FE

平成23年度　春期
基本情報技術者試験
午後　問題

特別試験

試験時間	13:00 ～ 15:30（2時間30分）

注意事項

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

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

6. 答案用紙の記入に当たっては，次の指示に従ってください。
   (1) 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**のとおりマークされていない場合は，読み取れず，採点されないことがありますので，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ご注意ください。
   (2) 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。
   (3) **受験番号欄**に，**受験番号**を記入及びマークしてください。正しくマークされていない場合は，採点されません。
   (4) **生年月日欄**に，受験票に印字されているとおりの**生年月日**を記入及びマークしてください。正しくマークされていない場合は，採点されないことがあります。
   (5) **選択した問題**については，右の例に従って，**選択欄**の**問題番号**の(選)をマークしてください。マークがない場合は，採点の対象になりません。問1～問7について，6問以上マークした場合は，はじめの5問を採点します。問9～問13について，2問以上マークした場合は，はじめの1問を採点します。

〔問1，問3，問4，問6，問7，問9を選択した場合の例〕

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	●				

   (6) 解答は，次の例題にならって，解答欄にマークしてください。

〔例題〕　次の［　　］に入れる正しい答えを，解答群の中から選べ。

秋の情報処理技術者試験は，［ a ］月に実施される。

解答群　ア　8　　イ　9　　ウ　10　　エ　11

正しい答えは“ウ　10”ですから，次のようにマークしてください。

例題	a	(ア) (イ) ● (エ)

裏表紙の注意事項も，必ず読んでください。

〔問題一覧〕

●問 1～問 7（7 問中 5 問選択）

問題番号	出題分野	テーマ
問 1	ハードウェア	機械語命令
問 2	ソフトウェア	CPU の割当て方式
問 3	データベース	トランザクション管理
問 4	ネットワーク	ルータの経路制御テーブルの更新
問 5	ソフトウェア設計	あて先作成プログラム
問 6	プロジェクトマネジメント	EVM によるプロジェクトの進捗管理
問 7	経営・関連法規	ゲーム理論を活用した出店戦略

●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	組合せ

●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）	劇場の空き座席の確認
問 10	ソフトウェア開発（COBOL）	株主優待処理と株の保有傾向分析
問 11	ソフトウェア開発（Java）	追加可能な文字列インタフェースの 2 種類の実装
問 12	ソフトウェア開発（アセンブラ）	図形の回転
問 13	ソフトウェア開発（表計算）	与信管理

共通に使用される擬似言語の記述形式

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

〔宣言，注釈及び処理〕

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

〔演算子と優先順位〕

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

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

〔論理型の定数〕

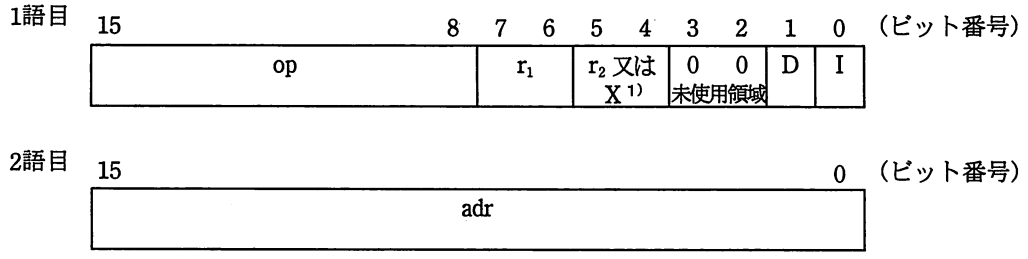
true, false

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　機械語命令に関する次の記述を読んで，設問1，2に答えよ。

この機械語が実行されるCPUの1語は16ビットで，CPUには1語長の汎用レジスタが四つ（レジスタ番号0～3）ある。主記憶容量は1,000語（番地0～999）あり，命令語は700番地以降に格納される。命令語の形式は図1に示すとおりである。命令には1語命令と2語命令があり，1語命令の場合は1語目だけで構成される。



注1) 1語命令のときにはr_2，2語命令のときにはX

図1　命令語の形式

図1で使用している記号の説明を表1に，命令の実効アドレスの算出方法を表2に，命令の対象となるデータが設定されているレジスタ（以下，ソースレジスタという）の指定方法を表3に，命令の仕様（一部）を表4に示す。数字の末尾にhが付いているものは16進数表記である。

表1　記号の説明

記号	ビット数	内容
op	8	00h～FFhで示されるいずれかの命令コードが指定される。
r_1	2	0～3で示されるいずれかのレジスタ番号が指定される。
r_2	2	0～3で示されるいずれかのレジスタ番号が指定される。
X	2	指標レジスタ修飾を行うときは，指標レジスタ修飾に使用するレジスタを表す1～3で示されるいずれかのレジスタ番号が指定される。 指標レジスタ修飾を行わないときは，0が指定される。
D	1	1語命令のときは，1が指定される。 2語命令のときは，0が指定される。
I	1	間接アドレス指定を行うときは，1が指定される。 間接アドレス指定を行わないときは，0が指定される。
adr	16	0～999で示されるいずれかの値（番地を表す値）が指定される。

表2　実効アドレスの算出方法

D	I	実効アドレス
0	0	adr+[Xで指定されたレジスタ]
0	1	[adr+[Xで指定されたレジスタ]]
1	1	[r_2で指定されたレジスタ]

注　[ ]は，[ ]内のレジスタ又は番地に格納されている内容を示す。

表3　ソースレジスタの指定方法

D	I	ソースレジスタ
1	0	r_2で指定されたレジスタ

表4　命令の仕様（一部）

命令コード	動作
10h	実効アドレスに格納されている内容又はソースレジスタの内容を，r_1で指定されたレジスタに足し込む。
20h	実効アドレスに格納されている内容又はソースレジスタの内容を，r_1で指定されたレジスタに設定する。
30h	r_1で指定されたレジスタの内容を，実効アドレスに格納する。
FFh	プログラムを終了する。

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。

命令を実行する前のレジスタの内容は，図2のとおりとする。この状態で図3のプログラムを先頭の700番地から順に実行した。

命令語2の実行が終わった時点での，レジスタ番号1のレジスタの内容は［ a ］である。命令語5の実行が終わった時点での，レジスタ番号1のレジスタの内容は［ b ］であり，レジスタ番号3のレジスタの内容は［ c ］である。

レジスタ番号	内容
0	100
1	200
2	300
3	400

図2　レジスタの内容

主記憶番地	内容	
700	2042h	命令語1
701	10D2h	命令語2
702	3090h	命令語3
703	0000h	
704	1013h	命令語4
705	2042h	命令語5
706	FF02h	命令語6

図3　プログラム

解答群

ア　100　　イ　200　　ウ　300　　エ　400

オ　500　　カ　600

設問2　レジスタと主記憶の内容が図4に示す値のとき，レジスタ番号1のレジスタに100を設定する命令語の記述として**誤りであるもの**を，解答群の中から選べ。

レジスタ番号	内容
2	100
3	101

主記憶番地	内容
100	101
101	100

図4　レジスタと主記憶の内容

解答群

	1語目					2語目
	op	r_1	r_2 又は X	D	I	adr
ア	20h	1	2	0	1	0000h
イ	20h	1	2	1	0	
ウ	20h	1	3	0	0	0000h
エ	20h	1	3	0	1	0000h
オ	20h	1	3	1	1	

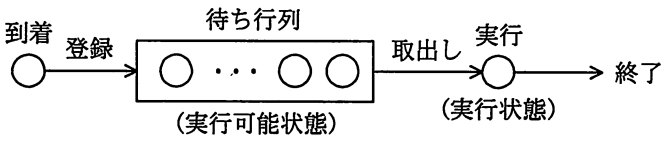
問2　CPUの割当て方式に関する次の記述を読んで，設問1，2に答えよ。

オペレーティングシステムの役割の一つとして，プロセスにCPUを割り当てることがある。そして，プロセスの実行順序を決定する方式には，次のようなものがある。

(1)　到着順方式

到着順にプロセスを待ち行列の末尾に登録する。実行中のプロセスが終了すると，待ち行列の先頭からプロセスを一つ取り出して実行を開始する。

到着順方式を図1に示す。待ち行列に登録されているプロセスの状態を実行可能状態，実行中のプロセスの状態を実行状態と呼ぶ。



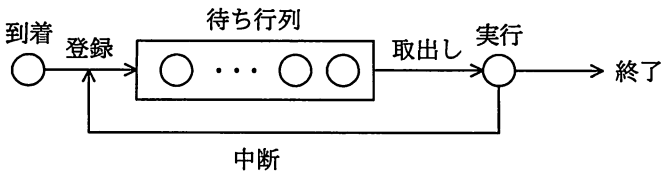
注　○はプロセスを表す。

図1　到着順方式

(2)　ラウンドロビン方式

到着順にプロセスを待ち行列の末尾に登録する。実行中のプロセスが終了すると，待ち行列の先頭からプロセスを一つ取り出して実行を開始する。また，実行中のプロセスが一定時間（以下，タイムクウォンタムという）を経過したら，実行を中断して，待ち行列の末尾に再登録し，待ち行列の先頭からプロセスを一つ取り出して実行を開始する。

ラウンドロビン方式を図2に示す。



注　○はプロセスを表す。

図2　ラウンドロビン方式

これらの方式の効率を示す指標としてターンアラウンドタイムがある。ここで，ターンアラウンドタイムとは，プロセスが待ち行列に到着してから実行が終了するまでの時間であり，プロセスの実行順序に影響される。

なお，このコンピュータシステムの CPU は一つであり，CPU は同時に一つのプロセスしか実行できない。

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。

四つのプロセス A～D があり，各プロセスの到着時刻と処理時間を表 1 に示す。表 1 において，到着時刻とは，プロセス A が待ち行列に到着した時刻を 0 としたときの各プロセスが到着する時刻であり，処理時間とは，各プロセスの処理が完了するために必要な CPU の処理時間である。

表 1　プロセスの到着時刻と処理時間

プロセス	到着時刻（ミリ秒）	処理時間（ミリ秒）
A	0	180
B	10	80
C	30	40
D	50	20

このとき，到着順方式におけるターンアラウンドタイムの平均は［ a ］ミリ秒である。そして，タイムクウォンタムが 20 ミリ秒のとき，ラウンドロビン方式におけるターンアラウンドタイムの平均は［ b ］ミリ秒である。ここで，プロセス A が到着したとき，実行可能状態及び実行状態のプロセスはないものとする。

なお，プロセスの登録と取出し，及び中断の処理でのオーバヘッドは考えない。また，CPU を割り当てられたプロセスは，タイムクウォンタム以外で中断することはない。

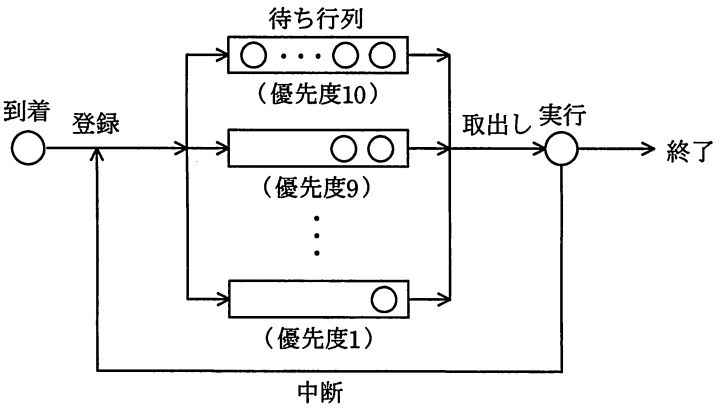
解答群

ア　80.0　　イ　102.5　　ウ　182.5

エ　192.5　　オ　242.5

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。

プロセスの実行順序を決める別の方式に優先度順方式がある。優先度順方式の例を図3に示す。プロセスにはあらかじめ優先度が付けてあり，待ち行列は優先度ごとに用意してある。ここで，優先度は1～10の10種類で，値の大きい方が優先度は高い。



注　○はプロセスを表す。

図3　優先度順方式の例

この方式では，次のとおりにプロセスの実行を制御する。

① プロセスを優先度に対応した待ち行列の末尾に登録する。

② プロセスが登録されている優先度の最も高い待ち行列の先頭からプロセスを一つ取り出して実行を開始する。

③ 実行中のプロセスの優先度が2以上のとき，実行時間が20ミリ秒経過するごとに優先度を一つ下げる。優先度を下げた結果，実行中のプロセスの優先度が実行可能状態にある優先度の最も高いプロセスよりも低くなった場合，実行中のプロセスを中断して，①に戻る。

④ 実行中のプロセスが終了した場合，②に戻る。

優先度順方式において，あるプロセスが終了した時点で表2に示す三つのプロセスだけが優先度に対応した待ち行列に登録されていたとする。このとき，三つのプロセスが終了する順番は［　c　］である。そして，プロセス

B の実行が終了したときのプロセス B の優先度は［ d ］である。ここで，三つのプロセスが終了するまで新たに到着するプロセスはないものとする。

なお，プロセスの登録と取出し，及び中断の処理でのオーバヘッドは考えない。また，CPU を割り当てられたプロセスは，タイムクウォンタム以外で中断することはない。

表2　プロセスの処理時間と優先度の初期状態

プロセス	処理時間（ミリ秒）	優先度
A	60	6
B	70	8
C	100	5

cに関する解答群

ア　A, B, C　　イ　A, C, B　　ウ　B, A, C　　エ　B, C, A

オ　C, A, B　　カ　C, B, A

dに関する解答群

ア　1　　イ　2　　ウ　3　　エ　4

オ　5　　カ　6

問3　データベースのトランザクション管理に関する次の記述を読んで，設問 1～4 に答えよ。

個人向けに，画材をインターネット販売する会社が運営する Web サイトがある。

この Web サイトが在庫管理に利用しているデータベースでは，絵の具の在庫数は色別に個々のデータとして管理されており，処理に応じて次の 3 種類のトランザクションが生成される。

① 1回の商品注文に対して，一つの出荷トランザクションが生成される。

② 1回の商品入荷に対して，一つの入荷トランザクションが生成される。

③ 1回の在庫照会に対して，一つの照会トランザクションが生成される。

なお，一つのトランザクションで，複数の色の絵の具を処理することができる。

設問1　ACID 特性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。

ACID 特性とは，データベースの一貫性を保証するために必要な特性で，原子性，一貫性，独立性，耐久性の四つがある。このうち，一貫性や独立性を保証するためにトランザクション管理では排他制御が必要となる。例えば，白絵の具の在庫数が 50 だった場合，表 1 に示すトランザクション T1 と T2 が同時に実行されたとき，排他制御を行わないと実行後の在庫数は 55 とならず，在庫数が［ a ］又は［ b ］となってしまう可能性がある。

なお，各トランザクションは，図 1 の①～③の順で在庫数データを処理する。

表 1　トランザクション T1，T2 の処理内容

トランザクション	処理内容
出荷トランザクション T1	白絵の具 5 本の出荷
入荷トランザクション T2	白絵の具 10 本の入荷

トランザクションT1の処理順序	トランザクションT2の処理順序
①白絵の具の在庫数データを読み込む。 ②白絵の具の在庫数データ ＝ 白絵の具の在庫数データ－5 ③白絵の具の在庫数データを書き込む。	①白絵の具の在庫数データを読み込む。 ②白絵の具の在庫数データ ＝ 白絵の具の在庫数データ＋10 ③白絵の具の在庫数データを書き込む。

図1　トランザクションT1，T2の処理順序

解答群

ア　40　　イ　45　　ウ　50　　エ　60　　オ　65

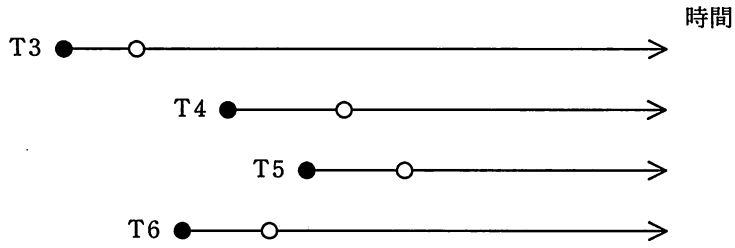
設問2　入荷トランザクション及び出荷トランザクションを処理する場合は対象データを占有ロックし，照会トランザクションを処理する場合は共有ロックする。

なお，このデータベースを管理するDBMSでは，あるトランザクションが共有ロックしているデータを，ほかのトランザクションからロックする場合，共有ロックの要求は成功するが，占有ロックの要求は共有ロックが解除されるまで待ち状態となる。

表2に示すトランザクションT3～T6を，図2に示すとおりに実行し，ロックを要求した場合，それぞれのトランザクションの状態について正しい説明を，解答群の中から選べ。

表2　トランザクションT3～T6の処理内容

トランザクション	処理内容
照会トランザクションT3	白絵の具の在庫数照会
入荷トランザクションT4	白絵の具10本の入荷
出荷トランザクションT5	白絵の具5本の出荷
照会トランザクションT6	白絵の具の在庫数照会



注 “●→”はトランザクションの実行開始と時間経過を，
“○”は白絵の具の在庫数データのロック要求を表す。

図2　トランザクションT3～T6の実行例

解答群

ア　T4，T5，T6とも待ち状態となる。

イ　T4，T5，T6とも待ち状態とならない。

ウ　T4，T5は待ち状態となるが，T6は待ち状態とならない。

エ　T4は待ち状態となるが，T5，T6は待ち状態とならない。

オ　T6は待ち状態となるが，T4，T5は待ち状態とならない。

設問3　出荷トランザクションT7の処理内容を表3に示す。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。

なお，トランザクションT7は，図3の①～⑧の順で在庫数データを処理する。

表3　トランザクションT7の処理内容

トランザクション	処理内容
出荷トランザクションT7	白絵の具5本と赤絵の具3本の出荷

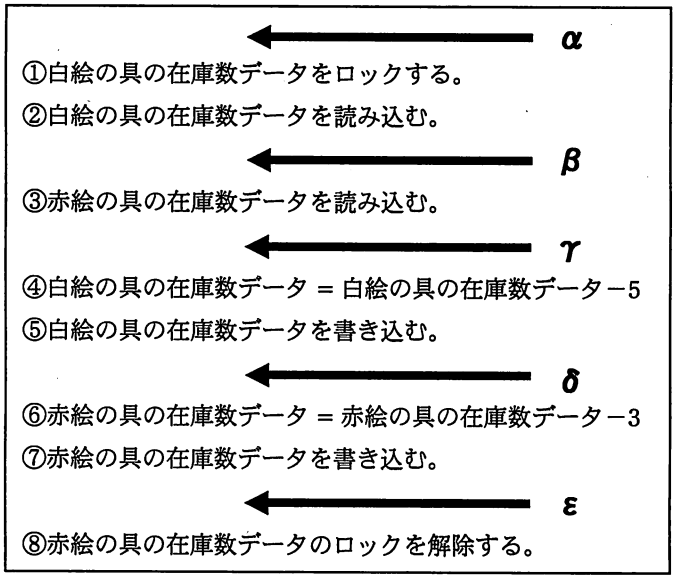


図3　トランザクションT7の処理順序

データをロックしている時間を最も短くするためには，“赤絵の具の在庫数データをロックする。”を挿入すべき適切な位置は［ c ］で，“白絵の具の在庫数データのロックを解除する。”を挿入すべき適切な位置は［ d ］である。

なお，ほかのトランザクションとのデッドロックの発生に対する考慮は不要とする。

解答群

ア　α　　イ　β　　ウ　γ　　エ　δ　　オ　ε

設問4　表4に示すトランザクションT8～T11のうち，解答群の組合せの中から，同時に処理された場合にデッドロックが発生する可能性のある組合せを選べ。

なお，トランザクションT8～T11では，各絵の具の在庫数データをどのような順番で処理するかは，分からないものとする。

表４　トランザクションT8～T11の処理内容

トランザクション	処理内容
出荷トランザクションT8	白絵の具５本と赤絵の具５本の出荷
入荷トランザクションT9	赤絵の具10本と青絵の具10本の入荷
出荷トランザクションT10	青絵の具２本と白絵の具５本の出荷
入荷トランザクションT11	青絵の具10本と黒絵の具10本の入荷

解答群

ア　T8, T9　　イ　T8, T9, T10　　ウ　T9, T10, T11

エ　T10, T11

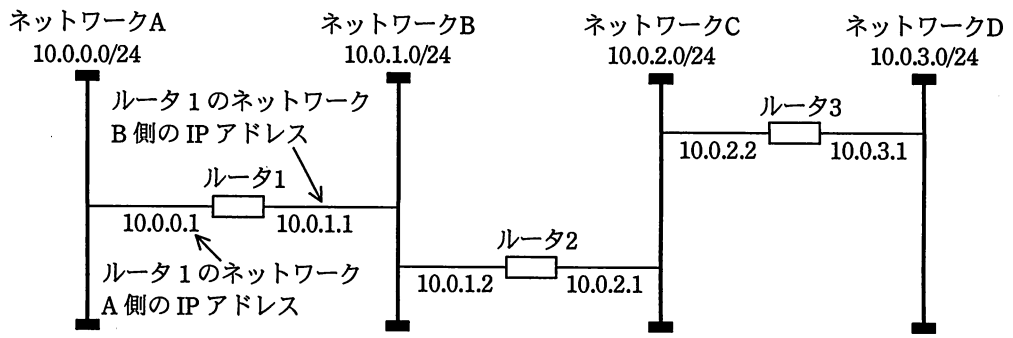
E
後

問4　ルータの経路制御テーブルの更新に関する次の記述を読んで，設問に答えよ。

ルータは，二つ以上の異なるネットワークをまたいだ通信における通信経路の選択を，ルータ内の経路制御情報を格納したテーブル（以下，テーブルという）に基づいて行う。同一のネットワークに接続された端末（ルータを含む）は同じネットワークアドレスをもつ。あるネットワークでブロードキャストしたパケットは異なるネットワークには転送されない。

テーブルの各レコードは送信先ネットワークのための経路制御情報を表し，送信先ネットワークアドレス（主キー），転送先ルータの IP アドレス及び距離で構成される。送信先ネットワークアドレスは受信したパケットのあて先のネットワークアドレスであり，転送先ルータの IP アドレスはそのパケットを転送すべきルータの IP アドレスである。あるネットワークあてのパケットは，該当する転送先ルータに転送すればよいことを表す。距離は，そのルータから転送されたパケットが，送信先のネットワークに到達するまでに経由する，ルータの個数である。

ネットワーク構成の例を図1に，図1中のルータ1のテーブルの例を図2に示す。



注 10.0.0.0/24は，ネットワークアドレス 10.0.0.0 とサブネットマスク 255.255.255.0 を表す。

図1　ネットワーク構成の例

送信先ネットワークアドレス	転送先ルータのIPアドレス	距離	
10.0.0.0/24	−	0	← 転送先ルータのIPアドレスの“−”は，送信先ネットワークとこのテーブルをもつルータが，直接つながっていることを表す。
10.0.1.0/24	−	0	
10.0.2.0/24	10.0.1.2	1	← ルータ1がネットワークCに接続された端末あてのパケットを受信したとき，そのパケットはIPアドレス10.0.1.2のルータ（ルータ2）に転送すればよいことを表す。
10.0.3.0/24	10.0.1.2	2	

図2　ルータ1のテーブルの例

ルータは，同一のネットワークに接続しているほかのルータが，ブロードキャストで定期的に送信する経路制御情報を受信することによって，自身のテーブルを更新する。これによって，異なるネットワーク上の端末にパケットを送信するときに，目的の端末にパケットが到達するまでに経由するルータの個数が最小になるよう，テーブルを動的に構成することができる。テーブルの更新手順を次に示す。

〔テーブルの更新手順〕

(1)　ルータは，そのルータに直接接続されたすべてのネットワークに，ブロードキャストで，保持するすべての経路制御情報を送信する。最初の送信は起動直後に行い，以後30秒間隔で送信する。

(2)　起動中のルータは，ほかのルータから送信された経路制御情報を受信し，自身のテーブルを次のとおりに更新する。

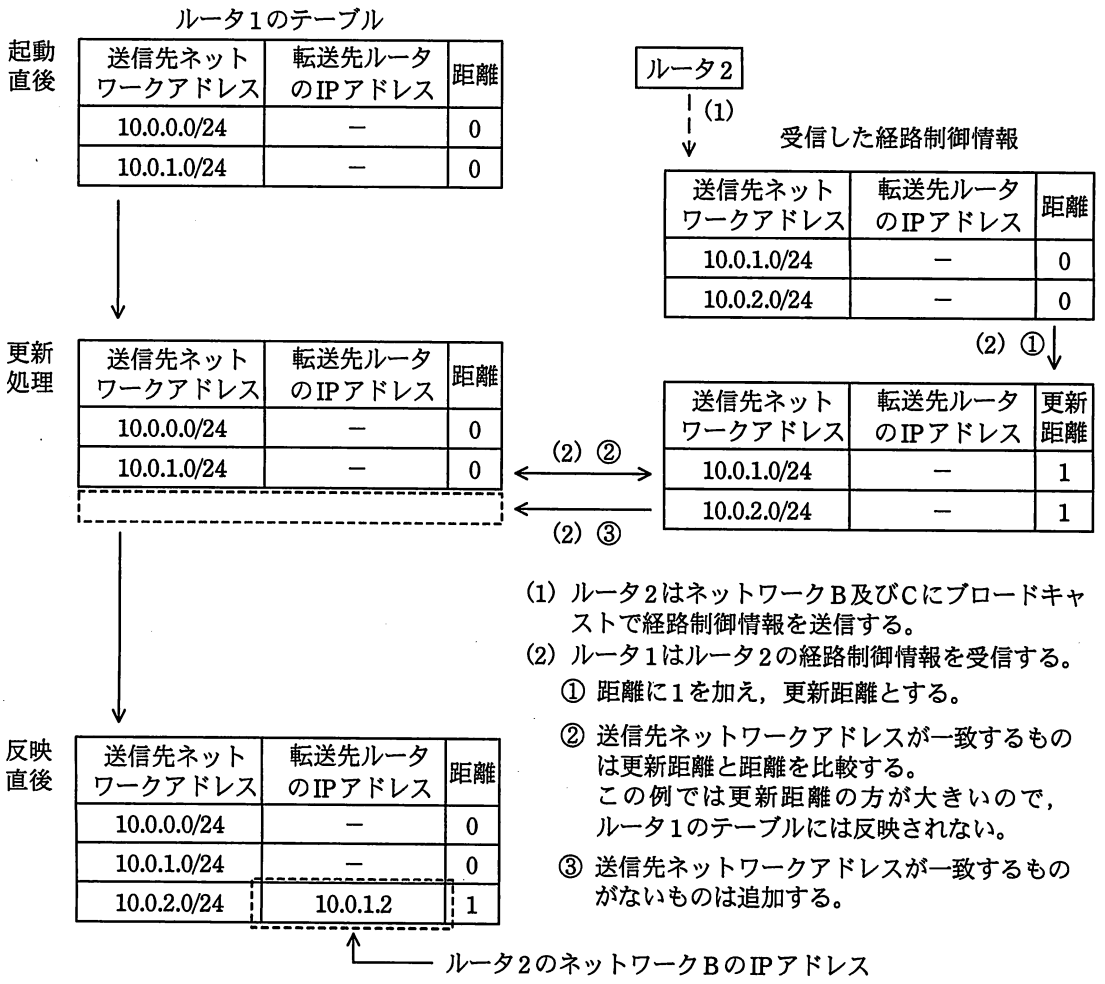
①　受信した経路制御情報のそれぞれの距離に1を加え，その距離を更新距離とする。

②　受信した経路制御情報のうち，送信先ネットワークアドレスが一致するレコードが自身のテーブルにあるものは，更新距離と該当するレコードの距離を比較し，更新距離の方が小さい場合は，距離を更新距離の値に，転送先ルータのIP アドレスを受信した経路制御情報の送信元ルータの IP アドレスに更新する。

③　受信した経路制御情報のうち，送信先ネットワークアドレスが一致するレコードが自身のテーブルにないものは，自身のテーブルに追加する。ただし，距離は更新距離の値とし，転送先ルータの IP アドレスは受信した経路制御情報の送信元ルータのIPアドレスとする。

設問　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

図１のネットワーク構成の例において，ルータ 1，2，3 を順に，5 秒間隔で起動した。各ルータの起動直後のテーブルには，それぞれのルータが直接接続されたネットワーク（ルータ 1 ではネットワーク A とネットワーク B）の経路制御情報だけがあるとすると，ルータ 1 にネットワーク C のための経路制御情報が反映されるのは，図３に示すように，ルータ１を起動した５秒後（ルータ２が起動直後に送信したテーブルを受信したとき）であり，ネットワーク D のための経路制御情報が反映されるのは，ルータ１の起動から［ a ］秒後である。



注 (1)，(2) ①～③は〔テーブルの更新手順〕の (1)，(2) ①～③を表す。

図３　ネットワーク C のための経路制御情報のルータ１への反映

ルータ 2 にネットワーク A のための経路制御情報が反映されるのは，ルータ 1 の起動から［ b ］秒後であり，そのときのルータ 2 のテーブルは，表 1 となる。

表 1　ルータ 2 のテーブル

c

また，ルータ 1 の起動から 20 秒後には，ルータ 3 のテーブルに，［ d ］のための経路制御情報が保持され，40 秒後には，［ e ］のための経路制御情報が保持されている。

a，b に関する解答群

ア　5　　イ　10　　ウ　30　　エ　35　　オ　40

c に関する解答群

ア

送信先ネットワークアドレス	転送先ルータのIPアドレス	距離
10.0.0.0/24	10.0.1.1	1
10.0.1.0/24	−	0
10.0.2.0/24	−	0

イ

送信先ネットワークアドレス	転送先ルータのIPアドレス	距離
10.0.0.0/24	10.0.1.1	1
10.0.1.0/24	−	0
10.0.2.0/24	−	0
10.0.3.0/24	10.0.2.2	1

ウ

送信先ネットワークアドレス	転送先ルータのIPアドレス	距離
10.0.0.0/24	10.0.2.2	1
10.0.1.0/24	−	0
10.0.2.0/24	−	0
10.0.3.0/24	10.0.1.2	1

エ

送信先ネットワークアドレス	転送先ルータのIPアドレス	距離
10.0.0.0/24	10.0.2.2	2
10.0.1.0/24	−	0
10.0.2.0/24	−	0
10.0.3.0/24	10.0.2.2	1

d，e に関する解答群

ア　ネットワーク A，B，C 及び D　　イ　ネットワーク A，B 及び C

ウ　ネットワーク A，B 及び D　　エ　ネットワーク A，C 及び D

オ　ネットワーク B，C 及び D　　カ　ネットワーク C 及び D

問5　あて先作成プログラムに関する次の記述を読んで，設問1～3に答えよ。

通信販売会社の Z 社では，顧客に対して顧客番号を発行し，顧客マスタファイルで管理している。

このたび，2011 年 5 月 10 日から 6 月 20 日までの販売促進キャンペーン期間中（以下，期間中という）の顧客の購入状況に応じて，懸賞応募券（以下，応募券という）と催物招待券（以下，招待券という）を郵送することになった。そこで，売上伝票ファイルから応募券と招待券を送る顧客を選び，あて先ファイルを出力するあて先作成プログラムを作成することにした。

このプログラムに必要な機能は，次のとおりである。

(1)　顧客ごとの応募券の枚数は，この販売促進キャンペーンの対象商品である商品コード A001～A199 の商品を期間中に購入した個数と同数とする。

(2)　顧客ごとの招待券の枚数は，期間中に購入したすべての商品の購入金額の合計が 5 万円以上の顧客に対して 1 枚とする。

(3)　応募券又は招待券を送る顧客ごとに，あて先ファイルに 1 件のレコードを作成する。応募券と招待券の両方を送る場合でも 1 顧客に対して 1 件のレコードを作成する。

(4)　応募券の総枚数，招待券の総枚数及びあて先ファイルのレコードの件数を，合計表に印字する。

顧客マスタファイル，売上伝票ファイル，あて先ファイルは順ファイルである。これらのレコード様式を，図 1 に示す。

顧客マスタファイル

顧客番号	顧客住所	顧客氏名

売上伝票ファイル

購入日付	顧客番号	商品コード	購入個数	購入金額

あて先ファイル

顧客番号	顧客住所	顧客氏名	応募券枚数	招待券枚数	購入金額合計

図 1　各ファイルのレコード様式

顧客マスタファイルは，顧客番号の昇順に整列されている。売上伝票ファイルを，顧客番号の昇順に整列した作業ファイルを作り，このプログラムに入力する。

このプログラムの入出力関連を図 2 に，プログラムの流れを図 3 に，主なモジュールの処理内容を表 1 に示す。また，このプログラムのテストに用いる作業ファイルのテストデータを表 2 に，このテストデータを用いた場合の合計表の出力結果を図 4 に示す。

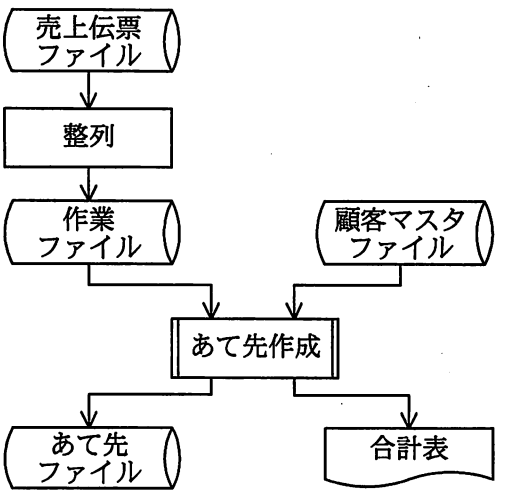
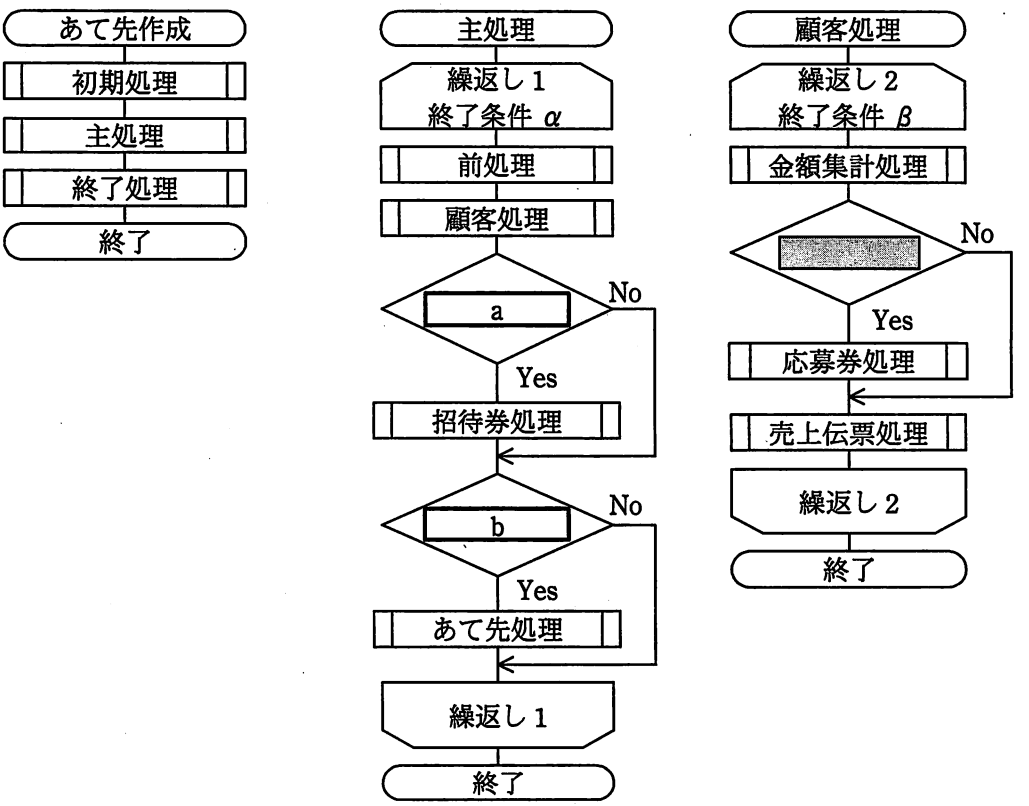


図2　あて先作成プログラムの入出力関連



注　網掛けの部分は，表示していない。

図3　あて先作成プログラムの流れ

表1　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。 応募券総枚数←0，招待券総枚数←0，あて先ファイルレコード件数←0 作業ファイルを読む（期間中以外のレコードは，読み飛ばす）。
前処理	応募券枚数←0，招待券枚数←0，購入金額合計←0， W顧客番号←作業ファイルレコードの顧客番号
金額集計処理	購入金額合計を求める。
応募券処理	応募券枚数← [　　]
売上伝票処理	作業ファイルを読む（期間中以外のレコードは，読み飛ばす）。
招待券処理	招待券枚数を求めるための計算を行う。
あて先処理	顧客マスタファイルを読む（顧客番号の値がW顧客番号の値と異なる場合は，読み飛ばす）。 あて先ファイルのレコードを編集し，あて先ファイルに出力する。 応募券総枚数を求めるための計算を行う。 招待券総枚数を求めるための計算を行う。 あて先ファイルレコード件数を求めるための計算を行う。
終了処理	合計表を出力する。 各ファイルを閉じる。

注　応募券枚数，招待券枚数，購入金額合計，W顧客番号，応募券総枚数，
招待券総枚数及びあて先ファイルレコード件数は作業領域である。

表2　作業ファイルのテストデータ

購入日付			顧客番号	商品コード	購入個数	購入金額(円)
年	月	日				
2011	5	8	001	A008	3	60000
2011	5	25	001	A180	8	44000
2011	6	3	002	B150	5	25000
2011	6	18	002	B125	5	20000
2011	5	25	005	A007	1	50000
2011	4	29	010	B001	1	45000
2011	5	25	010	B150	12	60000
2011	6	3	081	A201	2	52000
2011	6	12	081	B002	2	60000
2011	6	20	081	A006	2	20000
2011	7	16	081	A200	5	48000
2011	5	10	258	B003	1	2500
2011	5	18	386	A182	1	25000
2011	6	18	386	A198	2	20000
2011	6	21	386	A108	1	50000

合計表

懸賞応募券総枚数	
催物招待券総枚数	c
あて先ファイルレコード件数	d

注　網掛けの部分は，表示していない。

図4　表2のテストデータを用いた場合の合計表の出力結果

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に関する解答群

ア　A001 ≦ 作業ファイルレコードの商品コード ≦ A199

イ　A001 ≦ 作業ファイルレコードの商品コード ≦ A199　又は
　　購入金額合計 ≧ 50000

ウ　購入金額合計 ≧ 50000

エ　作業ファイルレコードの購入個数 ＞ 0

オ　招待券枚数 ＞ 0

bに関する解答群

ア　A001 ≦ 作業ファイルレコードの商品コード ≦ A199

イ　A001 ≦ 作業ファイルレコードの商品コード ≦ A199　かつ
　　作業ファイルレコードの購入金額 ≧ 50000

ウ　A001 ≦ 作業ファイルレコードの商品コード ≦ A199　又は
　　作業ファイルレコードの購入金額 ≧ 50000

エ　応募券枚数 ＞ 0

オ　応募券枚数 ＞ 0　かつ　招待券枚数 ＞ 0

カ　応募券枚数 ＞ 0　又は　招待券枚数 ＞ 0

キ　作業ファイルレコードの購入金額 ≧ 50000

ク　作業ファイルレコードの購入個数 ＞ 0

E
後

設問2　表1中の［　　］に入れる正しい答えを，解答群の中から選べ。解答群中の［］はガウス記号であり，[x]は，xを超えない最大の整数値を表す。

解答群

ア　［作業ファイルレコードの購入金額／50000］

イ　1

ウ　応募券枚数＋［作業ファイルレコードの購入金額／50000］

エ　応募券枚数＋1

オ　応募券枚数＋作業ファイルレコードの購入個数

カ　作業ファイルレコードの購入個数

設問3　図4中の［　　］に入れる正しい答えを，解答群の中から選べ。

cに関する解答群

ア　1	イ　2	ウ　3	エ　4	オ　5
カ　6	キ　7	ク　8	ケ　9	コ　10

dに関する解答群

ア　3	イ　4	ウ　5	エ　7	オ　9
カ　10	キ　11	ク　12	ケ　13	コ　14

問6　EVMによるプロジェクトの進捗管理に関する次の記述を読んで，設問1～3に答えよ。

ソフトウェア開発会社の D 社では，Web アプリケーション開発プロジェクト（以下，プロジェクト P という）の進捗管理に EVM（Earned Value Management）を活用することにした。

〔EVM についての説明〕

(1) EVM では，出来高計画値 PV（Planned Value），コスト実績値 AC（Actual Cost）及び出来高実績値 EV（Earned Value）といった三つの指標を用いて，プロジェクトのコスト及びスケジュールを管理する。

(2) PV は，計画時にプロジェクトの各工程での作業に割り当てられたコストの計画値であり，AC は，各工程での作業実行後のコストの実績値である。EV は，各工程の実行過程での進捗度をコストに換算した実績値であり，その時点での計画作業の完成率に PV を乗じた値である。

(3) EV と AC，EV と PV それぞれの差をとることで，プロジェクトのある時点での計画値と実績値との差異を把握できる。EV と AC との差（EV－AC）をコスト差異 CV（Cost Variance）といい，EV と PV との差（EV－PV）をスケジュール差異 SV（Schedule Variance）という。

(4) プロジェクトのある時点での計画値と実績値との差異を測る別の指標として，コスト効率指数 CPI（Cost Performance Index）とスケジュール効率指数 SPI（Schedule Performance Index）の二つがあり，次の式で求められる。

$$CPI = EV / AC$$

$$SPI = EV / PV$$

(5) “CV が 0，すなわち CPI が 1 の場合は，計画どおりのコストでプロジェクトが進捗している。”，“CV が正，すなわち CPI が 1 を超える場合は，計画よりも少ないコストで進捗している。”，そして，“CV が負，すなわち CPI が 1 未満の場合にはコスト超過である。”と判断できる。

同様に，“SV が 0，すなわち SPI が 1 の場合は，プロジェクトが計画どおりのスケジュールで進捗している。”，“SV が正，すなわち SPI が 1 を超える場合は，計画よりもスケジュールが早まっている。”，そして，“SV が負，すなわち SPI が 1 未満の場合は，スケジュール遅延である。”と判断できる。

〔プロジェクトPの説明〕

(1) プロジェクトPでは40個の機能の開発が必要であり，その開発スケジュール及びコスト計画は，表1のとおりである。

なお，ここではコストを表す単位として工数を使用する。

表1 プロジェクトPの開発スケジュール及びコスト計画

工程	標準工数	1月		2月		3月		4月		5月	
		機能数	計画工数	機能数	計画工数	機能数	計画工数	機能数	計画工数	機能数	計画工数
外部設計	40	25	1,000	10	400	5	200				
内部設計	40			25	1,000	15	600				
実装	30					25	750	15	450		
テスト	30							20	600	20	600
合計工数			1,000		1,400		1,550		1,050		600
累積工数			1,000		2,400		3,950		5,000		5,600

(2) 表1中の機能数とは，各月に作業を予定している機能の個数である。

なお，各機能はそれぞれ独立している。

(3) 表1中の標準工数とは，開発するアプリケーションの1機能当たりに予定される工数である。計画工数は，標準工数×機能数で算出する。

(4) プロジェクトPの1～3月の開発実績は，表2のとおりであった。

表2 1～3月の開発実績

工程	完了機能数		
	1月	2月	3月
外部設計	25	5	10
内部設計		25	5
実装			25

2月に計画していた外部設計10機能のうち，5機能は計画どおりに2月に完了した。しかし，残り5機能については，2月途中に要件見直しの要請があり，外部設計が計画よりも遅れ，3月末に完了した。

設問1　表3及び表4は，プロジェクトPの途中段階での各指標（PV，AC，EV）の値を，工程別に示したものである。表3は2月末時点の値（1月と2月の合計）であり，表4は3月末時点の値（1～3月の合計）である。表中の［　　］に入れる正しい答えを，解答群の中から選べ。

なお，ACは各月での工程別の工数の実績値を基に算出している。

表3　2月末時点での各指標の値

	PV	AC	EV
外部設計	1,400	1,200	a
内部設計	1,000	1,050	1,000

表4　3月末時点での各指標の値

	PV	AC	EV
外部設計	1,600	1,600	1,600
内部設計	b	1,260	1,200
実装	750	625	750

aに関する解答群

ア　1,000　　イ　1,050　　ウ　1,200　　エ　1,400

bに関する解答群

ア　1,200　　イ　1,400　　ウ　1,600　　エ　1,800

設問2　次の記述は，プロジェクト P の 3 月末時点でのスケジュール差異及びコスト差異の分析について述べたものである。［　　］に入れる正しい答えを，解答群の中から選べ。

なお，各値は小数第3位を四捨五入するものとする。

スケジュールの進捗に関して，表2の結果から，内部設計はスケジュール遅延を起こしていることが明らかである。残りの外部設計と実装に関して，SV を用いてスケジュール差異の分析を行うと，［ c ］ことが分かる。

次に，工程別のコスト差異を分析する。CV の値は，［ d ］が負であり，コスト超過になっているが，［ e ］が正であり，計画よりもコスト低減されている。プロジェクト全体では，CPI が［ f ］であり，計画よりもコスト低減になっていることが分かる。

cに関する解答群

ア　外部設計と実装はSVがともに0で，計画どおりのスケジュールで進捗している
イ　外部設計と実装はSVがともに正で，計画よりもスケジュールが早まっている
ウ　外部設計は SV が負でスケジュール遅延であるが，実装は SV が 0 で計画どおりのスケジュールで進捗している
エ　外部設計は SV が負でスケジュール遅延であるが，実装は SV が正で計画よりもスケジュールが早まっている
オ　外部設計は SV が 0 で計画どおりであり，実装は SV が正で計画よりもスケジュールが早まっている

d，eに関する解答群

ア　外部設計　　イ　内部設計　　ウ　実装

fに関する解答群

ア　1.01　　イ　1.02　　ウ　1.03　　エ　1.04

設問3　プロジェクト P の今後の予測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。

プロジェクトメンバの努力で開発の遅れは順調に改善し，内部設計は4月半ばに完了し，実装も4月末までに完了した。その結果，4月のテストも順調に進捗し，スケジュールに関しては，プロジェクト全体として計画どおりに完了できる見込みである。

次に，コストについて予測する。4月の内部設計及び実装における1機能当たりの工数の実績値は，それぞれの1月から3月までの実績値と等しく，4月の1か月間での内部設計及び実装の工数の合計は［　g　］であった。4月のテストは計画どおりの工数で進捗した。そこで，5月のテストも計画どおりの工数で進捗すると仮定すると，プロジェクト全体での総コスト（総工数）の予測値は［　h　］となり，コストに関しても当初の計画値以下で完了できる見込みである。

gに関する解答群

ア　775　　イ　795　　ウ　850　　エ　870

hに関する解答群

ア　5,320　　イ　5,480　　ウ　5,560　　エ　5,600

問7　ゲーム理論を活用した出店戦略に関する次の記述を読んで，設問1，2に答えよ。

A社はドラッグストアチェーンで，地方都市X市を中心に20店舗を展開している。A社の店舗には，駅ビル内店舗と，郊外ショッピングモール内店舗の2種類がある。

A社のライバルであるB社は，同じく地方都市X市を中心に12店舗を展開しているドラッグストアチェーンである。B社の店舗には，駅ビル内店舗と，駅前商店街店舗の2種類がある。A社とB社の各店舗の種類と立地は，表1のとおりである。

なお，A社とB社が各店舗で取り扱う商品には，大きな相違点はない。

表1　店舗の種類と立地

店舗の種類	立地
駅ビル内店舗	駅に直結する建物内
駅前商店街店舗	駅前の商店街
郊外ショッピングモール内店舗	郊外にあるショッピングモール内

X市内のY地区は，私鉄のY駅を中心に開発が活発に進められている地区である。従って，表1に示すどの種類の店舗でも出店のための店舗スペースの確保が十分可能である。A社は来年度の事業展開としてY地区への1店舗の出店を計画している。A社は出店の方針として，駅ビル内店舗又は郊外ショッピングモール内店舗の2種類の店舗に絞っている。A社はY地区について，どちらの種類の店舗を出店すべきか戦略を立案することになった。

A社は，Y地区への出店に関して外部の調査機関に依頼して，Y地区に店舗を出店した場合の売上見込みなどの調査結果を得た。

〔市場環境〕

購買動機などの基準によって，消費者全体を幾つかの独立した小部分に区分したものを消費者セグメントと呼ぶ。Y地区における，ドラッグストアを利用する消費者全体を，利用する店舗の種類で四つの独立した消費者セグメントに区分した。それぞれのセグメントに対する月間売上見込みと，各セグメントが利用する店舗の種類を表2に示す。例えば，セグメント2に対する月間売上見込みは，駅ビル内店舗と駅前商店街店舗との合計で1,000万円となる。

表２　Ｙ地区の消費者セグメント別の売上見込みと利用する店舗の種類

消費者セグメント	セグメントに対する月間売上見込み	利用する店舗の種類		
		駅ビル内店舗	駅前商店街店舗	郊外ショッピングモール内店舗
セグメント１	2,000万円	○	×	×
セグメント２	1,000万円	○	○	×
セグメント３	1,000万円	×	○	○
セグメント４	1,000万円	×	×	○

注　○：対象となる消費者セグメント　×：対象とならない消費者セグメント

Ｙ地区における競合環境に関して，次のような情報が得られている。

〔競合環境〕

(1)　Ｘ市のＹ地区は，これまでドラッグストアチェーン店が出店したことはない。しかし，最近のＹ地区の人口増加傾向を受けて，Ａ社のライバルである Ｂ社も来年度，Ｙ地区に駅ビル内店舗又は駅前商店街店舗のいずれか１店舗を出店する可能性が高い。Ｂ社がどちらの種類の店舗を出店するのか，又は出店しないのかに関しての情報は入手できていない。

(2)　Ａ社とＢ社が競合する他地区での売上実績から推測して，Ｙ地区でＡ社とＢ社の店舗が同じ消費者セグメントを対象として販売する場合，対象とする消費者セグメントに対する売上は，双方の店舗で50％ずつ獲得するものと予想される。

設問１　調査結果に基づいて，Ｙ地区へのＡ社が採り得る出店戦略とＢ社が採り得る出店戦略との組合せによって，売上高がどうなるかの予測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)　［ a ］出店した場合，セグメント１及びセグメント２で見込まれる売上はＢ社が，セグメント３及びセグメント４で見込まれる売上はＡ社が独占して獲得する。

(2)　［ b ］出店した場合，セグメント１及びセグメント２で見込まれる売上の合計額を，両社が50％ずつ獲得する。

解答群

ア　A社が駅ビル内店舗を，B社が駅前商店街店舗を

イ　A社が郊外ショッピングモール内店舗を，B社が駅ビル内店舗を

ウ　A社が郊外ショッピングモール内店舗を，B社が駅前商店街店舗を

エ　A社，B社ともに駅ビル内店舗を

A社では，Y地区への出店戦略の検討に当たって，B社との競合が発生する可能性があることから，B社が採り得る出店戦略を考慮した上で，A社の売上を最大化すべく，ゲーム理論を活用することとした。そこで，調査結果に基づいて，A社が採り得る出店戦略とB社が採り得る出店戦略との組合せによって，売上がどうなるか利得行列を使って整理した。

利得行列とは，ゲームの要素である“プレイヤ”，“戦略”，“利得”の3要素を，表3のような行列の形で表したものである。例えば，プレイヤAが戦略A-1，プレイヤBが戦略B-1を採ったときのプレイヤA及びプレイヤBの利得は，網掛け部分で表される。

表3　利得行列

プレイヤB / プレイヤA	戦略B-1	戦略B-2
戦略A-1	（プレイヤAの利得，プレイヤBの利得）	（プレイヤAの利得，プレイヤBの利得）
戦略A-2	（プレイヤAの利得，プレイヤBの利得）	（プレイヤAの利得，プレイヤBの利得）

設問2　市場環境及び競合環境の記述に基づいて作成された，表4の利得行列の中，及び次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

表4　Y地区のA社並びにB社の月間売上高予測の利得行列

単位　百万円

A社＼B社	駅ビル内店舗	駅前商店街店舗	出店しない
駅ビル内店舗	（15，15）	（［c］，15）	（［d］，0）
郊外ショッピングモール内店舗	（20，30）	（［e］，15）	（20，0）

ゲーム理論では，相手がどのような戦略を採ったとしても，自分にとって最も有利となる戦略を支配戦略と呼ぶ。表4で予測した利得行列をB社の立場からみると，A社がどの戦略を採った場合でも，B社は［f］ことによって自社の売上を最大とすることができる。

そこで，B社が自社の売上を最大とすることができる戦略である［f］ことを仮定した場合，A社として自社の売上を最大とすることができる戦略は［g］ことであることが分かる。

c～eに関する解答群

ア　0　　イ　5　　ウ　10

エ　15　　オ　20　　カ　25

キ　30

f，gに関する解答群

ア　駅ビル内店舗を出店する

イ　駅前商店街店舗を出店する

ウ　郊外ショッピングモール内店舗を出店する

エ　Y地区への出店を見送る

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。

N個の要素中からK個の要素を選ぶ組合せをすべて求める。例えば，5個の要素中から3個の要素を選ぶ組合せの場合，計10通りある組合せをすべて求める。

プログラムでは，N個の要素（要素番号1～N）からなる配列Sを用意し，このうちK個の要素には1を，残りの要素には0を設定することによって，組合せの一つを表現する。例えば，図1(1)のように5個の要素1～5中から3個の要素2，4，5を選んだ状態は，プログラム中では図1(2)のとおりに表現する。

図1　5個の要素中から3個の要素を選ぶ例とそのプログラム中での表現

〔プログラムの説明〕

プログラムは，主プログラム Main 並びに組合せを求めるための関数 Init 及び Next からなる。

主プログラム Main

機能：N＝5，K＝3として，5個の要素中から3個の要素を選ぶ組合せ計10通りを順次求めて，配列Sに設定する。

整数型関数：Init(整数型: S[], 整数型: N, 整数型: K)

引数：S[]は出力用，N及びKは入力用の引数である。

機能：1≦K≦Nの場合，配列Sの先頭からK個の要素に1を，続くN－K個の要素に0をそれぞれ設定し，返却値として0を返す。それ以外の場合，配列Sには値を設定せずに，返却値として－1を返す。

整数型関数：Next(整数型: S[], 整数型: N)

引数：S[]は入出力用，Nは入力用の引数である。

機能：渡された配列 S の先頭から N 個の要素には，直近に求めた組合せの状態が設定されている。この渡された組合せの状態に対して所定の操作を行い，次の組合せの状態を求めて配列Sに設定し，返却値として0を返す。ただし，渡された組合せの状態が，この関数のアルゴリズムで得られる最終形である場合，配列Sには値を設定せずに，返却値として－1を返す。

〔プログラム〕

```
○主プログラム: Main
○整数型: S[5], K, N, R          /*  1 ≦ K ≦ N  */
• K ← 3                         /*  選択する要素の個数  */
• N ← 5                         /*  要素の個数  */
• R ← Init(S, N, K)
■ R = 0
  • R ← Next(S, N)
■
```

α

```
○整数型関数: Init(整数型: S[], 整数型: N, 整数型: K)
○整数型: L
▲ 1 ≦ K and K ≦ N
  ■ L: 1, L ≦ N, 1
    ▲ L ≦ K
      • S[L] ← 1
    ―――
      • S[L] ← 0
  ■
  • return 0
―――
  • return −1
▼
```

```
○整数型関数: Next(整数型: S[], 整数型: N)
○整数型: C, L, R
• C ← 0
• L ← 1
• R ← −1
■ L < N and R = −1
  ▲ S[L] = 1
    ▲ S[L+1] = 0
      • S[L] ← 0
      • S[L+1] ← 1
      • Init(S, L−1, C)
      • R ← 0
    ―――
      • C ← C + 1
  ▼
  • L ← L + 1
■
• return R
```

設問　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

(1) 主プログラム Main で，配列 S に組合せの一つの状態が得られるたびに，配列 S の内容を印字したい。印字には次の副プログラムを用いる。

副プログラム Dump(整数型: S[], 整数型: N)
　引数：S[]及び N は入力用の引数である。
　機能：配列 S の先頭から N 個の要素に格納されている値を，1行に印字する。

そのためには，主プログラム Main の α の部分を［ a ］に示す部分と入れ替えればよい。

(2) 関数 Next は，受け取った配列 S を要素番号の小さい方から検査し，連続する2要素の値が［ b ］に見つかったものについて，その内容を入れ替える。続いて，配列 S の一部でその2要素［ c ］の部分について関数 Init を呼ぶ。例えば，関数 Next の実行開始時点で，配列 S の要素番号 1～5 の内容が 1，0，1，0，1 であったとき，実行終了時点での配列 S の要素番号 1～5 の内容は［ d ］となる。

(3) このプログラムを実行して，関数 Init が関数 Next から呼ばれるとき，関数 Init が受け取る N の値の範囲は［ e ］，K の値の範囲は［ f ］である。したがって，関数 Init が受け取る N と K の値は，1 ≦ K ≦ N を満たさない場合がある。

(4) 主プログラム Main の実行終了時点において，配列 S の要素番号 1～5 の内容は［ g ］となっている。

aに関する解答群

ア

```
• R ← Init(S, N, K)
■ R = 0
  • Dump(S, N)
  • R ← Next(S, N)
■
```

イ

```
• R ← Init(S, N, K)
■ R = 0
  • R ← Next(S, N)
  • Dump(S, N)
■
```

ウ

```
• R ← Init(S, N, K)
• Dump(S, N)
■ R = 0
  • R ← Next(S, N)
  • Dump(S, N)
■
```

エ

```
• R ← Init(S, N, K)
■ R = 0
  • Dump(S, N)
  • R ← Next(S, N)
■
• Dump(S, N)
```

bに関する解答群

ア 0, 1で最後　　イ 0, 1で最初　　ウ 1, 0で最後　　エ 1, 0で最初

cに関する解答群

ア 及びその後　　イ 及びその前　　ウ より後　　エ より前

dに関する解答群

ア 0, 1, 1, 0, 1　　イ 1, 0, 0, 1, 1　　ウ 1, 0, 1, 1, 0　　エ 1, 1, 0, 0, 1

e, fに関する解答群

ア 0～2　　イ 0～3　　ウ 1～3　　エ 1～4

オ 2～4　　カ 2～5

gに関する解答群

ア 0, 0, 0, 0, 0　　イ 0, 0, 1, 1, 1　　ウ 1, 1, 1, 0, 0　　エ 1, 1, 1, 1, 1

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1の説明〕

U劇場の座席予約システムにおいて，希望する座席種別と座席数を指定すると，連続した空き（未予約）座席があるかどうかを調べるプログラムである。

(1)　U劇場の座席配置は図1に示すとおりである。

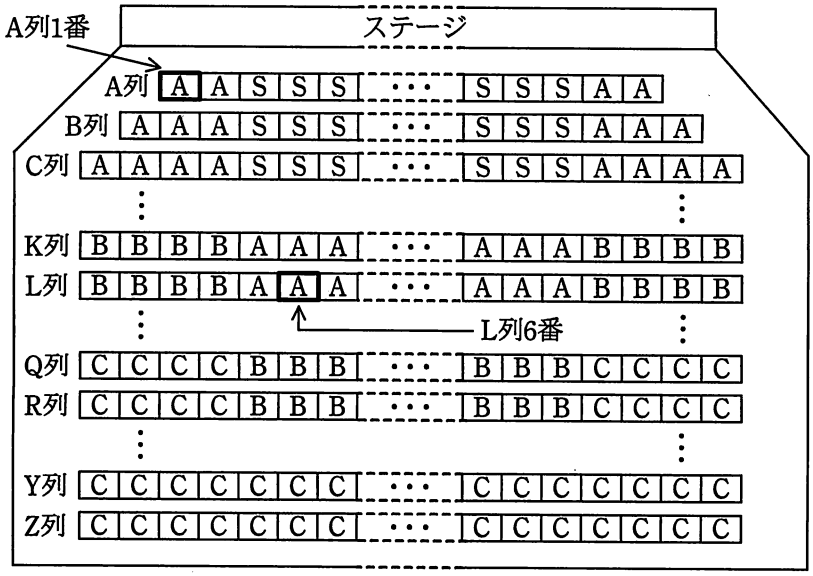


図1　U劇場の座席配置

① U劇場には，ステージ側から列名がA～Zの26列の座席があり，各列の座席数は異なる。

② 各列の座席は，ステージに向かって一番左から順に1，2，…と数え，列名のA～Zと，この数（以下，番という）を組み合わせた座席番号で識別される。例えば，“L列6番”の座席とは，12列目（L列）のステージに向かって一番左から6番目の座席を示す。

③ 各座席には，料金の違いを示す種別（以下，座席種別という）が割り振られている。座席種別は，料金が高いものから順にS，A，B，Cの四つがある。図1中の座席の文字は座席種別を表している。

(2) 空き座席を探すときには，ステージに向かって最前列の一番左（A列1番）から開始して，同一の列に希望する座席種別と座席数の連続した空きがあるかどうかを調べる。A列で見つからなかった場合は，順次B列，C列，…と後列を対象にして同様に調べる。

(3) 関数 check_seats 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。

class	希望の座席種別
num	希望の座席数
hall	座席表
pos	確認結果（POSITION 型の構造体へのポインタ）

① 連続した空き座席が見つかった場合，最初に見つかった空きの中で，ステージに向かって一番左に位置する座席の座席番号をposが指す構造体に格納する。

② 連続した空き座席が見つからなかった場合，pos が指す構造体の座席の列名にナル文字を格納する。

(4) 構造体 SEAT の構造は次のとおりである。

```
typedef struct {
    char seat_class; /* 座席種別 */
    int  reserved;   /* 予約状態 */
} SEAT;
```

(5) 構造体 POSITION の構造は次のとおりである。

```
typedef struct {
    char seat_row;   /* 座席の列名 */
    int  seat_no;    /* 座席の番 */
} POSITION;
```

(6) 座席表 hall の要素 hall[i][j]は SEAT 型の構造体であり，各座席の座席種別と予約状態が次のとおり格納されている。ここで，添字 i の 0～25 が列名 A～Z に，添字 j の 0～Ni－1が番 1～Ni(Niは，添字 i に対応する列の座席数)に対応している。

hall[i][j].seat_class：座席種別を表す“S”，“A”，“B”，“C”のいずれかの文字が格納されている。

hall[i][j].reserved：予約状態が格納されている。0 は空きを，1 は予約済を表す。

各列のステージに向かって一番右の座席の次の要素の座席種別 hall[i][Ni].seat_class には，ナル文字が格納されている。

〔プログラム 1〕

```
#define ROWNUM  26    /* 座席の列数 */

typedef struct {
    char seat_class;  /* 座席種別 */
    int  reserved;    /* 予約状態 */
} SEAT;
typedef struct {
    char seat_row;    /* 座席の列名 */
    int  seat_no;     /* 座席の番 */
} POSITION;

static char row_s[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

void check_seats(char, int, SEAT *[ROWNUM], POSITION *);

void check_seats(char class, int num, SEAT *hall[ROWNUM],
                 POSITION *pos) {
    int cnt, found = 0, no, row;

    for (row = 0; row < ROWNUM; row++) {
        cnt = 0;
        for (no = 0; hall[row][no].seat_class != '\0'; no++) {
            if ((hall[row][no].seat_class == class) &&
                (hall[row][no].reserved == 0)) {
                if (++cnt >= num) {
                    [ a ];
                    break;
                }
            } else {
                [ b ];
            }
        }
        if (found != 0) {
            break;
        }
    }

    if (found != 0) {
        pos->seat_row = [ c ];
        pos->seat_no  = [ d ];
```

```
    } else {
        pos->seat_row  = '\0';
    }
}
```

設問1　プログラム1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　cnt = 0	イ　cnt--	ウ　cnt++
エ　found = 0	オ　found = 1	

c，dに関する解答群

ア　no	イ　no - num	ウ　no - num + 1
エ　no - num + 2	オ　row	カ　row + 1
キ　row_s[row]	ク　row_s[row + 1]	

設問2　関数 check_seats を使って，希望する座席種別の連続した空き座席があるかどうかを調べ，その結果を出力する。希望する座席種別の連続した空き座席がない場合には，ほかの座席種別で希望する座席数の連続した空き座席があるかどうかを調べ，その結果を出力するプログラムを作成した。作成したプログラムに関する説明文中の［　　］に入れる正しい答えを，解答群の中から選べ。

(1)　関数 check_service 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。

class	希望の座席種別
num	希望の座席数
hall	座席表
opt	結果の出力順

(2)　関数 check_service で使用している関数 print_seats の仕様は次のとおりである。

```
void print_seats(char class, int num, POSITION *pos)
```

機能：指定の座席種別の連続した空き座席があるかどうかの確認結果を出力する。

引数：class　座席種別
　　num　座席数
　　pos　確認結果（POSITION型の構造体へのポインタ）

〔プログラム2〕

```
#define ROWNUM  26    /* 座席の列数 */
#define CLSNUM   4    /* 座席種別数 */

typedef struct {
    char seat_class;  /* 座席種別 */
    int  reserved;    /* 予約状態 */
} SEAT;
typedef struct {
    char seat_row;    /* 座席の列名 */
    int  seat_no;     /* 座席の番 */
} POSITION;

static char class_s[CLSNUM] = "SABC";

void check_service(char, int, SEAT *[ROWNUM], int);
void check_seats(char, int, SEAT *[ROWNUM], POSITION *);
void print_seats(char, int, POSITION *);

void check_service(char class, int num, SEAT *hall[ROWNUM], int opt)
{
    int i;
    char c;
    POSITION pos;

    check_seats(class, num, hall, &pos);
    print_seats(class, num, &pos);
    if (pos.seat_row == '\0') {
        for (i =0; i < CLSNUM; i++) {
            if (opt == 0) {
                c = class_s[i];
            } else {
                c = class_s[CLSNUM - i - 1];
            }
            if (class != c) {
                check_seats(c, num, hall, &pos);
                print_seats(c, num, &pos);
            }
        }
    }
}
```

〔プログラム２の説明〕

(1) 希望する座席種別の連続した空き座席があるかどうかを調べ，その結果を出力する。

(2) 希望する座席種別の連続した空き座席がない場合には，希望する座席種別［ e ］に対して，結果の出力順 opt が 0 の場合は座席種別の料金が［ f ］ものから順に，結果の出力順 opt が 0 以外の場合は座席種別の料金が［ g ］ものから順に希望する座席数の連続した空きがあるかどうかを調べ，その結果を順次出力する。

C

eに関する解答群

ア　よりも料金の高いすべての座席種別
イ　よりも料金の安いすべての座席種別
ウ　よりも一つ料金の高い座席種別
エ　よりも一つ料金の安い座席種別
オ　を除くすべての座席種別

fに関する解答群

ア　高い　　イ　安い

gに関する解答群

ア　高い　　イ　安い

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〕

小売業のC社では，自社株主へ株主優待として，自社店舗で利用できる商品券を発行することにした。株主の保有株数と保有期間によって，発行額を決定する。この会社では，毎年3月31日の株取引終了時点で自社株を保有している株主を対象とした株主一覧を作成しており，商品券は，2011年3月31日に作成した株主一覧に登録されている個人株主に対して発行する。

このプログラムは，株主ファイルを読み込み，発行する商品券の発行額を商品券ファイルに出力する。

(1)　株主ファイルは，2011年3月31日に作成した株主一覧に登録されている個人株主を対象に作成した，図1に示すレコード様式の順ファイルである。

株主番号 8けた	保有株数 6けた	登録年 4けた

図1　株主ファイルのレコード様式

①　株主番号には，株主に対して一意に割り当てた番号が設定されている。

②　保有株数には，保有する株数が設定されている。保有株数の最大は999,999とする。

③　登録年には，最初に株主一覧に登録された年が設定されている。ただし，株をすべて売却した後に再取得した株主の場合は，再取得後，株主一覧に登録された年が設定される。

(2)　保有期間は，登録年から2011年までの年数で決定する。例えば，登録年が2006年の場合は5年である。

(3)　この会社が株式公開したのは1990年4月1日であり，最初に株主一覧へ登録されるのは1991年3月31日である。したがって，最長保有期間は20年である。

(4)　商品券ファイルは，図2に示すレコード様式の順ファイルである。

株主番号 8けた	発行額 6けた

図2　商品券ファイルのレコード様式

発行額には，各株主に対する商品券の発行額を設定する。発行額は，表1に示す決定表によって決まる。

表1　決定表

条件1 （保有株数）	999株 以下	Y	Y	N	N	N	N
	1,000 ～ 9,999株	N	N	Y	Y	N	N
	10,000株 以上	N	N	N	N	Y	Y
条件2 （保有期間）	5年未満	Y	N	Y	N	Y	N
	5年以上	N	Y	N	Y	N	Y
動作 （発行額）	1,000円	X	−	−	−	−	−
	3,000円	−	X	X	−	−	−
	5,000円	−	−	−	X	X	−
	10,000円	−	−	−	−	−	X

〔プログラム〕

（行番号）

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD  HOLDER.
 4  01  HLD-REC.
 5      02  HLD-NO          PIC 9(8).
 6      02  HLD-STOCK       PIC 9(6).
 7      02  HLD-YEAR        PIC 9(4).
 8  FD  GIFT.
 9  01  GFT-REC.
10      02  GFT-NO          PIC 9(8).
11      02  GFT-SUM         PIC 9(6).
12  WORKING-STORAGE SECTION.
13  77  READ-FLAG           PIC X(1) VALUE SPACE.
14      88  EOF             VALUE "E".
15  77  BASE-YEAR           PIC 9(4) VALUE 2011.
16  77  TERM                PIC 9(4).
17  PROCEDURE DIVISION.
18  MAIN-PROC SECTION.
19      OPEN INPUT HOLDER.
20      OPEN OUTPUT GIFT.
21      PERFORM UNTIL EOF
22         READ HOLDER AT END     SET EOF TO TRUE
23                        NOT AT END PERFORM SUM-PROC
24         END-READ
25      END-PERFORM.
26      CLOSE HOLDER GIFT.
27      STOP RUN.
```

```
28  SUM-PROC SECTION.
29      MOVE HLD-NO TO GFT-NO.
30      COMPUTE TERM = BASE-YEAR - HLD-YEAR.
31      EVALUATE HLD-STOCK ALSO TERM
32          WHEN [ a ] THRU [ b ] ALSO [ c ] THRU [ d ]
33              MOVE 1000 TO GFT-SUM
34          WHEN [ a ] THRU [ b ] ALSO [ e ]
35          WHEN [ f ] THRU [ g ] ALSO [ c ] THRU [ d ]
36              MOVE 3000 TO GFT-SUM
37          WHEN [ f ] THRU [ g ] ALSO [ e ]
38          WHEN [ e ]            ALSO [ c ] THRU [ d ]
39              MOVE 5000 TO GFT-SUM
40          WHEN [ e ]            ALSO [ e ]
41              MOVE 10000 TO GFT-SUM
42      END-EVALUATE.
43      WRITE GFT-REC.
```

COBOL

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　1	イ　9	ウ　10	エ　99
オ　999	カ　1000	キ　ANY	ク　OTHER

c～eに関する解答群

ア　0	イ　1	ウ　4	エ　5
オ　9	カ　10	キ　ANY	ク　OTHER

f，gに関する解答群

ア　1000	イ　3000	ウ　5000	エ　9999
オ　10000	カ　20000	キ　ANY	ク　OTHER

設問2　保有期間の傾向を確認するため，商品券の発行額の算出と同時に，図3に示す棒グラフを表示するようプログラムを変更したい。表2中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔グラフの説明〕

(1)　行の先頭に表示してある数字は，保有期間を表す。

(2)　“＊”は，当該保有期間に該当する株主比率を表す。

(3)　株主比率は，2％未満の端数を切り捨て，“＊”一つを2％として表示する。

(4)　目盛りの数字の単位は10％である。

```
   ----1----2----3----4----5----6----7----8----9----10
00: ***********
01: **
02: ***
03: *
 :
20: ****
```

図3　保有期間別の株主比率の表示例

表2　プログラムの変更

処置	変更内容
行番号16と17の間に追加	77 CNT PIC 9(2). 77 CNT-ALL PIC 9(8) VALUE 0. 01 . 02 CNT-HLD OCCURS 21 PIC 9(8) VALUE 0. 01 RATIO PIC 9(2) VALUE 0. 01 PRT-DATA. 02 PRT-TERM PIC 9(2). 02 PRT-DLIM PIC X(2). 02 PRT-EDIT PIC X(51). 01 PRT-HEAD PIC X(51) VALUE "----1----2----3----4----5----6----7----8----9----10".

行番号26と27の間に追加	IF CNT-ALL NOT = 0 THEN PERFORM GRAPH-PROC END-IF.
行番号43の後ろに追加	[h]. ADD 1 TO CNT-ALL. GRAPH-PROC SECTION. MOVE SPACE TO PRT-DATA. MOVE PRT-HEAD TO PRT-EDIT. DISPLAY PRT-DATA. MOVE ":" TO PRT-DLIM. PERFORM VARYING CNT FROM 0 BY 1 UNTIL CNT > 20 MOVE SPACE TO PRT-EDIT MOVE CNT TO PRT-TERM [i] IF RATIO > 0 THEN MOVE ALL "*" TO PRT-EDIT(1:RATIO) END-IF DISPLAY PRT-DATA END-PERFORM.

h に関する解答群

ア　ADD 1 TO CNT-HLD(CNT)

イ　ADD 1 TO CNT-HLD(CNT + 1)

ウ　ADD 1 TO CNT-HLD(CNT - 1)

エ　ADD 1 TO CNT-HLD(TERM)

オ　ADD 1 TO CNT-HLD(TERM + 1)

カ　ADD 1 TO CNT-HLD(TERM - 1)

i に関する解答群

ア　COMPUTE RATIO = (CNT-ALL * 100 / CNT-HLD(CNT)) / 2

イ　COMPUTE RATIO = (CNT-ALL * 100 / CNT-HLD(CNT + 1)) / 2

ウ　COMPUTE RATIO = (CNT-ALL * 100 / CNT-HLD(CNT - 1)) / 2

エ　COMPUTE RATIO = (CNT-HLD(CNT) * 100 / CNT-ALL) / 2

オ　COMPUTE RATIO = (CNT-HLD(CNT + 1) * 100 / CNT-ALL) / 2

カ　COMPUTE RATIO = (CNT-HLD(CNT - 1) * 100 / CNT-ALL) / 2

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

〔プログラムの説明〕

クラス ArrayAppendableCharSequence 及び ListAppendableCharSequence は，追加可能な0個以上の文字からなる文字列（文字の並び）を表現するインタフェース AppendableCharSequence を異なるデータ構造で実装したプログラムである。

インタフェース AppendableCharSequence は，パッケージ java.lang のインタフェース CharSequence 及び Appendable で定義されている次のメソッドからなる。

(1) メソッド charAt は，引数 index で指定された位置の文字を返す。文字の位置はインデックス値で表され，文字列の長さが1以上のとき，最初の文字の位置は0，最後の文字の位置は文字列の長さ－1で表される。引数で指定されたインデックス値が負又は文字列の長さ以上の場合は，IndexOutOfBoundsException を投げる。

(2) メソッド length は，文字列の長さを返す。

(3) メソッド append は，引数で指定された文字を文字列の末尾に追加する。戻り値として，この AppendableCharSequence のインスタンス自身を返す。

(4) メソッド toString は，このインスタンスの文字列を String オブジェクトで返す。

クラス ArrayAppendableCharSequence は，char 型の配列を用いてインタフェース AppendableCharSequence を実装したものである。メソッド append は，配列の空いている最初の要素に文字を格納する。配列に空き要素がないときは，要素の個数が現在の配列よりも EXT_SIZE の値だけ大きな配列を生成し，既存の文字データを移し，空いている最初の要素に文字を格納する。

クラス ListAppendableCharSequence は，連結リスト構造を用いてインタフェース AppendableCharSequence を実装したものである。入れ子クラス ListAppendableCharSequence.Bucket（以下，Bucket という）は，連結リストの要素である。Bucket のインスタンス1個につき，要素の個数が EXT_SIZE である char 型の配列を用意し，そこに文字データを保持する。Bucket の配列に空き要素がなくなり，次の文字を追加できないときは，新規に Bucket のインスタンスを生成し，連結リストに追加する。

クラス Test は，上記二つのクラスそれぞれの実装が異なるメソッド append の実

行時間を測定するプログラムである。メソッド measureTime は，引数 n で指定された回数だけ，引数 a で指定されたインスタンスのメソッド append を呼び出し，実行時間をミリ秒単位で測定する。

〔プログラム1〕

```
public interface AppendableCharSequence {
   public char charAt(int index);
   public int length();
   public AppendableCharSequence append(char c);
   public String toString();
}
```

〔プログラム2〕

```
public class ArrayAppendableCharSequence
             implements AppendableCharSequence {
   private static final int EXT_SIZE = 10;
   private int length;
   private char[] data;

   public ArrayAppendableCharSequence() {
      data = new char[EXT_SIZE];
   }

   public char charAt(int index) {
      if (index < 0 || index >= length)
         throw new IndexOutOfBoundsException();
      return data[index];
   }

   public int length() {
      return length;
   }

   public AppendableCharSequence append(char c) {
      if (data.length == length) {
         char[] temp = new char[length + EXT_SIZE];
         for (int i = 0; i < [ a ]; i++) {
             temp[i] = data[i];
         }
         data = temp;
      }
      data[length++] = c;
      return this;
   }
```

```
    public String toString() {
       return new String(data, 0, length);
    }
  }
```

〔プログラム3〕

```
public class ListAppendableCharSequence
              implements AppendableCharSequence {
   private static final int EXT_SIZE = 10;
   private Bucket bucketList;
   private int length;

   public ListAppendableCharSequence() {
      bucketList = new Bucket();
   }

   public char charAt(int index) {
      if (index < 0 || index >= length)
         throw new IndexOutOfBoundsException();
      Bucket bucket = getBucket(index);
      return bucket.data[    b    ];
   }

   public int length() {
      return length;
   }

   public AppendableCharSequence append(char c) {
      int offset = length % EXT_SIZE;
      Bucket bucket = getBucket(    c    );
      if (offset == 0 && length != 0) {
         bucket.next = new Bucket();
         bucket = bucket.next;
      }
      bucket.data[offset] = c;
      length++;
      return this;
   }

   public String toString() {
      char[] data = new char[length];
      Bucket bucket = bucketList;
      int size;
      for (int len = length; len > 0; len -= size) {
         size = (len >= EXT_SIZE) ? EXT_SIZE : len;
         for (int i = 0, j =    d    ; i < size; i++, j++) {
             data[j] = bucket.data[i];
         }
```

```
            bucket = bucket.next;
        }
        return new String(data);
    }

    private Bucket getBucket(int index) {
        int n = index / EXT_SIZE;
        Bucket b = bucketList;
        for (int i = 0; i < n; i++) {
            b = b.next;
        }
        return b;
    }

    private static class Bucket {
        private Bucket next;
        private char[] data = new char[EXT_SIZE];
    }
}
```

〔プログラム4〕

```
public class Test {
    static final int[] TIMES = {
        5000, 10000, 50000, 100000
    };

    public static void main(String[] args) {
        for (int n : TIMES) {
            measureTime(n, new ArrayAppendableCharSequence());
            measureTime(n, new ListAppendableCharSequence());
        }
    }

    static void measureTime(int n, [ e ] a) {
        long start = System.currentTimeMillis();
        for (int i = 0; i < n; i++) {
            a.append((char) (i % 26 + 'a'));
        }
        long end = System.currentTimeMillis();
        System.out.printf("%s: %d [times] %d [ms]%n",
                          a.getClass().getName(),
                          n, end - start);
    }
}
```

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　data.length + length　　イ　data.length + temp.length

ウ　length　　エ　length + EXT_SIZE

オ　temp.length

bに関する解答群

ア　index　　イ　index % EXT_SIZE

ウ　index / EXT_SIZE　　エ　length

オ　length % EXT_SIZE　　カ　length / EXT_SIZE

cに関する解答群

ア　length　　イ　length + 1　　ウ　length - 1

エ　offset　　オ　offset + 1　　カ　offset - 1

dに関する解答群

ア　len　　イ　len % EXT_SIZE　　ウ　len - length

エ　length　　オ　length % EXT_SIZE　　カ　length - len

eに関する解答群

ア　AppendableCharSequence

イ　ArrayAppendableCharSequence

ウ　ListAppendableCharSequence

エ　ListAppendableCharSequence.Bucket

オ　Object

カ　String

設問2　クラス Test を用いて各クラスのメソッド append の実行時間（ミリ秒）を測定したところ，表1の結果になった。

表1　各クラスのメソッド append の実行時間

単位　ミリ秒

呼出し回数 / クラス名	5,000回	10,000回	50,000回	100,000回
ArrayAppendableCharSequence	5	18	370	1,455
ListAppendableCharSequence	4	8	960	4,085

文字の追加回数が多くなるにつれて ListAppendableCharSequence の方が遅くなるのは，連結リストをたどる回数が増えるためであると推測される。そこで，ListAppendableCharSequence.append の性能を改善するために，次の修正を行った。［　　］に入れる正しい答えを，解答群の中から選べ。

なお，ほかの変更はないものとする。

(1)　連結リストの最後の要素を常に参照するためのフィールド last を追加する。

```
private Bucket last;
```

(2)　コンストラクタでフィールド last を初期化する。

```
public ListAppendableCharSequence() {
   bucketList = new Bucket();
   last = [    f    ];
}
```

(3)　メソッド append でフィールド last を利用する。

```
public AppendableCharSequence append(char c) {
   int offset = length % EXT_SIZE;
   if (offset == 0 && length != 0) {
      last.next = new Bucket();
      last = [    g    ];
   }
   last.data[offset] = c;
   length++;
   return this;
}
```

解答群

ア　bucketList	イ　bucketList.next	ウ　last
エ　last.next	オ　new Bucket()	カ　null

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 の説明〕

連続する 16 語に格納した 16×16 ドットの白黒の図形を，時計回りに 90 度回転する副プログラム ROTATE である。図形の回転の実行例を図 1 に示す。このとき，1 ドットを 1 ビットで表し，白は 0，黒は 1 が格納されている。

(1)　図形を表す 16 語の先頭アドレスは GR1 に設定されて，主プログラムから渡される。

(2)　回転した結果の図形を格納する領域の先頭アドレスは GR2 に設定されて，主プログラムから渡される。

(3)　元の図形と回転した図形は，異なる領域に格納される。

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

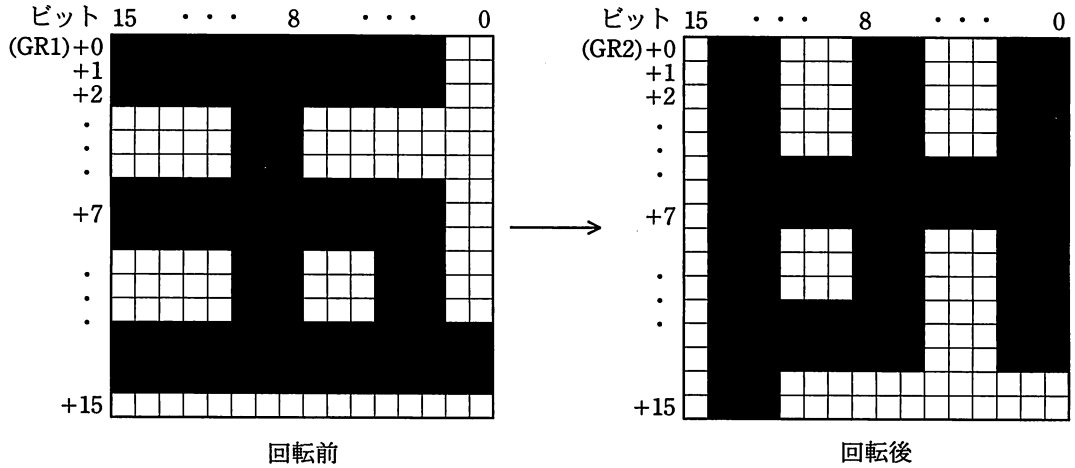


図 1　図形の回転の実行例

アセンブラ

〔プログラム1〕

(行番号)

```
 1  ROTATE  START
 2          RPUSH
 3          LD      GR3,=16
 4  LOOP1   LD      GR4,=16
 5          LD      GR5,GR2         ; GR5 ← 結果の領域のアドレス
 6          LD      GR6,0,GR1       ; GR6 ← 元の図形の先頭語の内容
 7  LOOP2   LD      GR7,0,GR5       ; GR7 ← 結果の領域の1語の内容
 8          SRL     GR7,1
 9          SLL     GR6,1
10          JOV     ON
11          JUMP    CONT
12  ON      OR      GR7,=#8000
13  CONT    ST      GR7,0,GR5       ; 処理した1語を結果の領域に格納
14          [      a      ]
15          SUBA    GR4,=1
16          JNZ     LOOP2
17          [      b      ]
18          SUBA    GR3,=1
19          JNZ     LOOP1
20          RPOP
21          RET
22          END
```

設問1　プログラム1中の［　　］に入れる正しい答えを，解答群の中から選べ。

解答群

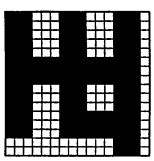
ア　LAD　GR1,-1,GR1　　イ　LAD　GR1,1,GR1　　ウ　LAD　GR2,-1,GR2

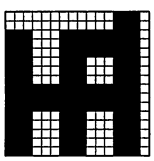
エ　LAD　GR2,1,GR2　　オ　LAD　GR5,-1,GR5　　カ　LAD　GR5,1,GR5

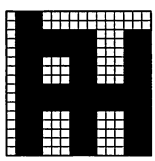
設問2　行番号9を次のとおりに変更し，図1の回転前を元の図形として実行した。実行結果の図形として正しい答えを，解答群の中から選べ。

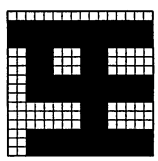
```
        SRL     GR6,1
```

解答群

ア　 

イ　 

ウ　 

エ　 

設問3　プログラム1中の行番号3，4をプログラム2に置き換えて，16×16ドットの図形のうち左上のn×nドットの部分だけを時計回りに90度回転するプログラムとした。左上の8×8ドットの部分だけを回転した実行例を図2に示す。

値n（1≦n≦16）はGR3に設定されて主プログラムから渡される。

置換え後のプログラムは，まず，回転の対象とならないドット（元の図形の上n語の右（16−n）ビットと下（16−n）語の全ビット）を結果の領域の適切な場所に複写する。その後，左上のn×nドットの部分を回転して結果の領域に格納する。

プログラム2中の［　　］に入れる正しい答えを，解答群の中から選べ。

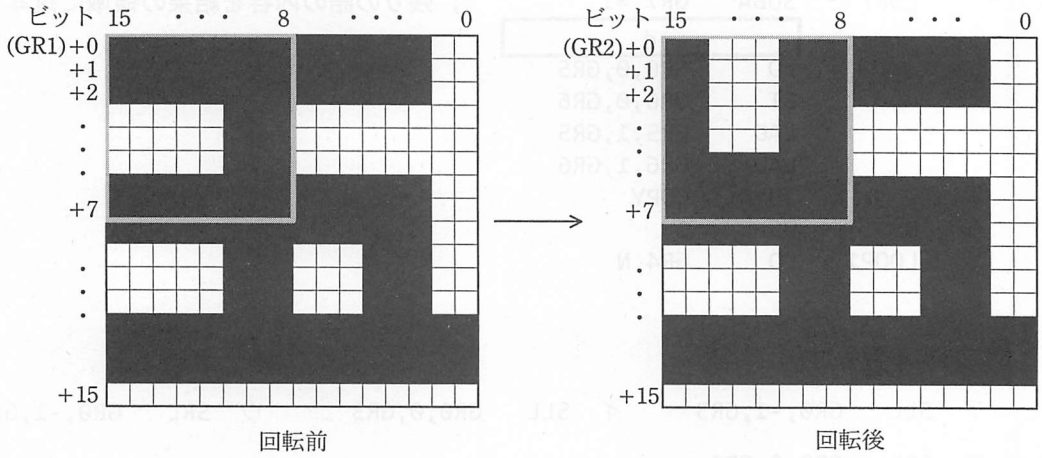


図2　左上の8×8ドットの部分だけを回転した実行例

アセンブラ

〔プログラム2〕

```
            ST      GR3,N           ; n を保存
            LD      GR4,GR3         ; GR4 ← n
            LD      GR5,GR1         ; GR5 ← 元の図形のアドレス
            LD      GR6,GR2         ; GR6 ← 結果の領域のアドレス
            LD      GR7,=16
            SUBA    GR7,GR3         ; GR7 ← 16-n
SHIFT       LD      GR0,0,GR5       ; GR0 ← 元の図形の1語の内容
            [   c   ]
            ST      GR0,0,GR6       ; 結果の領域 ← GR0
            LAD     GR5,1,GR5       ; 元の図形の1語のアドレス更新
            LAD     GR6,1,GR6       ; 結果の領域の1語のアドレス更新
            SUBA    GR4,=1          ; n 語処理済み？
            JNZ     SHIFT
COPY        SUBA    GR7,=1          ; 残りの語の内容を結果の領域に複写
            [   d   ]
            LD      GR0,0,GR5
            ST      GR0,0,GR6
            LAD     GR5,1,GR5
            LAD     GR6,1,GR6
            JUMP    COPY
N           DS      1
LOOP1       LD      GR4,N
```

cに関する解答群

ア　SLL　GR0,-1,GR3　　イ　SLL　GR0,0,GR3　　ウ　SRL　GR0,-1,GR3

エ　SRL　GR0,0,GR3

dに関する解答群

ア　JMI　LOOP1　　イ　JNZ　LOOP1　　ウ　JPL　LOOP1

エ　JZE　LOOP1

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

〔表計算の説明〕

企業が行う取引には，商品を販売してから一定期間後に販売代金を取引先から回収する方法がある。回収していない状態の代金のことを売掛金という。取引先の信用度を様々な角度から評価して，取引先に対する売掛金の限度額である与信枠を設定し，売掛金を確実に回収できるように管理することを与信管理という。K 社では，21 の取引先ごとに月 1 回の注文を受けると，即日納品をして，月末に請求を行っている。K 社では，与信管理を次の手順で行っている。

① 取引先の流動資産，流動負債，自己資本などの情報を収集する。

② 流動比率及び自己資本比率を表 1 に示すレベルに当てはめ，取引先の信用度を評価するとともに，それぞれのレベルに対応する与信枠の計算式を用いて，基準 1 及び基準 2 の計算値を求める。

③ 基準 1 と基準 2 の計算値のうち，小さい方を与信枠とする。ただし，小さい方の値が負の場合，0 とする。

表 1　取引先の信用度の評価と与信枠の計算式

評価指標		レベル	信用度	与信枠の計算式
基準 1	流動比率	90％未満	取引停止	0
		90～130％未満	慎重	（流動資産－流動負債）×0.5
		130～150％未満	現状維持	（流動資産－流動負債）×1
		150～200％未満	積極	（流動資産－流動負債）×1.5
		200％以上	最優先	（流動資産－流動負債）×2
基準 2	自己資本比率	5％未満	取引停止	0
		5～20％未満	慎重	自己資本×0.2
		20～40％未満	現状維持	自己資本×0.5
		40～70％未満	積極	自己資本×1
		70％以上	最優先	自己資本×1.5

④ ②の二つの基準に基づく信用度の組合せから，表 2 に示すとおり，上限を 3 か月とした支払いサイトを決めている。ここで，支払いサイトとは請求月から支払期限までの月数である。表 2 中の“－”は，取引停止であること示している。

⑤ 前月の売掛金の残額に当月の注文額を加えた額が与信枠を超える場合，及び取引停止の場合には，当月の注文を受けない。

表２　信用度に基づく支払いサイト

単位　月数

基準１ ＼ 基準２	取引停止	慎重	現状維持	積極	最優先
取引停止	－	－	－	－	－
慎重	－	0	1	2	2
現状維持	－	1	2	2	2
積極	－	2	2	3	3
最優先	－	2	2	3	3

設問１　取引先の信用度評価と与信枠の計算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。

〔ワークシート：信用度評価基準〕

二つの基準のそれぞれのレベルに応じた信用度及び与信枠を求める計算式で利用する係数（以下，与信枠係数という）を登録した図１のワークシート“信用度評価基準”を作成した。

	A	B	C	D	E	F
1	評価指標		レベル	下限値（%）	信用度	与信枠係数
2	基準１	流動比率	90％未満	0	取引停止	0.0
3			90～130％未満	90	慎重	0.5
4			130～150％未満	130	現状維持	1.0
5			150～200％未満	150	積極	1.5
6			200％以上	200	最優先	2.0
7	基準２	自己資本比率	5％未満	0	取引停止	0.0
8			5～20％未満	5	慎重	0.2
9			20～40％未満	20	現状維持	0.5
10			40～70％未満	40	積極	1.0
11			70％以上	70	最優先	1.5

図１　ワークシート“信用度評価基準”

(1)　流動比率のレベルごとの下限値をセル D2～D6 に，自己資本比率のレベルごとの下限値をセル D7～D11 に入力する。

(2)　流動比率及び自己資本比率のレベルに応じた信用度をセル E2～E6 及びセル E7～E11 に入力する。

(3)　流動比率及び自己資本比率のレベルに応じた与信枠係数をセル F2～F6 及

表計算

びセル F7～F11 に入力する。

〔ワークシート：与信枠〕

取引先の信用度及び与信枠を求めるために，図 2 のワークシート“与信枠”を作成した。ここで，複数のワークシート間でデータを参照するには“ワークシート名！セル”又は“ワークシート名！範囲”という形式で指定する。

	A	B	C	D	E	F	G	H	I	J	K	L
1	取引先	財務データ						信用度		計算値（千円）		与信枠（千円）
2		流動資産（百万円）	流動負債（百万円）	流動比率（%）	自己資本（百万円）	総資本（百万円）	自己資本比率（%）	基準 1	基準 2	基準 1 での計算値	基準 2 での計算値	
3	K1	491	351	139.9	60	497	12.1	現状維持	慎重	140,000	12,000	12,000
4	K2	1,011	742	136.3	275	553	49.7	現状維持	積極	269,000	275,000	269,000
5	K3	855	680	125.7	301	818	36.8	慎重	現状維持	87,500	150,500	87,500
6	K4	587	800	73.4	50	211	23.7	取引停止	現状維持	0	25,000	0
7	K5	1,347	976	138.0	292	706	41.4	現状維持	積極	371,000	292,000	292,000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	K17	2,330	1,900	122.6	200	397	50.4	慎重	積極	215,000	200,000	200,000
20	K18	1,829	1,722	106.2	267	852	31.3	慎重	現状維持	53,500	133,500	53,500
21	K19	433	380	113.9	310	1,087	28.5	慎重	現状維持	26,500	155,000	26,500
22	K20	739	532	138.9	444	1,167	38.0	現状維持	現状維持	207,000	222,000	207,000
23	K21	543	489	111.0	59	430	13.7	慎重	慎重	27,000	11,800	11,800

図 2　ワークシート“与信枠”

(1)　セル A3 ～ G23 に，取引先ごとの名称及び財務データ（流動資産，流動負債，流動比率，自己資本，総資本，自己資本比率）を入力する。

(2)　基準 1 の信用度の評価結果を求めるために次の計算式をセル H3 に入力し，セル H4～H23 に複写する。

垂直照合(a)

基準 2 の信用度の評価結果を求めるために次の計算式をセル I3 に入力し，セル I4～I23 に複写する。

垂直照合(b)

(3)　基準 1 での計算値を求めるために次の計算式をセル J3 に入力し，セル J4～

J23 に複写する。

[c] ＊1000

基準 2 での計算値を求めるために次の計算式をセル K3 に入力し，セル K4～K23 に複写する。

[d] ＊1000

(4) 二つの基準によって求めた計算値のうち，小さい方を与信枠とするために次の計算式をセル L3 に入力し，セル L4～L23 に複写する（小さい方の値が負の値である場合，0 とする）。

[e]

(5) ワークシート“与信枠”で用いる関数を表 3 に示す。

表 3 ワークシート“与信枠”で用いる関数

書式	説明
垂直照合(式,範囲,列の位置,検索の指定)	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，範囲の左端列から列を 1，2，… と数え，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。

a，b に関する解答群

ア D3,信用度評価基準!$D$2～$E$6,2,1

イ D3,信用度評価基準!$D$2～$F$6,3,1

ウ D3,信用度評価基準!$D$7～$E$11,2,1

エ D3,信用度評価基準!$D$7～$F$11,3,1

オ G3,信用度評価基準!$D$2～$E$6,2,1

カ G3,信用度評価基準!$D$2～$F$6,3,1

キ G3,信用度評価基準!$D$7～$E$11,2,1

ク G3,信用度評価基準!$D$7～$F$11,3,1

c，dに関する解答群

ア　(B3－C3)＊垂直照合(D3,信用度評価基準!$D$2～$E$6,2,1)

イ　(B3－C3)＊垂直照合(D3,信用度評価基準!$D$2～$F$6,3,0)

ウ　(B3－C3)＊垂直照合(D3,信用度評価基準!$D$2～$F$6,3,1)

エ　(B3－C3)＊垂直照合(D3,信用度評価基準!$D$7～$F$11,3,0)

オ　(B3－C3)＊垂直照合(G3,信用度評価基準!$D$2～$F$6,3,0)

カ　E3＊垂直照合(D3,信用度評価基準!$D$7～$F$11,3,1)

キ　E3＊垂直照合(G3,信用度評価基準!$D$2～$F$6,3,0)

ク　E3＊垂直照合(G3,信用度評価基準!$D$2～$F$6,3,1)

ケ　E3＊垂直照合(G3,信用度評価基準!$D$7～$F$11,3,1)

コ　F3＊垂直照合(G3,信用度評価基準!$D$7～$F$11,3,0)

eに関する解答群

ア　IF(最小(J3～K3)＜0,0,最小(J3～K3))

イ　IF(最大(J3～K3)＜0,0,最大(J3～K3))

ウ　最小(J3～K3)

エ　最大(J3～K3)

設問2　取引先の与信管理に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔ワークシート：支払いサイト〕

表2の信用度に基づく支払いサイトに対応した図3のワークシート“支払いサイト”を作成した。セル B2～F6 に決められた支払いサイトの月数を入力する。ここで，取引停止“－”に対しては数値0を入力する。

	A	B	C	D	E	F
1	基準1＼基準2	取引停止	慎重	現状維持	積極	最優先
2	取引停止	0	0	0	0	0
3	慎重	0	0	1	2	2
4	現状維持	0	1	2	2	2
5	積極	0	2	2	3	3
6	最優先	0	2	2	3	3

図3　ワークシート“支払いサイト”

〔ワークシート：与信管理〕

取引先の信用度評価結果に基づく支払いサイトの決定及び与信枠に基づく当月受注の可否の判定を行うため，図2のワークシート“与信枠”にM列以降を追加して，図4のワークシート“与信管理”を作成した。

	A	…	H	I	…	L	M	N	O	P	Q	R	S
1	取引先	…	信用度		…	与信枠(千円)	支払いサイト(月数)	前月売上(千円)	2か月前売上(千円)	3か月前売上(千円)	受注可否判定		
2		…	基準1	基準2	…			1	2	3	前月売掛残(千円)	当月注文額(千円)	判定結果
3	K1	…	現状維持	慎重	…	12,000	1	2,438	2,212	2,543	2,438	2,611	○
4	K2	…	現状維持	積極	…	269,000	2	121,806	75,058	29,972	196,864	50,346	○
5	K3	…	慎重	現状維持	…	87,500	1	23,169	36,507	31,572	23,169	47,483	○
6	K4	…	取引停止	現状維持	…	0	0	0	0	0	0	0	×
7	K5	…	現状維持	積極	…	292,000	2	93,930	1,894	75,405	95,824	50,346	○
⋮	⋮	…	⋮	⋮	…	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	K17	…	慎重	積極	…	200,000	2	60,896	46,210	82,942	107,106	98,770	×
20	K18	…	慎重	現状維持	…	53,500	1	24,811	19,113	14,800	24,811	93,530	×
21	K19	…	慎重	現状維持	…	26,500	1	7,979	3,333	5,221	7,979	2,403	○
22	K20	…	現状維持	現状維持	…	207,000	2	77,803	30,729	38,580	108,532	81,151	○
23	K21	…	慎重	慎重	…	11,800	0	5,216	4,610	2,675	0	10,400	○

図4　ワークシート“与信管理”

(1)　信用度の評価結果の組合せから決定される支払いサイトを求めるために次の計算式をセルM3に入力し，セルM4～M23に複写する。

f

(2) セルN3～P23に，取引先ごとの前月～3か月前の売上を入力する。セルN2～P2に，売上月に対する当月からさかのぼった月数を入力する。

(3) 前月までの売上金額に対して支払いサイトに応じて回収した結果の売掛金の残額（以下，前月売掛残という）を求めるために，次の計算式をセルQ3に入力し，セルQ4～Q23に複写する。ここで，各取引先に対する売掛金回収は，前倒し及び延滞はなく，支払いサイトどおりに行われているものとする。

g

(4) 当月注文額をセルR3～R23に入力する。

(5) 前月売掛残と当月注文額の合計と与信枠を比較し，受注可否の判定を行う。前月売掛残と当月注文額の合計が，与信枠を超える場合又は与信枠が0の場合，受注不可とする。受注可の場合は“○”，受注不可の場合は“×”を表示する次の計算式をセルS3に入力し，セルS4～S23に複写する。

h

(6) 表3の関数に加えて，ワークシート“与信管理”に用いる可能性のある関数を，表4に示す。

表4 ワークシート“与信管理”で用いる可能性のある関数

書式	説明
水平照合(式,範囲,行の位置,検索の指定)	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，範囲の上端行から行を1，2，… と数え，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。
表引き(範囲,行の位置,列の位置)	範囲の左上端から行と列をそれぞれ1，2，… と数え，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。
照合一致(式,範囲,検索の指定)	1行又は1列を対象とする範囲に対して，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。範囲の左端又は上端から1，2，… と数え，範囲の中で見つかったセルの位置の数を返す。 ・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，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，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。
条件付合計(検索の範囲,検索条件の記述,合計の範囲)	行数及び列数がともに同じ大きさの検索の範囲及び合計の範囲に対して，検索と合計を行う。検索の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すべて探す。検索条件の記述を満たした各セルを左上端から数えた位置と，合計の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。

fに関する解答群

ア　照合一致(支払いサイト!$B$2～$F$6,垂直照合(H3,支払いサイト!$A$2～$A$6,3,0),水平照合(I3,支払いサイト!$B$1～$F$1,3,0))

イ　照合一致(支払いサイト!$B$2～$F$6,水平照合(H3,支払いサイト!$A$2～$A$6,3,0),垂直照合(I3,支払いサイト!$B$1～$F$1,3,0))

ウ　表引き(支払いサイト!$B$2～$F$6,垂直照合(H3,支払いサイト!$A$2～$A$6,3,0),水平照合(I3,支払いサイト!$B$1～$F$1,3,0))

エ　表引き(支払いサイト!$B$2～$F$6,水平照合(H3,支払いサイト!$A$2～$A$6,3,0),垂直照合(I3,支払いサイト!$B$1～$F$1,3,0))

オ　表引き(支払いサイト!$B$2～$F$6,照合一致(H3,支払いサイト!$A$2～$A$6,0),照合一致(I3,支払いサイト!$B$1～$F$1,0))

カ　表引き(支払いサイト!$B$2～$F$6,照合一致(H3,支払いサイト!$A$2～$A$6,1),照合一致(I3,支払いサイト!$B$1～$F$1,1))

キ　表引き(支払いサイト!$B$2～$F$6,照合一致(I3,支払いサイト!$B$1～$F$1,0),照合一致(H3,支払いサイト!$A$2～$A$6,0))

ク　表引き(支払いサイト!$B$2～$F$6,照合一致(I3,支払いサイト!$B$1～$F$1,1),照合一致(H3,支払いサイト!$A$2～$A$6,1))

gに関する解答群

ア　N3

イ　合計(N3～P3)

ウ　合計(N3～P3)＋R3

エ　条件付合計(N$2～P$2,'＝M3',N3～P3)

オ　条件付合計(N$2～P$2,'＝M3',N3～P3)＋R3

カ　条件付合計(N$2～P$2,'＜M3',N3～P3)

キ　条件付合計(N$2～P$2,'＜M3',N3～P3)＋R3

ク　条件付合計(N$2～P$2,'≦M3',N3～P3)

ケ　条件付合計(N$2～P$2,'≦M3',N3～P3)＋R3

hに関する解答群

ア　IF(Q3＞L3,'×','○')

イ　IF(Q3≧L3,'×','○')

ウ　IF((Q3＋R3)＞L3,'×','○')

エ　IF((Q3＋R3)≧L3,'×','○')

オ　IF(論理積((Q3＋R3)＞L3,L3＝0),'×','○')

カ　IF(論理積((Q3＋R3)≧L3,L3＝0),'×','○')

キ　IF(論理和(Q3＞L3,L3＝0),'×','○')

ク　IF(論理和(Q3≧L3,L3＝0),'×','○')

ケ　IF(論理和((Q3＋R3)＞L3,L3＝0),'×','○')

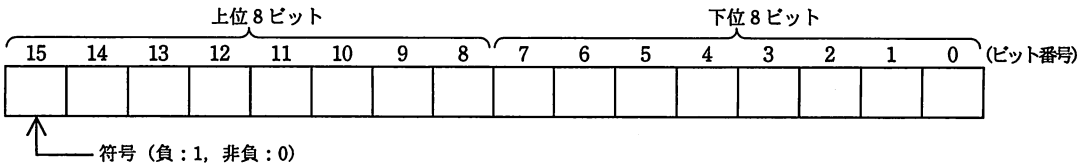
コ　IF(論理和((Q3＋R3)≧L3,L3＝0),'×','○')

■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 r,adr [,x]	(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
論理比較 ComPare Logical	CPL	r1,r2 r,adr [,x]		

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

(4) シフト演算命令

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

(5) 分岐命令

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

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

(6) スタック操作命令

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

(7) コール，リターン命令

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

(8) その他

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

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

1.3 文字の符号表

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

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

2. アセンブラ言語 CASLⅡの仕様

2.1 言語の仕様

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

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

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

2.2 命令の種類

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

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

2.3 アセンブラ命令

アセンブラ命令は，アセンブラの制御などを行う。

(1) START [実行開始番地]

START 命令は，プログラムの先頭を定義する。

実行開始番地は，そのプログラム内で定義されたラベルで指定する。指定がある場合はその番地から，省略した場合は START 命令の次の命令から，実行を開始する。

また，この命令につけられたラベルは，他のプログラムから入口名として参照できる。

(2) | END | |

END 命令は，プログラムの終わりを定義する。

(3) | DS | 語数 |

DS 命令は，指定した語数の領域を確保する。

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

(4) | DC | 定数［,定数］… |

DC 命令は，定数で指定したデータを（連続する）語に格納する。

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

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

2.4　マクロ命令

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

(1) | IN | 入力領域,入力文字長領域 |

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

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

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

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

(2) | OUT | 出力領域,出力文字長領域 |

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

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

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

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

(3) | RPUSH |

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

(4) | RPOP |

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

2.5 機械語命令

機械語命令のオペランドは，次の形式で記述する。

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

2.6 その他

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

(2) 生成された命令語，領域は，主記憶上で連続した領域を占める。

3. プログラム実行の手引

3.1 OS

プログラムの実行に関して，次の取決めがある。

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

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

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

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

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

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

3.2 未定義事項

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

表計算ソフトの機能・用語

表計算ソフトの機能，用語などは，原則として次による。

1. **ワークシート**

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

2. **セル**

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

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

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

(4) データが入力されていないセルを，空白セルという。

3. **セルへの入力**

(1) セルに数値，文字列，計算式を入力できる。

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

(3) セルA1に数値5を入力するときは，“セルA1に5を入力”と表す。

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

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

4. **セルの内容の表示**

(1) セルに数値を入力すると，右詰めで表示される。

(2) セルに文字列を入力すると，左詰めで表示される。

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

(4) セルの内容の表示については，左詰め，中央揃え，右詰めに変更できる。

5. **計算式**

(1) 計算式には，数学で用いられる数式が利用できる。

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

(3) 算術演算子による計算の優先順位は，数学での優先順位と同じである。

6. 再計算

(1) セルに計算式を入力すると，直ちに計算結果を表示する。

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

7. 関数

(1) 計算式には次の表で定義する関数を利用することができる。

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

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

8. セルの複写

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

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

9. 絶対参照

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

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

10. マクロ

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

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

11. その他

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

〔 メ モ 用 紙 〕

7. 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

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

お知らせ

1. システムの構築や試験会場の確保などの諸準備が整えば，平成23年11月からITパスポート試験においてCBT※方式による試験を実施する予定です。
2. CBT方式による試験の実施に伴い，現行の筆記による試験は，廃止する予定です。
3. 詳細が決定しましたら，ホームページなどでお知らせします。

※CBT（Computer Based Testing）：コンピュータを使用して実施する試験。