Pindex ← Pindex + 1

イ　Pindex ← Pindex + 3

ウ　Pindex ← Pindex + Maxdistance

エ　Pindex ← Pindex + Maxfitnum

eに関する解答群

ア　Compresseddata[Pindex + Start + Fitcnt]

イ　Compresseddata[Pindex − Start + Fitcnt]

ウ　Plaindata[Pindex + Start + Fitcnt]

エ　Plaindata[Pindex − Start + Fitcnt]

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。

次の文字列を圧縮した文字列を副プログラム Decompress を使って復元する場合，プログラム2のαの部分は［ f ］回実行される。

文字列　：　ABCDEFGABCDEABCDFEFGABCD

fに関する解答群

ア　3　　イ　4　　ウ　5

エ　6　　オ　7　　カ　8

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1～3に答えよ。

二つの英単語を辞書順で比較する関数 diccmp を作成した。

一般に，英大文字，英小文字，記号文字を含む英単語を一定の順に並べる方法として，次の例のように，文字列中の文字コード順に並べる方法の他に，英和辞典や書籍の欧文索引のような順序で並べる方法（以下，辞書順という）がある。

例　4個の単語“A.M.”，“CPU”，“all”及び“best”を並べる方法

文字コード順：“A.M.” ＜ “CPU” ＜ “all” ＜ “best”

辞書順：　　“all” ＜ “A.M.” ＜ “best” ＜ “CPU”

検索などの場合は，英単語を辞書順に整列しておくと検索の効率がよく，精度も向上する。

関数 diccmp は，英大文字，英小文字，“-”及び“.”を含む二つの英単語の大小を辞書順で比較する。

〔プログラムの説明〕

関数 diccmp は，二つの英単語の大小を辞書順で比較する。その引数と返却値は，次のとおりである。

引数：	word1, word2	英単語を格納した文字列	
返却値：	負の値	word1＜word2	（例：“map” ＜ “May”）
	0	word1＝word2	（例：“Mr.” ＝ “Mr.”）
	正の値	word1＞word2	（例：“U.S.” ＞ “US”）

(1) 引数 word1 と word2 は，いずれも次の条件を満たしている。

① 各文字は，英大文字（“A” ～ “Z”），英小文字（“a” ～ “z”），“-” 又は “.” のいずれかである。英大文字と英小文字を合わせて，英字という。

② 文字列の長さは，1 以上 30 以下である。

③ 文字列の先頭の文字は，英字である。

④ 文字列中の “-” 及び “.” の直前の文字は，英字である。

C

(2) 辞書順での比較の方法は，次のとおりである。

① 各引数の文字列から，基本文字列と文字情報列を生成する。基本文字列とは，引数の文字列から英字だけを順に取り出して，大文字を小文字に変換した文字列である。文字情報列とは，基本文字列中の各文字に対応する 6 種類の文字情報（大文字，小文字，大文字 “-” 付き，小文字 “-” 付き，大文字 “.” 付き，小文字 “.” 付き）を表す情報の列である。ここで，“-” 及び “.” は，直前の英字に属する文字情報とみなす。

例 引数の文字列：“Fri.” → 基本文字列：

f	r	i

文字情報列：

大文字	小文字	小文字 “.” 付き

② 各引数の基本文字列同士を関数 strcmp で比較し，異なっていれば，その返却値（負の値 又は 正の値）を返す。一致していれば，各引数の文字情報列同士を関数 strcmp で比較し，その返却値（0，負の値 又は 正の値）を返す。

③ 6 種類の文字情報について，その値の大小関係はプログラム中で定めている。

(3) プログラム中で使用しているライブラリ関数の概要は，次のとおりである。

isalpha(c)：c が英字のとき 0 以外の値を返し，それ以外のとき 0 を返す。
islower(c)：c が英小文字のとき 0 以外の値を返し，それ以外のとき 0 を返す。
strcmp(s1, s2)：文字列 s1 と s2 を比較し，s1 ＜ s2 のとき負の値を，s1 ＝ s2 のとき 0 を，s1 ＞ s2 のとき正の値を，それぞれ返す。
tolower(c)：c が英大文字のときその文字に対応する英小文字を返し，それ以外のとき c を返す。

〔プログラム〕

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

int  diccmp(char *, char *);
void diccnv(char *, char *, char *);

int diccmp(char *word1, char *word2) {
    int  rc;

    char char1[31], char2[31], attr1[31], attr2[31];
    diccnv(word1, char1, attr1);
    diccnv(word2, char2, attr2);
    rc = strcmp(char1, char2);
    if (rc == 0)
        rc = strcmp(attr1, attr2);
    return rc;
}

void diccnv(char *wordx, char *charx, char *attrx) {
    int  ch, cpos, wpos;

    cpos = 0;
    wpos = 0;
    while (wpos < 30 && (ch = wordx[wpos++]) != '\0') {
        if (isalpha(ch)) {
            if (islower(ch)) {
                charx[cpos] = ch;
                attrx[cpos] = '0';
            }
            else {
                charx[cpos] = tolower(ch);
                attrx[cpos] = '4';
            }
            ch = wordx[wpos];
            if (ch == '-' || ch == '.') {
                if (ch == '-')
                    attrx[cpos] += 1;
                else
                    attrx[cpos] += 2;
                wpos++;
            }
            cpos++;
        }
    }
    charx[cpos] = '\0';
    attrx[cpos] = '\0';
}
```

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。

引数 word1 の内容を“A.D.”，word2 の内容を“ad-”として，関数 diccmp を実行した。関数 diccmp 中の return 文を実行する時点で，attr1 の内容は“［　a　］”，attr2 の内容は“［　b　］”，rc の内容は［　c　］となる。

a，b に関する解答群

ア　01	イ　02	ウ　11	エ　22
オ　41	カ　42	キ　55	ク　66

c に関する解答群

ア　0	イ　正の値	ウ　負の値

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。

関数 diccmp を用いて，4 個の単語“CO”，“Co.”，“co-”及び“co.”を相互に比較したとき，その大小関係は，［　d　］となる。

d に関する解答群

ア　“CO” ＜ “Co.” ＜ “co-” ＜ “co.”	イ　“CO” ＜ “Co.” ＜ “co.” ＜ “co-”
ウ　“Co.” ＜ “CO” ＜ “co-” ＜ “co.”	エ　“co-” ＜ “co.” ＜ “Co.” ＜ “CO”
オ　“co.” ＜ “co-” ＜ “CO” ＜ “Co.”	カ　“co.” ＜ “co-” ＜ “Co.” ＜ “CO”

設問3　次の表1中の［　　］に入れる正しい答えを，解答群の中から選べ。

このプログラムは，引数 word1 及び word2 がプログラムの説明中の (1) に示した条件①～④を満たしているものとして作成している。しかし，引数が条件を満たさない場合のプログラムの動作についても確認しておきたい。そこで，引数が条件を満たさない場合のプログラムの動作を，表1にまとめた。

なお，文字列中の文字は，全て1バイト文字とする。

表1　引数が条件を満たさない場合のプログラムの動作

条件	条件を満たさない場合	プログラムの動作
①	文字列中に，英字，“-”，“.” 以外の文字がある。	
②	文字列の長さが0である。	[e]。
	文字列の長さが31以上である。	[f]までを有効とし，後続の文字を無視する。
③	先頭の文字が英字でない。	[g]。
④	“-” 及び “.” の直前の文字が英字でない。	

注記　網掛けの部分は表示していない。

eに関する解答群

ア　空文字列として扱い，プログラムは正常に終了する

イ　配列に何も値を設定せずに比較をするので，予期できない結果となる

ウ　配列の定義範囲外への書込みが発生するので，予期できない結果となる

エ　プログラムが終了しない

fに関する解答群

ア　30個目の英字

イ　30個目の英字（直後の文字が “-” 又は “.” の場合はその文字）

ウ　30文字目

エ　30文字目（30文字目が英字で31文字目が “-” 又は “.” の場合は31文字目）

gに関する解答群

ア　その文字が “-”，“.” 以外の場合はその文字を無視するが，“-” 又は “.” の場合は配列の定義範囲外への書込みが発生するので，予期できない結果となる

イ　その文字を無視する

ウ　配列の定義範囲外への書込みが発生するので，予期できない結果となる

エ　プログラムが終了しない

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〕

X社では，自社の事業所内にある4面のテニスコートについて，従業員が予約やキャンセル，空き状況の確認ができるシステムを開発することになった。このプログラムは，テニスコートの予約を受け付けるサブプログラムであり，利用希望者からの予約情報をパラメタで受け取り，予約管理ファイルで管理している予約状況と照らし合わせて，予約結果をパラメタで呼出し元に返却する。

(1) プログラムには二つのパラメタがある。各パラメタの様式は，図1のとおりである。

予約情報

予約番号 6桁	希望面数 1桁	希望日 8桁	希望時刻 2桁	希望時間 1桁

予約結果

結果 1桁	予約コート 1桁

4回繰返し

図1 パラメタの様式

① 予約番号には，予約希望に対して一意に割り振られる000001 ～ 999999の番号が格納されている。

② 希望面数には，希望するコートの面数が格納されている。

③ 希望日には，年，月，日が，それぞれ4桁，2桁，2桁の西暦で格納されている。

④ 希望時刻には，希望する利用開始時刻の“時”が，24時間表記で格納されている。利用可能な時間帯は8時から17時までであり，1時間単位で予約できる。

⑤ 希望時間には，希望する利用時間数が格納されている。

⑥ 結果には，希望どおりに予約できた場合は0を，予約できなかった場合は9を設定する。

⑦ 予約コートには，予約できたコートのコート番号を設定する。例えば3番コートを予約できた場合は3000を設定し，1番コートと2番コートを予約できた場合は1200を設定する。予約ができなかった場合は，0000を設定する。

⑧ コートは1番 ～ 4番の順に検索し，番号の小さいコートから確保する。

⑨ 予約情報の各項目の値は呼出し側プログラムで検査するので，①～⑤で示した範囲以外の値が設定されることはない。

(2) 予約管理ファイルは，日付を主キーとする図2に示すレコード様式の索引ファイルで，予約状況を管理するために使用する。

日付 8桁	予約状況（1番コート）				…	予約状況（4番コート）			
	8時台 6桁	9時台 6桁	…	16時台 6桁	…	8時台 6桁	9時台 6桁	…	16時台 6桁

図2 予約管理ファイルのレコード様式

① 日付には，年，月，日が，それぞれ4桁，2桁，2桁の西暦で格納される。

② 予約状況の各要素には，予約済の場合は予約番号が，予約が入ってない場合は0が格納される。

③ 予約が入っていない日付をキーとするレコードは存在しない。

〔プログラム〕

(行番号)

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD RSV-FILE.
 4  01 RSV-REC.
 5     02 RSV-DATE      PIC X(8).
 6     02 RSV-AREA.
 7        03 RSV-COURT  OCCURS 4.
 8           04 RSV-RNO    PIC 9(6) OCCURS 9.
 9  WORKING-STORAGE SECTION.
10  77 W-CHK         PIC 9(1).
11     88 CHK-OK     VALUE 0.
12     88 CHK-NG     VALUE 1.
13  77 COURT-CNT     PIC 9(1).
14  77 TIME-CNT      PIC 9(2).
15  77 TIME-START    PIC 9(2).
16  77 TIME-END      PIC 9(2).
17  77 KEEP-COURT    PIC 9(1).
18  77 BKUP-AREA     PIC X(54).
19  LINKAGE SECTION.
20  01 PRM-DATA.
21     02 PRM-RNO       PIC 9(6).
22     02 PRM-NUM       PIC 9(1).
23     02 PRM-DATE      PIC 9(8).
24     02 PRM-TIME      PIC 9(2).
```

```
25     02 PRM-HOURS      PIC 9(1).
26  01 RET-DATA.
27     02 RET-RSLT       PIC 9(1).
28        88 RSV-OK      VALUE 0.
29        88 RSV-NG      VALUE 9.
30     02 RET-COURT.
31        03 COURT-NO      PIC 9(1) OCCURS 4.
32  PROCEDURE DIVISION USING PRM-DATA RET-DATA.
33  MAIN-PROC.
34      SET RSV-OK TO TRUE.
35      INITIALIZE RET-COURT.
36      OPEN I-O RSV-FILE.
37      MOVE PRM-DATE TO RSV-DATE.
38      READ RSV-FILE
39         INVALID KEY      INITIALIZE RSV-AREA
40                          MOVE PRM-DATE TO RSV-DATE
41                          PERFORM RSV-PROC
42                          WRITE RSV-REC END-WRITE
43         NOT INVALID KEY PERFORM RSV-PROC
44                          IF RSV-OK THEN
45                             REWRITE RSV-REC END-REWRITE
46                          END-IF
47      END-READ.
48      CLOSE RSV-FILE.
49      EXIT PROGRAM.
50  RSV-PROC.
51      MOVE ZERO TO KEEP-COURT.
52      [          a          ].
53      COMPUTE TIME-END = TIME-START + PRM-HOURS.
54      PERFORM VARYING COURT-CNT FROM 1 BY 1
55              UNTIL COURT-CNT > 4 OR PRM-NUM = KEEP-COURT
56         [          b          ]
57         MOVE RSV-COURT(COURT-CNT) TO BKUP-AREA
58         PERFORM VARYING TIME-CNT FROM TIME-START BY 1
59                 UNTIL CHK-NG OR TIME-CNT >= TIME-END
60            IF RSV-RNO(COURT-CNT TIME-CNT) = ZERO THEN
61               MOVE PRM-RNO TO RSV-RNO(COURT-CNT TIME-CNT)
62            ELSE
63               SET CHK-NG TO TRUE
64               [          c          ]
65            END-IF
66         END-PERFORM
67         IF CHK-OK THEN
68            ADD 1 TO KEEP-COURT
69            MOVE COURT-CNT TO COURT-NO(KEEP-COURT)
70         END-IF
71      END-PERFORM.
72      IF [          d          ] THEN
73         SET RSV-NG TO TRUE
74         INITIALIZE RET-COURT
75      END-IF.
```

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～cに関する解答群

ア　COMPUTE TIME-START = PRM-TIME - 7

イ　MOVE BKUP-AREA TO RSV-COURT(COURT-CNT)

ウ　MOVE COURT-CNT TO KEEP-COURT

エ　MOVE PRM-TIME TO TIME-CNT

オ　MOVE RSV-RNO(COURT-CNT TIME-CNT) TO PRM-RNO

カ　MOVE TIME-CNT TO COURT-CNT

キ　SET CHK-NG TO TRUE

ク　SET CHK-OK TO TRUE

dに関する解答群

ア　CHK-NG

イ　CHK-OK

ウ　KEEP-COURT NOT = ZERO

エ　PRM-NUM NOT = KEEP-COURT

オ　PRM-NUM = KEEP-COURT

設問2　システムの運用を開始したところ，“隣り合わせのコートを希望する機能が欲しい”との要望が挙がった。コートは図3に示すとおりに配置されていて，2面を予約した利用者にフェンスを挟んだコート，例えば1番コートと3番コートが割り当てられたことがあったためである。

パラメタにデータ項目PRM-NEXTを追加し，1が設定されている場合，確保できたコートがフェンスを挟んでいたら予約はせずに，結果に4を設定して返却するようにプログラムを変更する。ただし，3面以上の予約だった場合は，フェンスを挟んでいても予約する。表1中の［　　］に入れる正しい答えを，解答群の中から選べ。

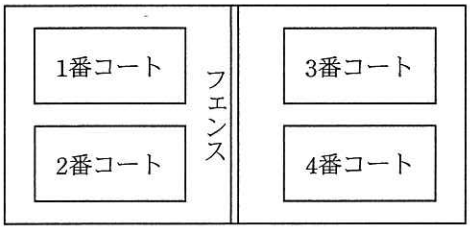


図3　テニスコートの配置

表1　プログラムの変更内容

処置	変更内容
行番号 25 と 26 の間に追加	02 PRM-NEXT　　PIC 9(1). 88 NEXT-OFF　VALUE 0. 88 NEXT-ON　　VALUE 1.
行番号 29 と 30 の間に追加	88 RSV-NOTE　VALUE 4.
行番号 74 と 75 の間に追加	ELSE IF [e] AND [f] AND [g] THEN SET RSV-NOTE TO TRUE INITIALIZE RET-COURT END-IF

e ～ g に関する解答群

ア　((COURT-NO(1) = 1 OR 2) AND (COURT-NO(2) = 3 OR 4))

イ　(COURT-CNT = 3 OR 4)

ウ　(COURT-NO(1) = 1 OR 3)

エ　NEXT-OFF

オ　NEXT-ON

カ　PRM-NUM = 2

キ　RSV-OK

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1～3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〕

ディレクトリパス（以下，パスという）から木構造を生成するプログラムである。

パスは，木構造をもつファイルシステムにおいてディレクトリを特定するために利用される文字列であり，ディレクトリの名前を“/”で区切って並べて表す。“/”で始まるパスを絶対パスという。絶対パスはルートディレクトリを起点として表したパスである。“/”で始まらないパスを相対パスという。相対パスは任意のノードを起点として表したパスである。

このプログラムが生成する木構造中の各ノードは，それぞれが一つのディレクトリを表し，ルートノードはルートディレクトリを表す。

図1に木構造の例を示す。図1中の楕円一つはノード一つに対応し，“ と ”で囲まれた文字列はノードの名前を表す。ルートノードの名前は空文字列とする。例えば，ノード usr を特定する絶対パスは“/usr”であり，ノード usr を起点とする相対パス“local/lib”が特定するノードは，絶対パス“/usr/local/lib”が特定するノードと同じノード lib である。

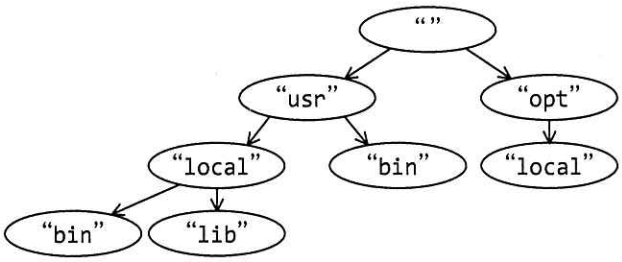


図 1　木構造の例

クラス DirectoryNode は木構造を構成するノードを表すクラスであり，一つのインスタンスが一つのノードを表す。フィールド name はノードの名前を，フィールド parent は親ノードへの参照を，フィールド children は子ノードのリストを保持する。引数を取らないコンストラクタはルートノードを生成する。

クラス DirectoryNode は次のメソッドをもつ。

(1) public DirectoryNode add(String path)

引数 path で与えられたパスが，このノードを起点に一つのノードを特定できるように木構造を拡張し，パスが特定するノードを返す。引数 path が空文字列又は絶対パスを表すなら，IllegalArgumentException を投げる。

パス中の連続する“/”は一つの“/”として扱い，末尾の“/”は無視する。つまり，パス“local//lib/”は“local/lib”とみなす。

(2) public String path()

このノードを特定する絶対パスを表す文字列（末尾は常に“/”）を返す。

例えば，図 1 の最下段のノード lib でこのメソッドを呼ぶと，“/usr/local/lib/”を返す。

(3) public List<DirectoryNode> find(String name)

このノードが保持する各子ノードを頂点とする全ての部分木から，引数 name で与えられた名前をもつノードを全て探し，見つかったノードをリストで返す。

クラス DirectoryNodeTester はテスト用のプログラムである。実行結果を図 2 に示す。

```
usr
opt
/usr/local/
/opt/local/
IllegalArgument
```

図 2　クラス DirectoryNodeTester の実行結果

〔プログラム 1〕

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class DirectoryNode {
   public final DirectoryNode parent;
   public final String name;
   public final List<DirectoryNode> children =
      new ArrayList<DirectoryNode>();

   public DirectoryNode() {
      this( a );
   }
```

```
private DirectoryNode(String name, DirectoryNode parent) {
    this.name = name;
    this.parent = parent;
}

public DirectoryNode add(String path) {
    if ( [ b ] ) {
        throw new IllegalArgumentException();
    }
    // 第1引数は文字列pathに含まれるディレクトリのリストである
    return add(Arrays.asList(path.split("/+")), 0);
}

private DirectoryNode add(List<String> path, int i) {
    DirectoryNode child = findChild(path.get(i));
    if (child == null) {
        children.add(child = new DirectoryNode(path.get(i), this));
    }
    if (path.size() > i + 1) {
        return child.add(path, i + 1);
    }
    return child;
}

public String path() {
    if ( [ c ] ) { // 自ノードがルートノードなら"/"を返す
        return "/";
    }
    return parent.path() + name + "/";
}

private DirectoryNode findChild(String name) {
    for (DirectoryNode child : children) {
        if (child.name.equals(name)) {
            return child;
        }
    }
    return null;
}

public List<DirectoryNode> find(String name) {
    List<DirectoryNode> ret = new ArrayList<DirectoryNode>();
    DirectoryNode node = findChild(name);
    if (node != null) {
        ret.add( [ d ] );
    }
    for (DirectoryNode child : children) {
        ret.addAll(child.find( [ e ] ));
    }
    return ret;
```

```
    }
}
```

〔プログラム2〕

```
public class DirectoryNodeTester {
    public static void main(String[] args) {
        try {
            DirectoryNode root = new DirectoryNode();
            DirectoryNode usr = root.add("usr");
            usr.add("bin");
            usr.add("local/bin");
            usr.add("local/lib");
            root.add("opt/local");
            for (DirectoryNode n : root.children) {
                System.out.println(n.name);
            }
            for (DirectoryNode n : root.find("local")) {
                System.out.println(n.path());
            }
            usr.add("");
        } catch (IllegalArgumentException e) {
            System.out.println("IllegalArgument");
        }
    }
}
```


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　""　　イ　"", null　　ウ　null　　エ　null, ""

bに関する解答群

ア　!path.startsWith("/") && !"".equals(path)

イ　!path.startsWith("/") || !"".equals(path)

ウ　path.startsWith("/") && "".equals(path)

エ　path.startsWith("/") || "".equals(path)

cに関する解答群

ア　parent != ""　　イ　parent != null

ウ　parent == ""　　エ　parent == null

d, eに関する解答群

ア child	イ name	ウ node
エ null	オ parent	カ root

設問2 クラス DirectoryNodeTester を実行したときに生成される，クラス DirectoryNode のインスタンスの個数として正しい答えを，解答群から選べ。

解答群

ア 6　　イ 7　　ウ 8　　エ 9　　オ 10

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。

クラス DirectoryNode のメソッド path が返す文字列の末尾は常に“/”である。これを，このメソッドが呼ばれたインスタンスがルートノードであるか，子ノードをもつときにだけ末尾が“/”になるように，メソッド path の最後の return 文の直前に次の3行を挿入した。クラス DirectoryNodeTester の実行結果は図3となる。

```
if ( [ f ] ) {
   return parent.path() + name;
}
```

```
usr
opt
/usr/local/
/opt/local
IllegalArgument
```

図3 クラス DirectoryNode 変更後のクラス DirectoryNodeTester の実行結果

fに関する解答群

ア !children.isEmpty()	イ children != null
ウ children == null	エ children.isEmpty()

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 1～3 に答えよ。

〔プログラム 1，2 の説明〕

図 1 のように，主プログラムから数字列として渡された 100 分未満の時間を数値の秒に変換する副プログラム TOSEC と，その逆変換を行う副プログラム TOTIME である。

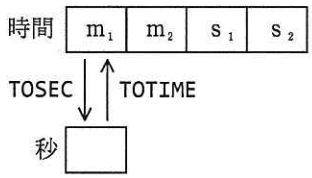


図 1　数字列の時間と数値の秒との変換

m_1m_2は分，s_1s_2は秒を表す。分は 0～99，秒は 0～59 の値とする。

(1) 副プログラム TOSEC は，主プログラムから数字列として渡された時間を，数値の秒に変換して主プログラムに返す。

時間が格納されている領域の先頭アドレスは GR1 に設定されて，主プログラムから渡される。秒は GR0 に設定する。

(2) 副プログラム TOTIME は，主プログラムから渡された数値の秒を，数字列の時間に変換して主プログラムに返す。

秒は GR0 に，結果の数字列の時間を格納する領域の先頭アドレスは GR1 に設定されて，主プログラムから渡される。

(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  TOSEC    START
 2           RPUSH
 3           LD      GR5,=0
 4           LD      GR2,=4          ; ループ回数
 5           LAD     GR3,VALUE1
 6  LOOP1    LD      GR0,0,GR1       ; GR0←時間の1桁
 7           AND     GR0,=#000F
 8           ADDA    GR0,GR5
 9           SUBA    GR2,=1
10           JZE     FIN
11           LD      GR4,0,GR3       ; GR4←掛ける値
12           LD      GR5,=0
13  LOOP2    SRL     GR4,1           ; GR0×GR4
14           JOV     INCR
15           [   a   ]
16           LAD     GR3,1,GR3
17           LAD     GR1,1,GR1
18           JUMP    LOOP1
19  INCR     ADDA    GR5,GR0
20  CONT     SLL     GR0,1
21           JUMP    LOOP2
22  FIN      RPOP
23           RET
24  VALUE1   DC      10,6,10
25           END
```

アセンブラ

〔プログラム 2〕

(行番号)

```
 1  TOTIME  START
 2          RPUSH
 3          LD      GR2,=3          ; ループ回数
 4          LAD     GR3,VALUE2
 5  LOOP3   LD      GR4,0,GR3       ; GR4←割る値
 6          LD      GR5,=0
 7  LOOP4   CPA     GR0,GR4         ; GR0÷GR4
 8          JMI     NEXT
 9          ADDA    GR5,=1
10          SUBA    GR0,GR4
11          JUMP    LOOP4
12  NEXT    OR      [ b ],='0'
13          ST      [ b ],0,GR1
14          LAD     GR3,1,GR3
15          LAD     GR1,1,GR1
16          SUBA    GR2,=1
17          JNZ     LOOP3
18          OR      [ c ],='0'
19          ST      [ c ],0,GR1
20          RPOP
21          RET
22  VALUE2  DC      600,60,10
23          END
```

アセンブラ

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　CONT　　イ　JNZ　CONT　　ウ　JUMP　CONT　　エ　JZE　CONT

b, c に関する解答群

ア　GR0　　イ　GR1　　ウ　GR2　　エ　GR3

オ　GR4　　カ　GR5

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。

プログラム1を1回呼び出したとき，行番号19のADDA命令は［　d　］回実行される。

プログラム2に渡された秒が10進数で274であるとき，行番号10のSUBA命令は［　e　］回実行される。

解答群

ア　3	イ　6	ウ　7	エ　10
オ　19	カ　272	キ　274	

設問3　1チーム4名の選手の10 km走のタイムの平均値を求めて出力する副プログラムAVERAGE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。

(1)　主プログラムから，図2に示すデータが格納されている領域の先頭アドレスがGR1に設定されて渡される。4名の選手のタイムは図1に示す数字列の時間の形式で格納されている。

(GR1)+0	選手1のタイムが格納されている領域の先頭アドレス
+1	選手2のタイムが格納されている領域の先頭アドレス
+2	選手3のタイムが格納されている領域の先頭アドレス
+3	選手4のタイムが格納されている領域の先頭アドレス

図2　主プログラムから渡される4名の選手のデータ

(2)　平均値は，図1に示す数字列の時間の形式で出力する。1秒未満は切り捨てる。

(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3〕

```
AVERAGE START
        RPUSH
        LD      GR2,=4
        LD      GR3,GR1
        LD      GR4,=0
LOOP5   LD      GR1,0,GR3
        CALL    TOSEC
        [ f ]
        LAD     GR3,1,GR3
        SUBA    GR2,=1
        JNZ     LOOP5
        [ g ]
        LD      GR0,GR4
        LAD     GR1,RESULT
        CALL    TOTIME
        OUT     RESULT,LEN4
        RPOP
        RET
LEN4    DC      4
RESULT  DS      4
        END
```

解答群

ア	ADDA	GR4,GR0	イ	LD	GR4,GR0	ウ	SUBA	GR4,GR0
エ	SLL	GR4,1	オ	SLL	GR4,2	カ	SRL	GR4,1
キ	SRL	GR4,2						

問 13　次の表計算のワークシート及びマクロの説明を読んで，設問 1，2 に答えよ。

〔表計算の説明〕

ある科目を受講する学生を，知人関係の情報に基づいてグループ分けするプログラムを表計算ソフトで作成した。グループ分けする対象の学生は 50 人である。グループはそれぞれ 10 人で，各グループにはあらかじめ 1 人のグループ長が決められている。

〔ワークシート：知人関係類似度行列〕

任意の 2 学生の知人関係の情報を表現した“知人関係類似度行列”を作成した。ワークシート“知人関係類似度行列”の例を図 1 に示す。

	A	B	C	D	E	F	…	AY	AZ
1		受講者番号	1	2	3	4	…	49	50
2	受講者番号	氏名	佐藤一郎	鈴木二郎	高橋三郎	田中四郎	…	伊藤花子	山本礼子
3	1	佐藤一郎	0	0.117	0.133	0.111	…	0.052	0.136
4	2	鈴木二郎	0.117	0	0	0.133	…	0.133	0.157
5	3	高橋三郎	0.133	0	0	0	…	0.153	0.111
6	4	田中四郎	0.111	0.133	0	0	…	0.125	0.095
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
51	49	伊藤花子	0.052	0.133	0.153	0.125	…	0	0.150
52	50	山本礼子	0.136	0.157	0.111	0.095	…	0.150	0

図 1　ワークシート“知人関係類似度行列”の例

(1)　各学生には 1 ～ 50 の受講者番号が振られている。セル A3 ～ A52 には，1 から順番に受講者番号を入力する。セル B3 ～ B52 には受講者番号に対応する氏名を入力する。同様にセル C1 ～ AZ1 には受講者番号を，セル C2 ～ AZ2 には氏名を入力する。

(2)　セル C3 ～ AZ52 には，対応する 2 人の知人関係の類似性をある指標に基づいて数値化した値（以下，類似度という）が格納されている。類似度は 0 以上 1 以下の数値である。各セルに対応する学生 2 人の組合せが同一であれば類似度は同じ値である。例えば，受講者番号 1 の佐藤一郎と受講者番号 2 の鈴木二郎の類似度に対応

するセル C4，D3 にはいずれも同じ値が格納される。また，同一学生同士に当たるセル C3，D4，…，AZ52 には 0 が入力されている。

〔ワークシート：グループ分け〕

ワークシート“知人関係類似度行列”を基に，図 2 に示すワークシート“グループ分け”を作成した。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		グループ番号	1	2	3	4	5			1	2	3	4	5			
2		グループ長	2	11	16	32	45	済		2	11	16	32	45			
3	1	佐藤一郎	0	0	0	0	1	1		−1	−1	−1	−1	−1	−1	16	1
4	2	鈴木二郎	1	0	0	0	0	1		−1	−1	−1	−1	−1	−1	16	1
5	3	高橋三郎	0	0	0	0	0	0		0	0.071	0.076	−1	0	0.076	13	3
6	4	田中四郎	0	0	0	0	0	0		0.133	0.125	0.133	−1	0.062	0.133	8	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
51	49	伊藤花子	0	0	0	1	0	1		−1	−1	−1	−1	−1	−1	16	1
52	50	山本礼子	0	0	0	1	0	1		−1	−1	−1	−1	−1	−1	16	1
53		定員充足	0	0	0	1	0										
54																	
55		受講者番号	18	の学生をグループ			1	に推薦する									

図 2　ワークシート“グループ分け”

(1)　セル A3 ～ A52 には，1 から順番に受講者番号を入力する。セル B3 ～ B52 には，受講者番号に対応する氏名を入力する。

(2)　セル C1 ～ G1 には，1 から順番にグループ番号を入力する。

(3)　セル C2 ～ G2 には，各グループのグループ長の受講者番号を入力する。

(4)　セル C3 ～ G52 には，それぞれの学生（グループ長を含む）が，対応するグループに割り当てられている場合は 1 を，そうでない場合には 0 を入力する。

(5)　セル H3 ～ H52 には，対応する学生が既にいずれかのグループに割り当てられている場合は 1 が，そうでない場合は 0 が表示される。

(6)　セル C53 ～ G53 には，対応するグループに割り当てられた学生が 10 人であれば 1 が，そうでない場合は 0 が表示される。

(7)　本ワークシートは，グループ分け作業を支援するための推薦機能を有している。

どのグループにも割り当てられていない学生と，所属する人数が 10 人に達していないグループのグループ長との組合せの中で，類似度が最大となる組合せを抽出し，その組合せに含まれる学生の受講者番号とグループ長が属するグループ番号を行 55 に表示する。ただし，類似度が最大となる組合せが複数ある場合，次の優先順で最も優先度が高い組合せを表示する。

① 割り当てようとする学生の受講者番号が最も小さい組合せ

② ①に該当する組合せが複数存在する場合，その中でグループ長が属するグループ番号が最も小さい組合せ

(8) 列 J から列 Q を，推薦機能のための計算領域に用いる。

設問 1 ワークシート“グループ分け”の推薦機能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。

(1) セル J3 ～ N52 には，それぞれの学生と対応するグループ長との類似度の値を入力する。ただし，学生が既にいずれかのグループに割り当てられている場合，又は対応するグループに割り当てられている学生数が 10 人である場合には－1 が入るようにしたい。そこで，次の式をセル J3 に入力し，セル J3 ～ N52 に複写する。

IF ([a] , －1, 表引き (知人関係類似度行列 !C3 ～ AZ52, [b]))

(2) それぞれの学生を割り当てる候補となるグループを選定するために，次の式をセル O3 に入力し，セル O4 ～ O52 に複写する。

最大(J3 ～ N3)

さらに，次の式をセル Q3 に入力し，セル Q4 ～ Q52 に複写する。

照合一致(O3, J3 ～ N3, 0)

(3) 推薦する受講者番号を表示するために，次の式をセル P3 に入力し，セル P4 ～ P52 に複写する。

順位(O3, O$3 ～ O$52, 1)

さらに，次の式をセル C55 に入力する。

[c]

(4) 推薦する学生の割当て先グループ番号を表示するために，次の式をセルG55に入力する。

d

aに関する解答群

ア 論理積(\$H3＝0, C\$53＝0)　　イ 論理積(\$H3＝0, C\$53＝1)

ウ 論理積(\$H3＝1, C\$53＝0)　　エ 論理積(\$H3＝1, C\$53＝1)

オ 論理和(\$H3＝0, C\$53＝0)　　カ 論理和(\$H3＝0, C\$53＝1)

キ 論理和(\$H3＝1, C\$53＝0)　　ク 論理和(\$H3＝1, C\$53＝1)

bに関する解答群

ア \$A3, C\$1　　イ \$A3, C\$2　　ウ A\$3, C\$1　　エ A\$3, C\$2

cに関する解答群

ア 照合一致(1, P3～P52, 0)　　イ 照合一致(50, P3～P52, 0)

ウ 照合検索(1, P3～P52, O3～O52)　　エ 照合検索(50, P3～P52, O3～O52)

オ 条件付個数(H3～H52, ＝1)　　カ 条件付個数(O3～O52, ＝－1)

キ 条件付個数(O3～O52, ≠－1)

dに関する解答群

ア 整数(条件付個数(H3～H52, ＝1)／10)＋1

イ 表引き(Q3～Q52, C55, 1)

ウ 照合一致(C55, P3～P52, 0)

エ 照合検索(C55, P3～P52, O3～O52)

オ 照合検索(C55, Q3～Q52, A3～A52)

カ 条件付個数(C53～G53, ＝1)＋1

表計算

設問2　ワークシート“グループ分け”のセル C2～G2 にグループ長の受講者番号を入力して実行すると以後のグループ分け処理を全て自動化するマクロ Grouping を，ワークシート“グループ分け”に格納した。マクロ Grouping は次の手順で処理を行う。

① セル C3～G52 を全て 0 で初期化する。

② 5 人のグループ長をそれぞれのグループに割り当てる。つまりセル C3～G52 の中で対応するセルの値を 1 に変更する。

③ 推薦機能によって，推薦する学生の受講者番号はセル C55 に，グループ番号はセル G55 に表示されるので，セル C3～G52 の中で対応するセルの値を 1 に変更する。

④ 全員がいずれかのグループに割り当てられるまで③を繰り返す。

マクロ Grouping 中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔マクロ：Grouping〕

```
○マクロ：Grouping
○数値型：I, J, NumUsers, NumGroups
• NumUsers ← 50
• NumGroups ← 5
■ I：1, I ≦ NumGroups, 1
  ■ J：1, J ≦ NumUsers, 1
    • 相対(B2, J, I) ← 0
  ■
  • [ e ]
■
■ I：1, I ≦ [ f ], 1
  • [ g ]
■
```

e, gに関する解答群

ア　相対(B2, C55, I) ← 1

イ　相対(B2, C55, G55) ← 1

ウ　相対(B2, I, G55) ← 1

エ　相対(B2, I, 相対(B2, 0, I)) ← 1

オ　相対(B2, I, 相対(B2, C55, I)) ← 1

カ　相対(B2, 相対(B2, 0, I), I) ← 1

キ　相対(B2, 相対(B2, C55, I), I) ← 1

ク　相対(B2, 相対(B2, 0, G55), I) ← 1

fに関する解答群

ア　NumGroups

イ　NumGroups * NumGroups

ウ　NumUsers

エ　NumUsers * (NumGroups − 1)

オ　NumUsers * NumGroups

カ　NumUsers + NumGroups

キ　NumUsers − NumGroups

ク　NumUsers ／ NumGroups

表計算

■ Java プログラムで使用する API の説明

java.util

public interface List<E>

リスト（順序付けられたコレクション）のためのインタフェースを提供する。インタフェース Collection を継承する。

メソッド

public boolean add(E e)
指定された要素をリストの最後に追加する。
引数： e — リストに追加する要素
戻り値：true

public boolean addAll(Collection<? extends E> c)
指定されたコレクション内の全ての要素を，指定されたコレクションの反復子によって返される順序でリストの最後に追加する。
戻り値：この呼出しの結果，このリストが変更されれば true
それ以外は false

public E get(int index)
リスト内の指定された位置にある要素を返す。
引数： index — 返される要素のインデックス
戻り値：リスト内の指定された位置にある要素

public boolean isEmpty()
リストに要素がなければ true を返す。
戻り値：リストに要素が一つもなければ true
それ以外は false

public int size()
リスト内の要素数を返す。
戻り値：リスト内の要素数

java.util

public class ArrayList<E>

インタフェース List の配列による実装である。
メソッドの説明は，インタフェース List の項を参照。

コンストラクタ

public ArrayList()
空のリストを作る。

java.util

public class Arrays

クラス Arrays は，配列を操作するクラスメソッドから成る。

メソッド

`public static <T> List<T> asList(T... a)`

指定された配列を基にする固定サイズのリストを返す。

引数： a — 型 T の可変個の要素の並び又は型 T の配列

戻り値：指定された要素のリスト

java.lang

public final class String

クラス String は，文字列を表す。

メソッド

`public boolean startsWith(String prefix)`

この文字列が指定された接頭辞で始まるかどうかを判定する。

引数： prefix — 接頭辞

戻り値：この文字列が指定された接頭辞で始まれば true
それ以外は false

`public String[] split(String regex)`

この文字列を指定された正規表現に一致する位置で分割する。

正規表現に一致する部分文字列は，分割後の文字列には含まれない。分割後の文字列の並びが空文字列で終わるときは，その空文字列は破棄する。

例えば，正規表現 "/+" は，一つ以上の連続する "/" の並びと一致するので，
"abc/def///gh/".split("/+") は，配列 {"abc", "def", "gh"} を返す。

引数： regex — 正規表現

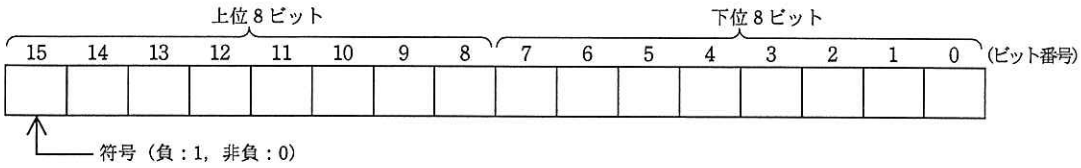
戻り値：正規表現に一致する位置で分割された文字列の配列
この文字列が正規表現に一致しないときはこの文字列だけを要素に持つ配列

■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（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 〕

〔 メ モ 用 紙 〕

〔 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。