FE

平成28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2～問7	問8	問9 ～ 問13
選択方法	必須	4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3，問4，問6，問7，問9を選択した場合の例〕

   (3) **選択した問題**については，次の例に従って，**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2～問7について5問以上マークした場合は，はじめの4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。

C　COBOL　Java　アセンブラ　表計算

正　誤　表

平成 28 年 10 月 16 日実施

基本情報技術者試験　午後　問題

ページ	問題番号	行	誤	正	訂正の内容
7	1	下から2行目	3 要素との関連（一部）を表 3 にまとめた。	3 要素との関連を表 3 にまとめた。	下線部分を削除する。
8	1	上から1行目	表 3　情報セキュリティの 3 要素との関連（一部）	表 3　情報セキュリティの 3 要素との関連	下線部分を削除する。

ページ	問題番号	行	誤	正	訂正の内容
40	9	下から1行目	③　目標作業期間は，今回の開発作業を開始する日から，…	③　目標作業期間は，半年ごとの一連の開発作業を開始する日から，…	下線部分を訂正する。
41	9	(6) の上から2行目	遅延日数の合計 wt は，次式で求められる。	遅延日数 wt は，次式で求められる。	下線部分を削除する。

〔問題一覧〕

●問 1（必須問題）

問題番号	出題分野	テーマ
問 1	情報セキュリティ	販売支援システムの情報セキュリティ

●問 2～問 7（6 問中 4 問選択）

問題番号	出題分野	テーマ
問 2	ソフトウェア	コンパイラの字句解析と構文解析
問 3	データベース	従業員の通勤情報を管理する関係データベース
問 4	ネットワーク	Web 画面の表示に要するデータ転送時間
問 5	ソフトウェア設計	レンタル業務システムの設計
問 6	プロジェクトマネジメント	単体テストにおける品質管理
問 7	経営戦略・企業と法務	業務提携と出資の検討

●問 8（必須問題）

問題番号	出題分野	テーマ
問 8	データ構造及びアルゴリズム	数値の編集

●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）	サブシステムの開発作業順序
問 10	ソフトウェア開発（COBOL）	健康診断結果の管理
問 11	ソフトウェア開発（Java）	電卓プログラム
問 12	ソフトウェア開発（アセンブラ）	リスト処理
問 13	ソフトウェア開発（表計算）	改築作業のスケジューリング

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は必須問題です。必ず解答してください。

問1　販売支援システムの情報セキュリティに関する次の記述を読んで，設問1～3に答えよ。

中堅の商社である A 社では，営業員が顧客先で営業活動を行い，自社に戻ってから見積書を作成している。

この度，営業員がタブレット端末（以下，タブレットという）を携帯し，顧客先で要求を聞きながら，タブレットを使って見積書を作成し，その場で顧客に提示できる販売支援システムを構築することにした。

営業員は，タブレットの Web ブラウザからインターネット経由で HTTP over TLS（以下，HTTPS という）によって販売支援システムにアクセスする。このとき，営業員は，社員 ID とパスワードを入力してログインする。

〔販売支援システムの構成〕

(1)　販売支援システムは次のサーバで構成され，A 社のネットワークに設置される。

①　リバースプロキシサーバ（以下，RP サーバという）1台

②　アプリケーションソフトウェアが稼働する Web サーバ2台

③　見積書作成に必要なデータを格納するデータベースサーバ（以下，DB サーバという）1台

(2)　Web サーバ2台はクラスタリング構成にして，1台が故障してもサービスが継続できるようにする。

(3)　DB サーバへのアクセスの監視は，PC と同じ LAN にある監視サーバで行う。

(4)　インターネットから販売支援システムへの通信は，RP サーバを経由して行う。RP サーバは，HTTPS を HTTP に変換し，販売支援システムの他のサーバと，HTTP で通信する。

A社のネットワーク構成を図1に示す。

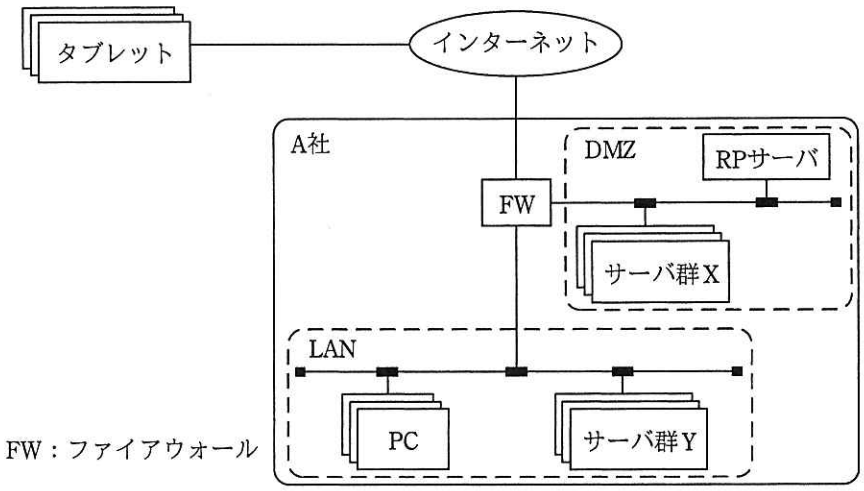


図1　A社のネットワーク構成

販売支援システムに関わる通信経路について，通信経路で利用するプロトコル及び宛先ポート番号を表1に示す。また，FWにおけるフィルタリングの設定を表2に示す。

表1　通信経路で利用するプロトコル及び宛先ポート番号

通信経路	プロトコル	宛先ポート番号
監視サーバからDBサーバ	監視専用	1600
タブレットからRPサーバ	HTTPS	443
RPサーバからWebサーバ	HTTP	80
WebサーバからDBサーバ	DB専用	1552

表2　FWにおけるフィルタリングの設定

項番	条件			動作
	送信元	宛先	宛先ポート番号	
1	任意	RPサーバ	443	許可
2	任意	DBサーバ	1552	許可
3	任意	任意	任意	拒否

注記　項番が小さいものから順に突き合わせ，最初に一致したものが適用される。

設問1　Webサーバと DBサーバについて，図1中の配置場所の組合せとして適切な答えを，解答群の中から選べ。

解答群

	Webサーバ	DBサーバ
ア	サーバ群X	サーバ群X
イ	サーバ群X	サーバ群Y
ウ	サーバ群Y	サーバ群X
エ	サーバ群Y	サーバ群Y

設問2　表2に示した FW におけるフィルタリングの設定では，インターネットから DB サーバに直接アクセスされるおそれがある。そこで，FW のフィルタリングの設定を変更することにした。フィルタリングの設定を変更する内容として適切な答えを，解答群の中から選べ。

解答群

ア　項番1の前に，送信元が“DB サーバ”，宛先が“Web サーバ”，宛先ポート番号が“80”で動作が“許可”という設定を追加する。

イ　項番 1 の前に，送信元が“任意”，宛先が“Web サーバ”，宛先ポート番号が“80”で動作が“許可”という設定を追加する。

ウ　項番2の送信元を“Webサーバ”に変更する。

エ　項番2の動作を“拒否”に変更する。

設問3　A社では，システムを構築する際に，情報セキュリティの3要素である機密性，完全性及び可用性を確保するための対応を整理して，情報セキュリティ担当者の確認を受けることになっている。そこで，次の(i)～(vi)に示す販売支援システムに関する対応と情報セキュリティの3要素との関連（一部）を表3にまとめた。表3中の［　　］に入れる適切な答えを，解答群の中から選べ。

表3　情報セキュリティの3要素との関連（一部）

3要素	対応
機密性	a
完全性	b
可用性	c

〔販売支援システムに関する対応〕

（ⅰ） DBサーバ中のデータの正規化

（ⅱ） RPサーバとWebサーバとの間でのHTTPの利用

（ⅲ） Webサーバのクラスタリング

（ⅳ） コンテンツが改ざんされていないことの定期的な確認

（ⅴ） 社員IDとパスワードによるログイン

（ⅵ） タブレットの利用

a～cに関する解答群

ア （ⅰ）　　イ （ⅱ）　　ウ （ⅲ）

エ （ⅳ）　　オ （ⅴ）　　カ （ⅵ）

次の問２から問７までの６問については，この中から４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，５問以上マークした場合には，はじめの４問について採点します。

問２　コンパイラの字句解析と構文解析に関する次の記述を読んで，設問 1，2 に答えよ。

コンパイラの字句解析では，原始プログラムを文字列として読み込んで，文字列中の文字の並びが字句として認識できるかどうかを解析し，その結果を字句の並びとして出力する。字句とは，原始プログラム中の名前や定数など，構文規則で規定されている文字の並びの最小単位である。構文解析では，字句解析が出力した字句を読み込みながら，字句の並びが構文規則で規定されている文法に合っているかどうかを解析し，その結果を構文木などの中間表現で出力する。

設問１　字句解析の処理について，次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

123. や 123.4e－1 など，構文規則で規定されている符号なし浮動小数点定数を例として字句解析の処理を考える。

“→”は，左側の構文要素が右側で定義されることを示す。

“｜”は，“又は”を意味する。

“[”と“]”で囲まれた部分は，省略可能を意味する。

〔符号なし浮動小数点定数の構文規則〕

符号なし浮動小数点定数	→	小数点定数 [指数部] ｜ 数字列 指数部
小数点定数	→	[数字列] . 数字列 ｜ 数字列 .
指数部	→	e [符号] 数字列
数字列	→	数字 ｜ 数字列 数字
符号	→	＋ ｜ －
数字	→	0 ｜ 1 ｜ 2 ｜ 3 ｜ 4 ｜ 5 ｜ 6 ｜ 7 ｜ 8 ｜ 9

構文規則は，状態遷移図で表現することもできる。符号なし浮動小数点定数の構文規則に対する状態遷移図を，図1に示す。字句解析では，文字の並び中の文字を読み込みながら初期状態から状態を遷移させて，文字の並びを読み終えたときの状態が最終状態ならば，その文字の並びは符号なし浮動小数点定数であると判定する。ここで，円の中の数字は状態番号を示す。初期状態の状態番号は0であり，最終状態は二重円で示している。また，文字の並びは左から右に向けて1文字ずつ処理される。

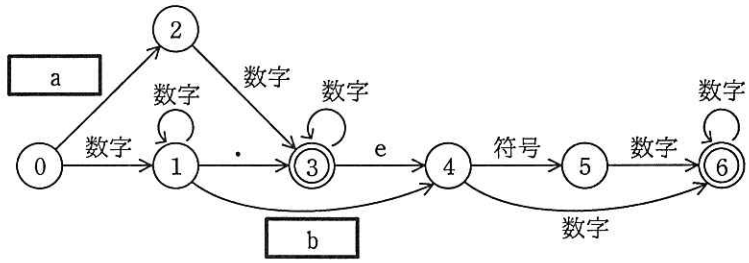


図1　符号なし浮動小数点定数の構文規則に対する状態遷移図

a，bに関する解答群

ア　.　　イ　e　　ウ　指数部

エ　数字　　オ　符号

設問2　構文解析の処理について，次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

構文規則で規定されている式を例として，構文解析の処理を考える。式の構文解析では，式を構成する演算子や名前などの字句を，式の左から右に読み込みながら，字句の並びが構文規則で規定されている文法に合っているかどうかを解析し，その結果を構文木として出力する。例えば，2項演算子 op，名前 v，w，x を構成要素とする式　v op w op x は，次の演算順序①，② になるように解釈され，その結果は，図2に示す2分木で表現する構文木として出力される。図2の構文木では，深さ優先でたどりながら，帰り掛けに節の演算子を評価する。

〔演算順序〕

① v と w に対して演算 op を施す。

② ①の結果と x に対して演算 op を施す。

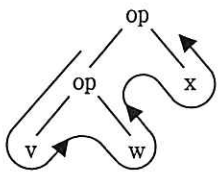


図 2　式の構文木と演算順序の例

式の構文規則では，式の構文を規定するだけではなく，演算子の優先順位も規定する。2 項演算子 op1 と op2，名前 v，w，x，y，z を構成要素とする式の構文規則を定義する。ここで，“演算子 op1 の優先順位は，演算子 op2 の優先順位よりも高い”とする。これを規定する場合，式の構文規則は次のとおりになる。この構文規則で受理される式の例を，例 1 に示す。

〔式の構文規則〕

式　→　項　|　c

項　→　因子 | 項 op1 因子

因子　→　名前

名前　→　v | w | x | y | z

例 1：　v op2 w op1 x

さらに，式の構文に括弧を追加し，“括弧を含む式では，演算の優先順位は，括弧内の演算の方が高い”とする。これを規定する場合，因子の構文規則は次のとおりになる。この構文規則で受理される式の例を，例 2 に示す。

〔因子の構文規則〕

因子 → 名前 | （ [d] ）

例2： v op2 w op1（x op2 y）op1 z

例2で示す式を解析したとき，出力される構文木は [e] となる。

cに関する解答群

ア 式　　イ 式 op2 因子

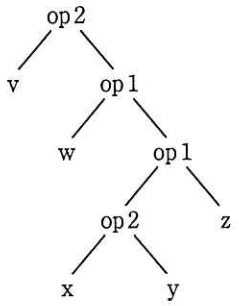
ウ 式 op2 項　　エ 式 op2 名前

dに関する解答群

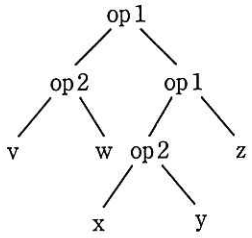
ア 因子　　イ 項　　ウ 式　　エ 名前

eに関する解答群

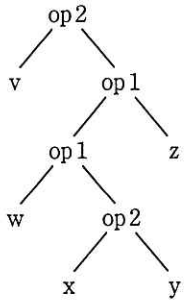
ア



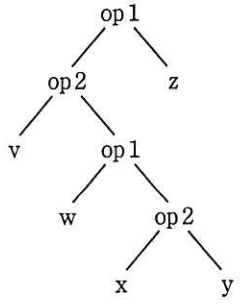
イ



ウ



エ



問３　従業員の通勤情報を管理する関係データベースに関する次の記述を読んで，設問１～４に答えよ。

Ｙ社では，従業員の１か月分の交通費の合計を通勤手当として支給している。交通費は，通勤に公共の交通機関を利用している場合は通勤経路の各区間運賃であり，自家用車を利用している場合は燃料費などの諸経費である。

通勤手当は図１に示す表で管理していたが，より詳細に情報を管理するために，図２のとおり変更した。下線付きの項目は，主キーを表す。

従業員表

従業員番号	氏名	住所	…	通勤手当
00980125	情報太郎	東京都文京区○○○	…	10650
⋮	⋮	⋮	⋮	⋮

図１　変更前の表構成とデータ格納例

従業員表

従業員番号	氏名	住所	…
00980125	情報太郎	東京都文京区○○○	…
⋮	⋮	⋮	⋮

通勤費表

従業員番号	交通機関コード	交通費
00980125	B02	4800
00980125	S01	5850
⋮	⋮	⋮

交通機関表

交通機関コード	交通機関名
B01	○○バス
B02	△△バス
⋮	⋮
C01	自家用車
⋮	⋮

図２　変更後の表構成とデータ格納例

設問1　変更後の表を用いて通勤手当に関するデータを集計する。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。

従業員ごとの通勤手当を求めるには，［ a ］グループ化して，集合関数［ b ］を用いればよい。また，交通機関ごとの利用者数を求めるには，［ c ］グループ化して，集合関数［ d ］を用いればよい。

a，cに関する解答群

ア　交通機関表を交通機関コードで　　イ　従業員表を従業員番号で

ウ　通勤費表を交通機関コードで　　エ　通勤費表を従業員番号で

b，dに関する解答群

ア　AVG　　イ　COUNT　　ウ　MAX　　エ　SUM

設問2　通勤にバスを利用している従業員の従業員番号と交通機関名を表示する。ここで，交通機関コードは3文字の固定長文字列であり，バスの交通機関コードだけが文字“B”で始まる。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 通勤費表.従業員番号, 交通機関表.交通機関名
   FROM 通勤費表, 交通機関表
   WHERE 通勤費表.交通機関コード = 交通機関表.交通機関コード AND
         [ e ]
```

eに関する解答群

ア　通勤費表.交通機関コード IN ('B00', 'B99')

イ　通勤費表.交通機関コード LIKE 'B%'

ウ　通勤費表.交通機関コード LIKE 'B_'

エ　通勤費表.交通機関コード LIKE '_B%'

設問３　従業員の通勤圏内に新しい路線が開通することになった。この影響を確認するために，通勤経路が変わる可能性がある従業員の従業員番号を抽出することにした。同じ従業員番号は一つだけ表示する。新しい路線が影響する交通機関名は“情報都市線”と“駒込バス”であり，いずれも同名の交通機関名は他にないものとする。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 通勤費表.従業員番号
  FROM 通勤費表
  WHERE [          f          ]
```

f に関する解答群

ア
```
通勤費表.交通機関コード = (
  SELECT 交通機関表.交通機関コード FROM 交通機関表
    WHERE 交通機関表.交通機関名 IN ('情報都市線', '駒込バス'))
```

イ
```
通勤費表.交通機関コード IN (
  SELECT 交通機関表.交通機関コード FROM 交通機関表
    WHERE 交通機関表.交通機関名 IN ('情報都市線', '駒込バス'))
```

ウ
```
通勤費表.交通機関コード IN (
  SELECT 交通機関表.交通機関コード FROM 交通機関表
    WHERE 交通機関表.交通機関名 IN ('情報都市線', '駒込バス'))
GROUP BY 通勤費表.交通機関コード
```

エ
```
通勤費表.交通機関コード IN (
  SELECT 交通機関表.交通機関コード FROM 交通機関表
    WHERE 交通機関表.交通機関名 IN ('情報都市線', '駒込バス'))
GROUP BY 通勤費表.従業員番号
```

設問４　Y 社では，毎年 4 月に交通安全講習会を開催しており，通勤に自家用車（交通機関コード“C01”）を利用している従業員には受講を義務付けている。通勤に自家用車を利用していない従業員の受講は任意である。交通安全講習会を受講した従業員は，図 3 に示す講習会受講表で管理している。受講していない従業員のレコードは存在しない。

通勤に自家用車を利用している従業員のうち，2016 年 4 月 20 日に開催された交通安全講習会を受講していない従業員の従業員番号を表示する。次の SQL 文

の［　　］に入れる正しい答えを，解答群の中から選べ。

講習会受講表

従業員番号	受講日
00980125	20160420
⋮	⋮

図3　講習会受講表とデータ格納例

```
SELECT 通勤費表.従業員番号
   FROM 通勤費表
   WHERE 通勤費表.交通機関コード = 'C01' AND
         [        g        ]
```

gに関する解答群

ア　通勤費表.従業員番号 = (SELECT 講習会受講表.従業員番号
　　FROM 講習会受講表 WHERE 受講日 IS NULL)

イ　通勤費表.従業員番号 IN (SELECT 講習会受講表.従業員番号
　　FROM 講習会受講表 WHERE 受講日 = '20160420')

ウ　通勤費表.従業員番号 IN (SELECT 講習会受講表.従業員番号
　　FROM 講習会受講表 WHERE 受講日 IS NULL)

エ　通勤費表.従業員番号 NOT IN (SELECT 講習会受講表.従業員番号
　　FROM 講習会受講表 WHERE 受講日 = '20160420')

問4　Web 画面の表示に要するデータ転送時間に関する次の記述を読んで，設問に答えよ。

D社は，自社の Webサイトをインターネット上に公開している。

D 社では，Web 画面 1 ページを構成する全データの転送に掛かる時間を，5 秒以内に収めるよう基準を定めている。個々のデータの転送に掛かる時間は，D 社が定めたデータ転送時間計算モデルを基に算出する。

D社が定めたデータ転送時間計算モデルは，次のとおりである。

(1)　ターンアラウンドタイム（ブラウザ，サーバ間で短いメッセージが往復するのに掛かる時間）t＝0.005（秒），実効転送速度（サーバ，ブラウザ間におけるデータ転送の速さ）e＝1×10^6（バイト／秒）とする。

(2)　ブラウザはキャッシュ機能をもっている。Web 画面を表示する際に，必要なデータがブラウザにキャッシュされているときは，サーバに当該データの再転送の要否確認を要求する。再転送が必要な場合は，応答として当該データが転送される。

(3)　ブラウザが，サーバに対して要求を送信し始めてから，要求に対する結果の受信が完了するまでに掛かる時間を，表 1 に示す。

(4)　表 1 に関わる通信処理は逐次実行され，複数の通信処理が並行して行われることはない。また，表 1 で示した時間以外は無視する。

表 1　要求に対する結果の受信が完了するまでに掛かる時間

データの状態	要求	結果	時間（秒）
キャッシュされていない	データの転送	当該データ（v バイト）の受信	3×t＋v÷e
キャッシュされている	データの再転送の要否確認	＜再転送が不要な場合＞ 再転送が不要であるとの応答	t
		＜再転送が必要な場合＞ 当該データ（v バイト）の受信	3×t＋v÷e

D 社では，自社が取り扱う商品の情報を画像付きで一覧表示する機能（以下，一覧表示機能という）を，Web サイトに付加することにした。

一覧表示機能を実現するに当たり，商品情報を一覧表示する Web 画面（以下，一

覧表示画面という）1 ページで表示する商品情報の最大数を，一覧表示画面を構成するデータの転送に掛かる時間を基に決定したい。一覧表示画面は，表示する商品の数にかかわらずに必要になるデータ（以下，固定データという）と，表示する商品の数に応じてデータサイズや個数が変動するデータ（以下，変動データという）で構成される。

固定データは，JavaScript ファイルやスタイルシート，画面の装飾に使う画像データなど合計 100 個あり，固定データ 1 個の平均サイズは 5×10^3 バイトである。

変動データは，一つの HTML 文書データと，表示する商品 n 個分の画像データから成る。HTML 文書データのサイズは，$10^4 + 500 \times n$ バイトである。商品の画像データ 1 個の平均サイズは 25×10^3 バイトである。

固定データの全てがブラウザにキャッシュされているとき，それぞれのデータごとにデータの再転送の要否確認をサーバに要求したところ，全てに再転送が不要であるとの応答を受け取った。最初の要求をしてから最後の要求に対する応答を受け取るまでに掛かる時間は［ a ］秒である。

固定データのいずれもがブラウザにキャッシュされていないとき，これらの固定データの総転送時間は［ b ］秒である。このとき，一覧表示画面 1 ページを構成するデータの転送時間が D 社の基準を満たす商品情報の最大数 n は，次式を解くことによって求めることができる。ここで，変動データもブラウザにキャッシュされていないものとする。

［ b ］$+ 3 \times t +$［ c ］$+ (3 \times t +$［ d ］$) \times n \leqq 5$

設問　本文中の［　　］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　0.005	イ　0.015	ウ　0.02	エ　0.5
オ　0.515	カ　1	キ　1.5	ク　2

c，dに関する解答群

ア　$10^4 + 500$

イ　$10^4 + 500 \times n$

ウ　25×10^3

エ　$25 \times 10^3 \times n$

オ　$\dfrac{10^4 + 500}{e}$

カ　$\dfrac{10^4 + 500 \times n}{e}$

キ　$\dfrac{25 \times 10^3}{e}$

ク　$\dfrac{25 \times 10^3 \times n}{e}$

問5　レンタル業務システムの設計に関する次の記述を読んで，設問1～3に答えよ。

E社は，レンタル業務システムを利用して，CD，DVD，書籍などの貸出し用の各種資産（以下，レンタル商品という）をレンタルするサービスを行っている。レンタル業務システムによるレンタル業務の概要は，次のとおりである。

〔レンタル業務の概要〕

(1)　会員登録の手順

①　担当者は，サービスの利用希望者から入会申込書を受け取り，氏名，住所，生年月日及び電話番号を，会員情報としてレンタル業務システムに登録する。

②　レンタル業務システムは，利用希望者に会員IDを割り当て，会員IDが印字された会員カードを発行する。会員情報の属性である登録年月日と最終来店年月日に本日日付を，累計支払額と貸出中延滞数に初期値として0を登録する。ここで，累計支払額は会員登録時からの支払額の累計（延滞料金を含む）であり，貸出中延滞数は現時点で返却予定年月日を過ぎているレンタル商品の件数である。

③　担当者は，レンタル業務システムから発行された会員カードを利用希望者に渡して，会員登録が完了する。

(2)　レンタル商品の貸出し手順

E社は，全てのレンタル商品のそれぞれに，一意となる識別子として資産管理番号を割り当てている。また，同じ種類のレンタル商品の情報（以下，商品情報という）ごとにも一意となる識別子として商品コードを割り当てている。図1は商品名が“オブジェクト指向設計”である3冊の書籍のレンタル商品に対して，資産管理番号を三つ，商品コードを二つ，それぞれ割り当てた例である。

商品コード：商品コード1 商品名：オブジェクト指向設計 著者名：試験太郎 資産管理番号：資産管理番号1	商品コード：商品コード1 商品名：オブジェクト指向設計 著者名：試験太郎 資産管理番号：資産管理番号2	商品コード：商品コード2 商品名：オブジェクト指向設計 著者名：情報花子 資産管理番号：資産管理番号3

図1　資産管理番号と商品コードを割り当てた例

① 会員は，会員カードと貸出しを希望するレンタル商品を担当者に渡す。E 社では，会員が 1 回の貸出し手順で借り出せるレンタル商品は最大 10 件までとしている。

② 担当者は，会員が提示した会員カードの会員 ID をレンタル業務システムに入力して会員情報を画面に表示する。

③ 担当者は，会員が貸出しを希望するレンタル商品に貼付された資産管理番号のバーコードをレンタル業務システムに読み込ませ，希望する貸出期間を会員から聞いて，レンタル業務システムに入力する。ここで，貸出期間は 1 日，2 日，1 週間のいずれかである。この作業を入力操作と呼ぶ。

④ レンタル業務システムは，入力操作が行われる都度，レンタル商品の商品コード，貸出期間，商品名，商品概要，返却予定年月日，貸出料金及び貸出料金合計を画面に表示する。貸出料金はレンタル商品の種類ごとに貸出期間によって設定されている単価から，返却予定年月日は貸出期間と本日日付から自動計算する。

⑤ 担当者は，会員が貸出しを希望するレンタル商品の数だけ入力操作を繰り返す。4 件の入力操作を行った後の画面の表示例を，図 2 に示す。

⑥ 担当者が，会員から貸出料金を受領した後，図 2 に示す確定ボタンを押すと，レンタル業務システムは，貸し出すレンタル商品ごとに貸出明細番号を割り当てて貸出明細の情報として登録する。さらに，返却予定年月日ごとにまとめて，伝票番号を割り当てて貸出の情報として登録する。図 2 の例では，貸出は 3 件，貸出明細は 4 件を登録する。加えて，貸出明細とレンタル商品を関連付けて，レンタル商品の貸出可否状態を“貸出中”にする。同時に会員情報の内容も更新する。その後，伝票番号ごとに貸出伝票を出力する。

⑦ 担当者が，貸出しを希望するレンタル商品と貸出伝票を会員に渡すことで，レンタル商品の貸出しが完了する。

貸出年月日：2016-10-16

商品コード	貸出期間	商品名	商品概要	返却予定年月日	貸出料金
C529157	1日	音楽 CD01	編曲者 A，63分	2016-10-17	130円
D700557	2日	映画 DVD01	監督 B，主演 C，122分	2016-10-18	200円
B905668	2日	UML 基礎編	著者 X，100ページ，ISBN-xxxx	2016-10-18	100円
B905669	1週間	データベース応用編	著者 Y，240ページ，ISBN-yyyy	2016-10-23	150円
				貸出料金合計	580円

確定

図2　4件の入力操作を行った後の画面の表示例

(3)　レンタル商品の返却手順

レンタル商品の返却では，貸出中の全てのレンタル商品が返却される場合と，貸出中のレンタル商品の一部だけが返却される場合とがある。

①　担当者は，会員が返却するレンタル商品を受け取り，レンタル商品に貼付された資産管理番号のバーコードをレンタル業務システムに読み込ませる。

②　レンタル業務システムは，伝票番号，商品コード，商品名及び返却予定年月日を画面に表示する。返却予定年月日を過ぎている場合は，返却されるレンタル商品ごとに延滞料金を表示し，延滞料金の合計も表示する。

③　延滞料金が表示された場合，担当者は，会員から延滞料金を受領する。

④　担当者がレンタル商品の返却の確定ボタンを押すと，レンタル業務システムは，貸出明細に関連付けたレンタル商品の貸出可否状態を“貸出可”にする。同時に貸出明細と会員情報の内容を更新する。これによって，レンタル商品の返却が完了する。

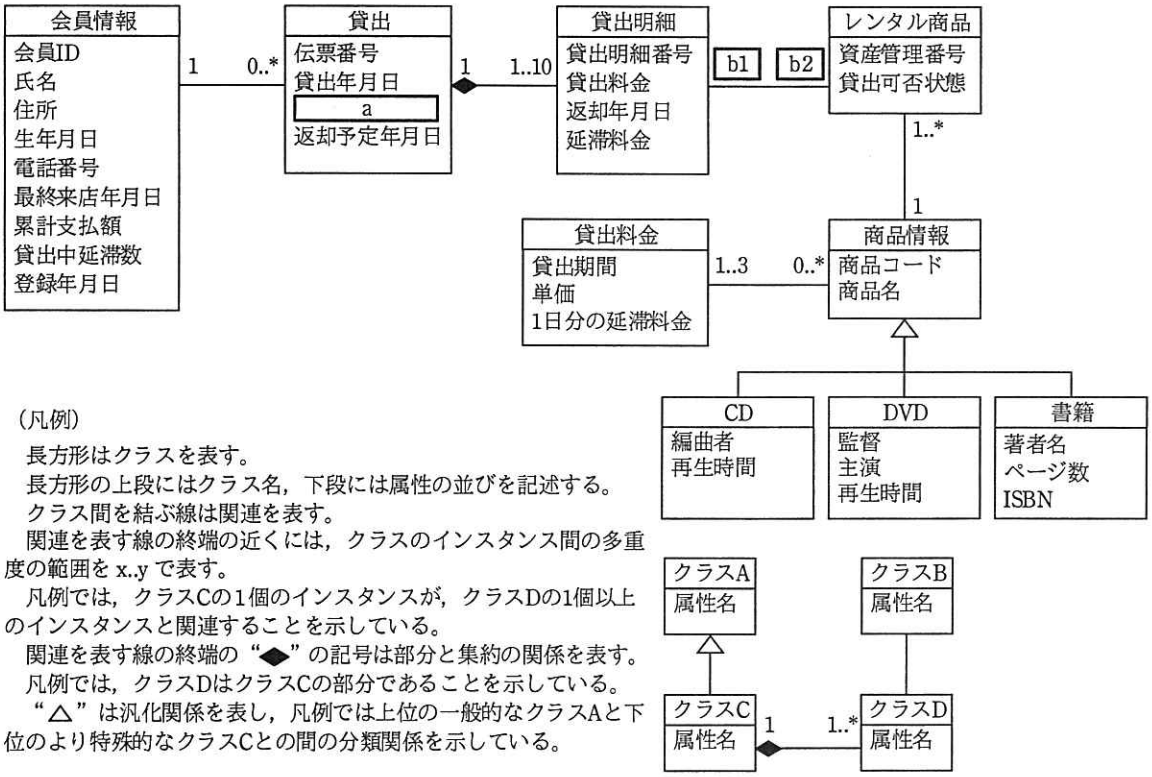
(4)　貸出中のレンタル商品のチェック処理

レンタル業務システムは，閉店後に毎日，貸出中のレンタル商品の延滞チェックを行っており，延滞が発生した場合は，会員情報の貸出中延滞数を更新する。

〔レンタル業務システムの UML 図〕

図3はレンタル業務システムのクラス図，図4はレンタル商品の貸出し手順における③の操作が行われたときのシーケンス図である。

なお，図3のクラス図には，エンティティクラスだけを記載している。



（凡例）

長方形はクラスを表す。

長方形の上段にはクラス名，下段には属性の並びを記述する。

クラス間を結ぶ線は関連を表す。

関連を表す線の終端の近くには，クラスのインスタンス間の多重度の範囲を x..y で表す。

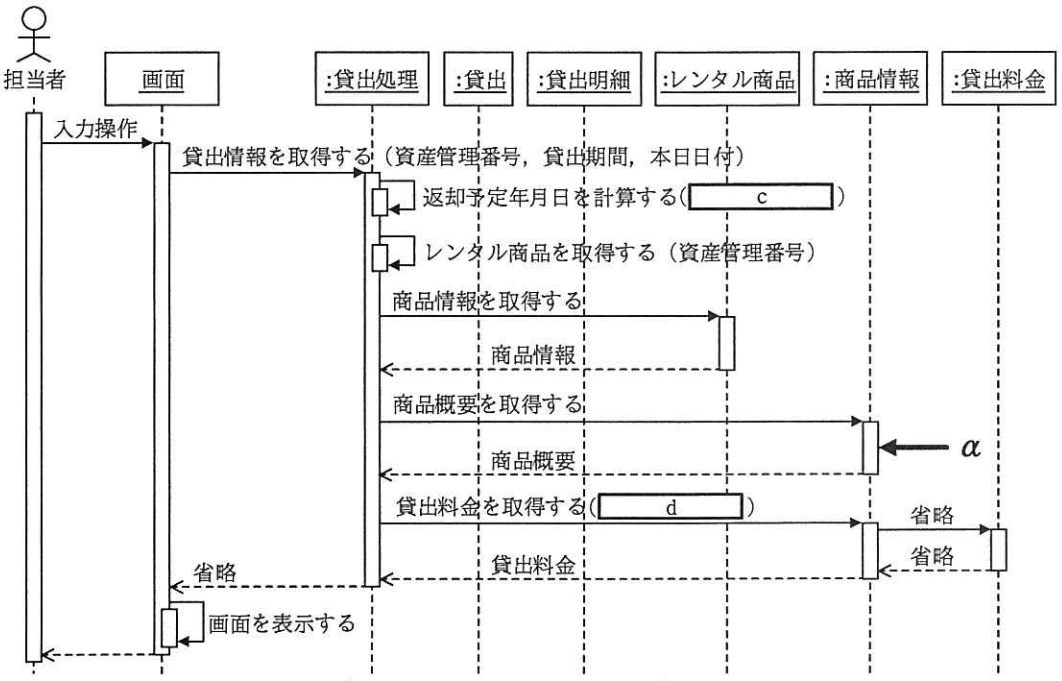
凡例では，クラスCの1個のインスタンスが，クラスDの1個以上のインスタンスと関連することを示している。

関連を表す線の終端の“◆”の記号は部分と集約の関係を表す。

凡例では，クラスDはクラスCの部分であることを示している。

“△”は汎化関係を表し，凡例では上位の一般的なクラスAと下位のより特殊的なクラスCとの間の分類関係を示している。

図3　レンタル業務システムのクラス図



注記　図中の“省略”はメッセージの記述を省略していることを示す。

図4　入力操作が行われたときのシーケンス図

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。ここで，b1 と b2 に入れる答えは，b に関する解答群の中から組合せとして正しいものを選ぶものとする。

a に関する解答群

ア　延滞日数　　イ　延滞料金　　ウ　貸出期間　　エ　貸出料金

オ　登録年月日

b に関する解答群

	b1	b2
ア	0	1
イ	0	1..*
ウ	0..1	1
エ	0..1	1..*
オ	1	0
カ	1	0..*
キ	1..*	1
ク	1..*	1..*

設問2　図4中の［　　］に入れる正しい答えを，解答群の中から選べ。

c，d に関する解答群

ア　貸出期間　　イ　貸出期間，単価

ウ　貸出期間，単価，本日日付　　エ　貸出期間，本日日付

オ　単価，返却予定年月日　　カ　単価，返却予定年月日，本日日付

設問3　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。ここで，e1 と e2 に入れる答えは，e に関する解答群の中から組合せとして適切なものを選ぶものとする。

CD クラス，DVD クラス，書籍クラスは商品情報を特化して実現しているので，商品情報クラスを［ e1 ］している。図 4 中の α の部分は，貸出処理クラスから商品情報クラスに対する商品概要を取得する要求のメッセージである。商品概要とは，CD ならば編曲者と再生時間，DVD ならば監督，主演及び再生時間，書籍ならば著者名，ページ数及び ISBN から生成する当該商品の商品概要を表現する文字列である。商品概要を取得するためのメッセージを受信した商品情報インスタンスは，クラスに応じて異なる処理が実行される。一方，貸出処理クラスはクラスや処理の違いを意識せず，全てを商品情報として取り扱うことができる。この性質を［ e2 ］という。

e に関する解答群

	e1	e2
ア	インヘリタンス	カプセル化
イ	インヘリタンス	ポリモーフィズム
ウ	カプセル化	インヘリタンス
エ	カプセル化	ポリモーフィズム
オ	ポリモーフィズム	インヘリタンス
カ	ポリモーフィズム	カプセル化

問6　単体テストにおける品質管理に関する次の記述を読んで，設問1，2に答えよ。

C社では，販売システムの構築を進めており，現在，単体テストを実施している。販売システムは，C社の情報システム部門によるプロジェクト管理の下で，ベンダL社が構築している。ベンダL社は，C社のシステムを構築するプロジェクトに，今回初めて参画している。

C社の品質管理基準では，テストケースの網羅性を示すテスト密度と，当該プログラムにおけるバグ摘出率という指標を用いてプログラムの品質を評価し，単体テストの完了を判断している。

設問1　C社の品質管理基準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。ここで，a1～a3に入れる答えは，aに関する解答群の中から組合せとして適切なものを選ぶものとする。

〔単体テスト完了の判断基準〕

単体テストの完了を判断する際に用いる指標の算出方法とその標準値を表1に示す。指標の標準値については，C社のプログラム開発における過去の同一形態の実績値を基に定めている。また，テスト密度は標準値の80%以上，バグ摘出率は標準値の80%以上かつ120%以下を単体テストの完了の基準範囲とする。

表1　単体テストの完了の指標の算出方法と標準値

指標	算出方法	標準値
テスト密度	テストケース数（件）÷ 当該プログラムの開発規模（kステップ）	100
バグ摘出率	プログラムのバグ数（件）÷ 当該プログラムの開発規模（kステップ）	5

単体テストの完了は，表1に基づいて算出した，プログラムごとのテスト密度とバグ摘出率を，図1に示す品質評価のグラフにプロットして判断する。

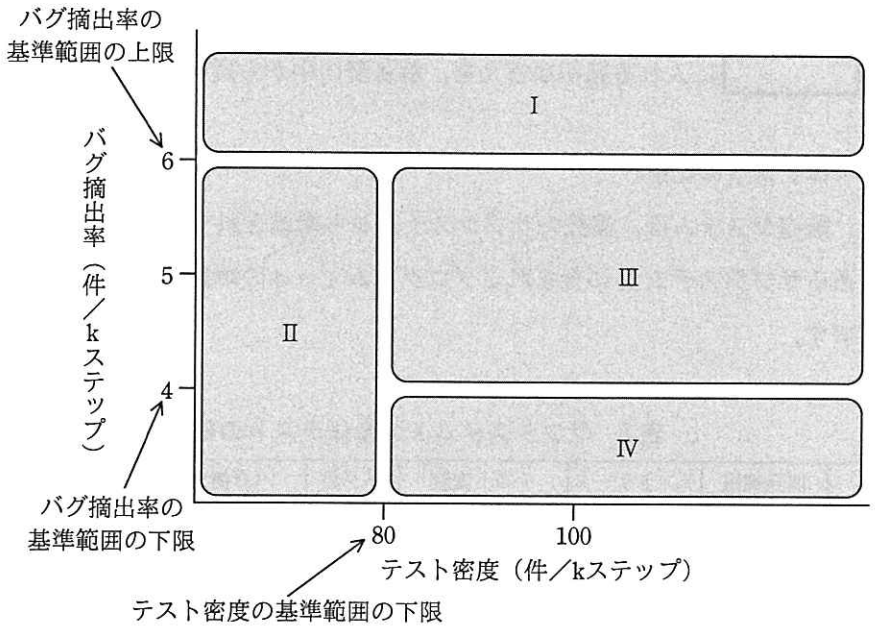


図１　品質評価のグラフ

図１の区分Ⅰ～Ⅳにおける単体テストの品質評価は次のとおりである。

Ⅰ：バグ摘出率が基準範囲の上限を超えているので，［ a1 ］である。

Ⅱ：テスト密度が基準を満たしていないので，テスト不足である。

Ⅲ：テスト密度，バグ摘出率とも基準を満たしているので，［ a2 ］である。

Ⅳ：テスト密度は基準を満たしているが，バグ摘出率が基準範囲の下限に満たないので，［ a3 ］。

aに関する解答群

	a1	a2	a3
ア	品質不良	品質良好	テスト不足である
イ	品質良好	品質不良	テスト不足である
ウ	品質不良	品質良好	テストの内容と摘出したバグの内容から品質を評価する
エ	品質良好	品質不良	テストの内容と摘出したバグの内容から品質を評価する

設問2　販売システムの単体テストの結果と改善策の実施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。

〔単体テストの結果〕

販売システムは，複数のサブシステムから構成されており，そのうちの一つであるサブシステムXに含まれるプログラム1～4の単体テストの結果を，表2に示す。

表2　サブシステムXの単体テストの結果

	開発規模（kステップ）	テストケース数（件）	テスト密度（件／kステップ）	バグ数（件）	バグ摘出率（件／kステップ）	評価
プログラム1	10	980	［b］	70		バグ摘出率が基準範囲の上限を超えた。
プログラム2	2	215		9		
プログラム3	6	750		20		
プログラム4	8	600		30		［c］。

注記　網掛けの部分は表示していない。

表2の結果から，プログラム1，プログラム3及びプログラム4は，“品質良好”と判断できないので，改善策を実施することにした。

〔改善策の実施〕

プログラム1は，バグ摘出率が基準範囲の上限を超えているので，バグの原因分析を行った。主な原因は，詳細設計書に曖昧な記述があり，プログラムで実現すべき機能に誤りが発生したことであった。そこで，詳細設計書の曖昧な記述を修正後，［　d　］。さらに，修正したプログラムについて，必要なテストケースを追加した上で，再度単体テストを実施し，バグ摘出率が基準を満たしていることを確認した。また，同一の観点で，他のプログラムに関しても点検を実施し，同様の問題が含まれていないことを確認した。

プログラム3については，［　e　］ことを確認し，テストケースの内容自体に問題がないことから，現在のバグ摘出で十分な品質が確保されていると判断したので，改善策は不要とした。

プログラム4については，［　c　］ことから，テストケースが不足してい

ることが懸念されるので，テストケースの網羅性を点検した上で，［ f ］，その結果から品質を再評価することにした。

bに関する解答群

ア　7　　イ　98　　ウ　980　　エ　9,800

cに関する解答群

ア　テスト密度は基準を満たしているが，バグ摘出率は基準範囲の下限に満たない

イ　テスト密度は基準を満たしているが，バグ摘出率は基準範囲の上限を超えている

ウ　バグ摘出率，テスト密度とも基準を満たしていない

エ　バグ摘出率，テスト密度とも基準を満たしている

dに関する解答群

ア　プログラムのソースコードが詳細設計書を正確に反映していることを点検した

イ　プログラムのソースコードに関する記述規定どおりに，プログラムが記述されていることを点検した

ウ　プログラムのソースコードに文法上の誤りがないことを点検した

eに関する解答群

ア　テストケースが特定の処理の流れを重点的に確認するように作成されている

イ　テストケースが全ての処理の流れを網羅的に確認できるように作成されている

ウ　テストケースについて，実データを使用した環境で確認している

エ　テストケースを実行する手順について，正しく定められている

fに関する解答群

ア　摘出したバグが修正されていることを確認し

イ　テストの実施が不足している処理の流れに対してテストケースを追加して再度単体テストを実施し

ウ　テストの実施方法に問題がないことを確認し

エ　バグを摘出した処理の流れに対して，テストケースを追加して再度単体テストを実施し

問7　業務提携と出資の検討に関する次の記述を読んで，設問1～3に答えよ。

消費財メーカのB社は，同業のT社との物流業務での提携を考えており，そのためにT社への出資を検討している。B社では，T社への出資額を算定するための準備として，T社の財務状況を調査して，企業価値を算出することになった。表1～3は，B社が入手したT社の2016年度の損益計算書，貸借対照表及びキャッシュフロー（以下，CFという）計算書の予測である。

表1　T社の2016年度の損益計算書の予測（単位：億円）

売上高	120
売上総利益	40
営業利益	7
経常利益	9
税引き前純利益	10
税引き後純利益	7

表2　T社の2016年度の貸借対照表の予測（単位：億円）

流動資産	50
固定資産	100
流動負債	40
固定負債	30
純資産	80

表3　T社の2016年度のCF計算書の予測（単位：億円）

営業活動によるCF	15
投資活動によるCF	−5
財務活動によるCF	−3

設問1　表 1～3 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。

B 社では，出資する相手企業が次の条件（以下，出資条件という）のうち二つ以上を満たすことを，出資の検討を進めるための要件としている。

(1)　売上高営業利益率が5%以上である。

(2)　総資産経常利益率が5%以上である。

(3)　営業活動による CF と投資活動による CF の和（以下，FCF という）が負でない。

表1～3から，T社は出資条件の［　a　］を満たしている。この結果から，B社では，T社への出資の検討を進めることにした。

また，B 社は，T 社の営業活動による CF が正で，投資活動による CF と財務活動による CF が負であることから，T 社は［　b　］を進めている企業と考えた。

B社では，物流業務での提携に加えて，T社の情報システムの一部を廃止してB社の情報システムを利用することを検討している。それによって T 社のシステム運用保守費を削減し，T 社の販売費及び一般管理費（以下，販管費という）を圧縮できると想定しており，T 社の売上高や総資産が変わらなければ［　c　］上げられると考えている。

a に関する解答群

ア　(1)と(2)の二つだけ　　イ　(1)と(3)の二つだけ

ウ　(2)と(3)の二つだけ　　エ　(1)～(3)の全て

b に関する解答群

ア　本業で得た利益に加えて，銀行からの借入れを増やして投資

イ　本業で得た利益に加えて，手持ちの資産を現金化して債務返済や株主還元

ウ　本業で得た利益を投資に回すとともに，債務返済や株主還元

cに関する解答群

ア　FCFだけは

イ　売上高営業利益率と総資産経常利益率だけは

ウ　売上高営業利益率，総資産経常利益率及びFCFの全てを

設問2　T社の企業価値の算出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。

B社では，各年度のFCFの現在価値を合計することで企業価値を算出している。そこで，B社は，T社の企業価値を算出する準備として，T社の今後のFCFを次のように予測した。ここで，T社との業務提携と出資は2017年度の初日に行われるものとする。

(1)　B社との業務提携によって，初年度（2017年度）に10%増加する。

(2)　2018年度から2020年度まで年率5%で増加する。

(3)　2021年度から2023年度まで年率3%で増加する。

(4)　2024年度からは一定である。

表4は，これらに基づくT社の各年度のFCFの予測である。

表4　T社の各年度のFCFの予測（単位：億円）

年度	2017	2018	2019	2020	2021	2022	2023	…
FCF		d			13.12	13.51	13.91	…

注記　網掛け部分は表示していない。

B社では，企業価値の算出に，割引率を使った現在価値の考え方を用いている。この考え方によると，割引率をrとして1年複利で計算し，n年後のFCFをCで表すとき，その現在価値は$C/(1+r)^n$と表せる。B社は，T社の企業価値を算出するために，各年度のFCFについて割引率を0.1として現在価値を計算した。2017年度のFCFの2016年度末時点の現在価値は，［　e　］である。ここで，各年度のFCFは各年度末に発生すると考える。

T 社の企業価値は，B 社と業務提携することによって大きくなると予想される。

誤って，割引率を考慮せずに各年度の FCF をそのまま合計して，B 社で定義している企業価値よりも［ f ］計算してしまわないように，B 社では定期的に担当者教育をしている。

d，e に関する解答群

ア 9.09	イ 9.84	ウ 10.00	エ 10.03	オ 10.50
カ 10.82	キ 11.00	ク 11.03	ケ 11.55	コ 12.10

f に関する解答群

ア 大きく　　イ 小さく

設問３　B 社では，様々な条件を変更して T 社の企業価値（各年度の FCF の現在価値の合計）がどのように変化するかを検証した。条件を変更する前に比べて T 社の企業価値が大きくなるものを，解答群の中から選べ。

解答群

ア B 社が T 社の企業価値を算出するときに使っている割引率を 0.12 に上げる。

イ T 社の 2016 年度の銀行からの借入れの予測額を減らす。

ウ T 社の 2016 年度の設備投資の予測額を増やす。

エ T 社の 2016 年度の販管費の予測額を減らす。

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１～３に答えよ。

事務計算においては，数値を見やすく表示（印字）するために，例えば３桁ごとに区切りの“,”を挿入するなどの編集処理がよく行われる。

関数 Edit は，指定された編集パターンに従って，数値を編集するプログラムである。表１に，関数 Edit を用いた編集例を示す。例１では，３桁ごとに区切りの“,”を挿入している。例２では，例１の編集に加え，上位の空いた桁を“*”で埋めている。例３では，数値の右端から２桁目と３桁目の間に“.”を挿入している。

表１　関数 Edit を用いた編集例

		編集パターン		
		例１	例２	例３
		“␣□□,□□□”	“*□□,□□□”	“␣□□▮.□□”
数値	123	“␣␣␣123”	“***123”	“␣␣1.23”
	1234	“␣␣1,234”	“**1,234”	“␣␣12.34”
	12345	“␣12,345”	“*12,345”	“␣123.45”

ここで，編集パターン中の文字“□”及び“▮”は，数字と対応付けされた制御文字を表している。また，“␣”は空白文字を表している。

〔プログラムの説明〕

(1) 関数 Edit は，次の形式で呼び出され，二つの引数をもつ。

関数: Edit(文字型: Pattern[], 文字型: Value[])

Pattern[] には，編集パターンの文字列が格納されている。Value[] には，編集する数値を表す文字列が格納されている。各配列の添字は，0 から始まる。文字列 Pattern[] の i 番目の文字は Pattern[i − 1] と表記する。文字列 Value[] についても同様である。

(2) Pattern[] は，1 文字以上から成る文字列であって，表示可能な図形文字及び制御文字（“□” 及び “■”）から構成される。

(3) Value[] は，数値を表す文字列であって，数字 “0” ～ “9” の並びの後に，数値が正又は 0 なら “+” を，負なら “-” を付加した形式である。数字の個数は，Pattern[] 中の文字 “□” 及び “■” の個数と一致するように，必要であれば前方に “0” を付加する。例えば，Pattern[] の内容が “*□□,□■□” のとき，Value[] には，数値が 123 なら “00123+”，0 なら “00000+”，−123 なら “00123-” を指定する。

(4) 関数 Edit は，Value[] で与えられた数値を Pattern[] に従って編集し，編集結果で Pattern[] を置き換える。

〔編集方法〕

Pattern[] 中の各文字について，先頭から順に 1 文字ずつ，次の ① ～ ③ のいずれか一つの操作を実行していく。

① 関数 Edit が呼び出されたときの Pattern[] 中の先頭の文字（以下，fill 文字という）で置き換える。

② Value[] 中の対応する桁の数字で置き換える。

③ 置き換えないで，そのまま残す。

(5) 論理型変数 signif は，on 又は off の値を取る。この変数の実行開始時の値は off であり，Value[] 中に最上位から “0” が連続した後に “0” でない数字が見つかると on になる，などの使い方をする。

(6) 関数 Edit が呼び出されるとき，各引数には正しい値が設定されているものとする。

〔プログラム〕

○関数: Edit(文字型: Pattern[], 文字型: Value[])
○文字型: fill
○論理型: signif
○整数型: p, v

```
・fill ← Pattern[0]
・signif ← off
・v ← 0
■ p: 0, p < Length(Pattern[]), 1    /* Length()は引数の文字列長を返す */
  ▲ Pattern[p] = "□"  or Pattern[p] = "■"  /* 表2のケース1～7の処理 */
    ┌ 現在の変数・配列要素の内容が，表2のケース1～7の ┐
    │ どれに該当するかを決定し，そのケースに従って   │
    │ Pattern[p] と signif の更新処理を行う。        │
    └ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┘
    ・v ← v + 1
  ──                          /* 表2のケース8，9の処理 */
    ▲ signif = off
      ・Pattern[p] ← fill
```

表2　現在の変数・配列要素の内容に応じた更新処理

	現在の変数・配列要素の内容				更新処理	
ケース	Pattern[p]	signif	Value[v]	Value[v+1]	Pattern[p]	signif
1	“□”	off	“0”		fill 文字	off
2	“■”	off	“0”	“+” 以外	fill 文字	on
3	“■”	off	“0”	“+”	fill 文字	off
4	“□” 又は “■”	off	“1” ～ “9”	“+” 以外	Value[v]	on
5	“□” 又は “■”	off	“1” ～ “9”	“+”	Value[v]	off
6	“□” 又は “■”	on	“0” ～ “9”	“+” 以外	Value[v]	on
7	“□” 又は “■”	on	“0” ～ “9”	“+”	Value[v]	off
8	“□” と “■” 以外	off			fill 文字	off
9	“□” と “■” 以外	on			そのまま残す	on

注記　網掛け部分は，内容を判定しない。

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。

引数 Pattern[] 及び Value[] に幾つかのデータを与えて，関数 Edit を実行した結果を，表3に示す。

表3　関数 Edit の実行結果

実行前の内容		実行後の内容
Pattern[]	Value[]	Pattern[]
“␣␣□□,□□□”	“01234+”	“␣␣␣1,234”
“*□□,□□□#”	“00000+”	［　a　］
“*□□□.□□#”	“00012-”	［　b　］
“*□□■.□□#”	“00012+”	［　c　］

aに関する解答群

ア　“*******#”　　イ　“********”

ウ　“******0#”　　エ　“******0*”

b，cに関する解答群

ア　“*****12#”　　イ　“*****12*”

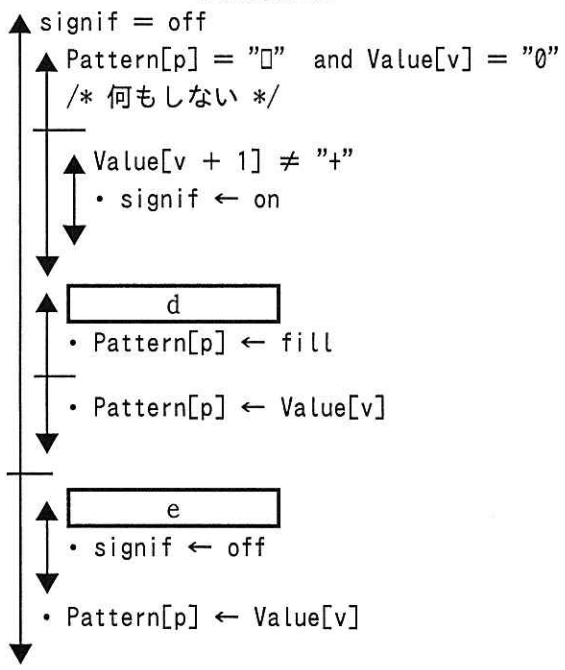
ウ　“****.12#”　　エ　“****.12*”

オ　“***0.12#”　　カ　“***0.12*”

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。

プログラム中の破線で囲んだ部分の処理（表2のケース1～7の処理）を，詳細なプログラムとして記述すると，次のようになる。

〔プログラム中の［　　］部分の処理〕



d，eに関する解答群

ア　"1" ≦ Value[v] and Value[v] ≦ "9"　　イ　Value[v] = "0"

ウ　Value[v + 1] = "-"　　エ　Value[v + 1] = "+"

オ　Value[v + 1] ≠ "-"　　カ　Value[v + 1] ≠ "+"

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。

関数 Edit では，例えば，fill 文字を“␣”とする編集パターンを指定することによって，数値が正なら“␣␣1,234␣”，負なら“␣␣1,234-”と編集することができる。表 2 のケース 1～7 のうち，数値が正なら数値の後に続く文字を fill 文字で置き換えるために用意されたケースは［ f ］である。

fに関する解答群

ア　2，4及び7　　イ　3，5及び7

ウ　4及び7　　エ　5及び7

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社では，半年ごとに社内システムの見直しを行い，4月と10月に新たに開発又は改修するサブシステム（以下，対象サブシステムという）を決定し，開発又は改修作業（以下，開発作業という）を実施している。対象サブシステムは複数あり，それぞれに対して目標作業終了日が設定される。二つ以上の対象サブシステムの開発作業を同時に実施することはない。各対象サブシステムについて，開発リソースの制約があるので，必ずしも目標作業終了日までに開発作業を終了できるとは限らない。しかし，開発作業の終了が目標作業終了日よりも遅れる日数（以下，遅延日数という）の合計はできるだけ少なくしたい。そこで，対象サブシステムの開発作業順序を求める関数job_schedulingを作成した。

(1) 関数job_schedulingの仕様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。

機能：　対象サブシステムの開発作業順序を求める。

引数：　num_s　　対象サブシステム数

　　job　　対象サブシステム情報（JOB型の配列）

　　job_sch　開発作業順序（int型の配列）

(2) 構造体を使ったJOB型の定義は次のとおりである。

```
typedef struct {
    char pr_code[9];     /* 開発コード */
    int  job_term;       /* 開発作業日数 */
    int  target_term;    /* 目標作業期間 */
} JOB;
```

① 開発コードは，対象サブシステムごとに一意となる長さが8の文字列

② 開発作業日数は，開発作業に掛かる日数

③ 目標作業期間は，今回の開発作業を開始する日から，この対象サブシステムの

目標作業終了日までの日数

(3) 対象サブシステム情報 job には，次の①～③の順で，データが格納されている。

① 目標作業期間の昇順

② 目標作業期間が同じならば，開発作業日数の昇順

③ 目標作業期間も開発作業日数も同じならば，開発コードの昇順

(4) (5)の手順で，開発作業順序を求めると，開発作業順序 job_sch の要素 job_sch[i] には，i ＋ 1 番目（i ＝ 0，1，…，num_s － 1）に開発作業を実施する対象サブシステムの情報が格納された job[j] の添字の値 j（j ＝ 0，1，…，num_s － 1）が入る。

(5) 開発作業順序を求める手順は，次のとおりである。

① 開発作業順序 job_sch を，対象サブシステム情報 job の並び順に初期設定し，開発作業順序の添字 i に初期値 0 を設定する。

② i ＜ num_s － 1 である間，③～⑥の処理を行う。

③ i ＋ 1 番目に開発作業を実施する対象サブシステム job[job_sch[i]] と i ＋ 2 番目に開発作業を実施する対象サブシステム job[job_sch[i ＋ 1]] の開発作業を，順序どおりに実施した場合の二つの対象サブシステムの遅延日数の合計 wt_a と，順序を入れ替えて実施した場合の二つの対象サブシステムの遅延日数の合計 wt_b を求める。

④ 開発作業の実施順序を入れ替えることによって遅延日数の合計が減る場合（wt_a ＞ wt_b）には，開発作業の実施順序を入れ替える。

⑤ 開発作業の実施順序を入れ替えた場合，i が 0 でなければ，i を 1 減らす。

⑥ 入れ替えなかった場合，i を 1 増やす。

(6) i 番目の対象サブシステムの開発作業が終わるまでの開発作業日数の合計を ft としたとき，i ＋ 1 番目の対象サブシステムの開発作業が終わったときの遅延日数の合計 wt は，次式で求められる。

$$wt = \begin{cases} ft + job\_term - target\_term & (ft + job\_term > target\_term) \\ 0 & (ft + job\_term \leqq target\_term) \end{cases}$$

ここで，job_term と target_term は，i ＋ 1 番目に開発作業を実施する対象サブシステムの開発作業日数と目標作業期間である。

〔プログラム1〕

```
typedef struct {
    char pr_code[9];    /* 開発コード */
    int  job_term;      /* 開発作業日数 */
    int  target_term;   /* 目標作業期間 */
} JOB;

void job_scheduling(int, JOB[], int[]);

void job_scheduling(int num_s, JOB job[num_s], int job_sch[num_s]) {
    int ft, ft_a, ft_b, wt_a, wt_b, job_no, i, j;

    for (i = 0; i < num_s; i++) {
        job_sch[i] = i;
    }

    ft = 0;
    i  = 0;
    while (i < num_s - 1) {
        ft_a = ft;
        ft_b = ft;
        wt_a = 0;
        wt_b = 0;
        for (j = 0; j < 2; j++) { /* 遅延日数の合計 wt_a, wt_b を求める */
            ft_a += job[job_sch[i + j]].job_term;
            if (ft_a > job[job_sch[i + j]].target_term) {
                wt_a += ft_a - job[job_sch[i + j]].target_term;
            }
            ft_b += job[job_sch[ a ]].job_term;
            if (ft_b > job[job_sch[ a ]].target_term) {
                wt_b += ft_b - job[job_sch[ a ]].target_term;
            }
        }

        if (wt_a > wt_b) {
            job_no         = job_sch[i];
            job_sch[i]     = job_sch[ b ];
            job_sch[ b ] = job_no;
            if (i > 0) {
                ft -= job[job_sch[ c ]].job_term;
            }
```

```
        } else {
            ft [ d ] job[job_sch[i++]].job_term;
        }
    }
}
```

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に関する解答群

ア　i + j　　イ　i + j - 1　　ウ　i - j　　エ　i - j + 1

bに関する解答群

ア　i + 1　　イ　i - 1　　ウ　job_no

エ　job_no + 1　　オ　job_no - 1

cに関する解答群

ア　i + 1　　イ　i - 1　　ウ　i++

エ　i--　　オ　++i　　カ　--i

dに関する解答群

ア　=　　イ　+=　　ウ　-=

設問2　関数 job_scheduling を実行したときに得られる対象サブシステムの開発作業順序などを出力する関数 print_schedule 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。

(1) 関数 print_schedule の仕様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。

機能： 対象サブシステムの開発作業順序などを出力する。

引数： num_s 対象サブシステム数

job 対象サブシステム情報（JOB 型の配列）

job_sch 開発作業順序（int 型の配列）

(2) 図 1 に示す五つの対象サブシステムの対象サブシステム情報に対して関数 job_scheduling を実行した。このときに得られた開発作業順序を，図 2 に示す。

	pr_code	job_term	target_term
job[0]	"APL12339"	25	27
job[1]	"GGL08001"	27	29
job[2]	"MSO16101"	21	30
job[3]	"CAN03022"	28	77
job[4]	"ORA14031"	12	93

図 1 対象サブシステム情報

job_sch[0]	0
job_sch[1]	2
job_sch[2]	1
job_sch[3]	4
job_sch[4]	3

図 2 開発作業順序

(3) 続けて，関数 print_schedule を実行した。その出力結果の 3 行目を図 3 に示す。

3	GGL08001	e	f

注記 枠線は，実際には出力されない。

図 3 関数 print_schedule の出力結果の 3 行目

〔プログラム2〕

```
#include <stdio.h>

typedef struct {
    char pr_code[9];    /* 開発コード */
    int  job_term;      /* 開発作業日数 */
    int  target_term;   /* 目標作業期間 */
} JOB;

void print_schedule(int, JOB[], int[]);

void print_schedule(int num_s, JOB job[num_s], int job_sch[num_s]) {
    int ft, wt, wt_sum, i;

    ft      = 0;
    wt_sum  = 0;
    for (i = 0; i < num_s; i++) {
        ft += job[job_sch[i]].job_term;
        if (ft > job[job_sch[i]].target_term) {
            wt = ft - job[job_sch[i]].target_term;
            wt_sum += wt;
        } else {
            wt = 0;
        }
        printf("%3d  %10s  %10d  %10d\n",
               i + 1, job[job_sch[i]].pr_code, wt, wt_sum);
    }
}
```

e, fに関する解答群

ア 0	イ 27	ウ 44	エ 46
オ 57	カ 60	キ 73	ク 86

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〕

Z 社では，毎年実施している従業員の健康診断を今年も実施した。健康診断の結果は，過去 10 回分が健康診断マスタファイルに記録されている。このプログラムは，今回の健康診断の結果が記録された結果ファイルを読み込み，健康診断マスタファイルに反映する。

(1)　健康診断マスタファイルは，図1に示すレコード様式の順ファイルである。

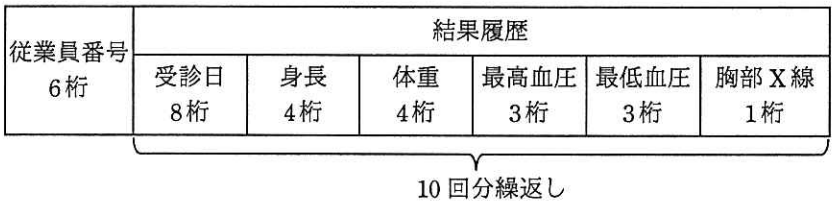


図1　健康診断マスタファイルのレコード様式

①　全ての従業員に対するレコードが，従業員番号の昇順に格納される。

②　結果履歴には，当該従業員の過去 10 回分の健康診断の結果が，受診日の降順に格納される。受診回数が 10 回に満たない場合，残りの結果履歴には全てゼロが設定される。

③　受診日には，年，月，日が，それぞれ4桁，2桁，2桁の西暦で格納される。

④　身長(cm)，体重(kg)，最高血圧(mmHg)，最低血圧(mmHg)には，それぞれの数値が格納される。身長と体重は小数点以下 1 桁までが格納される。胸部 X 線には，所見がない場合は0が，所見がある場合は1が格納される。

(2)　結果ファイルは，図2に示すレコード様式の順ファイルである。

様式1

受診日 8桁	未使用域 13桁

様式2

従業員番号 6桁	身長 4桁	体重 4桁	最高血圧 3桁	最低血圧 3桁	胸部X線 1桁

図2　結果ファイルのレコード様式

① 結果ファイルの先頭レコードとして，様式1のレコードが1件だけ格納される。受診日には，年，月，日が，それぞれ4桁，2桁，2桁の西暦で格納される。受診日は，健康診断マスタファイルに記録されているどの受診日よりも新しい。

② 2件目以降のレコードとして，受診した従業員の様式2のレコードが順不同で格納される。受診していない従業員に対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  SD SORT-FILE.
 4  01 SORT-REC.
 5     02 SORT-NO          PIC 9(6).
 6     02 SORT-ELM         PIC X(15).
 7  FD MST-FILE.
 8  01 MST-REC.
 9     02 MST-NO           PIC 9(6).
10     02 MST-HISTORY      OCCURS 10.
11        03 MST-DATE          PIC 9(8).
12        03 MST-ELM.
13           04 MST-HEIGHT      PIC 9(3)V9(1).
14           04 MST-WEIGHT      PIC 9(3)V9(1).
15           04 MST-HIGHPRES    PIC 9(3).
16           04 MST-LOWPRES     PIC 9(3).
17           04 MST-XRAY        PIC 9(1).
18 FD RSLT-FILE.
19 01 FIRST-REC.
20    02 RSLT-DATE         PIC 9(8).
21    02                   PIC X(13).
22 01 RSLT-REC         PIC X(21).
```

```
23  WORKING-STORAGE SECTION.
24  77 CNT          PIC 9(2).
25  77 NEW-DATE     PIC 9(8).
26  77 RSLT-FLAG    PIC X(1) VALUE SPACE.
27     88 RSLT-EOF     VALUE "E".
28  77 SORT-FLAG    PIC X(1) VALUE SPACE.
29     88 SORT-EOF     VALUE "E".
30  PROCEDURE DIVISION.
31  MAIN-PROC.
32        OPEN I-O MST-FILE INPUT RSLT-FILE.
33        READ RSLT-FILE.
34        [          a          ].
35        SORT SORT-FILE ASCENDING KEY SORT-NO
36           INPUT  PROCEDURE IS REL-PROC
37           OUTPUT PROCEDURE IS RET-PROC.
38        CLOSE MST-FILE RSLT-FILE.
39        STOP RUN.
40  REL-PROC.
41        PERFORM UNTIL RSLT-EOF
42           READ RSLT-FILE AT END     SET RSLT-EOF TO TRUE
43                          NOT AT END [          b          ]
44           END-READ
45        END-PERFORM.
46  RET-PROC.
47        PERFORM UNTIL SORT-EOF
48           RETURN SORT-FILE AT END     SET SORT-EOF TO TRUE
49                            NOT AT END PERFORM MATCHING-PROC
50           END-RETURN
51        END-PERFORM.
52  MATCHING-PROC.
53        READ MST-FILE.
54        PERFORM TEST BEFORE UNTIL [          c          ]
55           READ MST-FILE END-READ
56        END-PERFORM.
57        PERFORM TEST BEFORE VARYING CNT [          d          ] UNTIL CNT = 0
58           MOVE MST-HISTORY(CNT) TO MST-HISTORY(CNT + 1)
59        END-PERFORM.
60        MOVE NEW-DATE TO MST-DATE(1).
61        MOVE SORT-ELM TO MST-ELM(1).
62        REWRITE MST-REC.
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a，bに関する解答群

ア　MOVE RSLT-DATE TO NEW-DATE

イ　MOVE ZERO TO CNT

ウ　MOVE ZERO TO NEW-DATE

エ　PERFORM RET-PROC

オ　RELEASE SORT-REC

カ　RELEASE SORT-REC FROM RSLT-REC

cに関する解答群

ア　MST-NO = SORT-NO　　イ　MST-NO < SORT-NO

ウ　MST-NO > SORT-NO　　エ　MST-NO NOT = SORT-NO

dに関する解答群

ア　FROM 0 BY 1　　イ　FROM 9 BY -1

ウ　FROM 10 BY -1　　エ　FROM CNT BY 1

設問2　プログラムを変更して，健康診断の結果を健康診断マスタファイルに反映するときに，全ての従業員の確認結果を図 3 に示すように表示する。表 1 中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
920186: NO-DATA
920189:
920221: WEIGHT BLOOD-PRESSURE XRAY
930015: WEIGHT
930016:
940047: NO-DATA
  :
```

図3　確認結果の表示例

〔確認結果の表示についての説明〕

次に該当する従業員は，従業員番号に続けて該当する文字列を表示する。

なお，複数該当する場合は，その全てを表示する。

① 今回受診していない場合は NO-DATA

② 今回と前回の体重を比較して，4kg 以上の増減がある場合は WEIGHT

③ 今回の最高血圧が 140 mmHg 以上（高血圧症状），又は 90 mmHg 以下（低血圧症状）である場合は BLOOD-PRESSURE

④ 今回の胸部 X 線の診断で所見がある場合は XRAY

表1 プログラムの変更内容

処置	変更内容
行番号 29 と 30 の間に追加	77 MST-FLAG PIC X(1) VALUE SPACE. 88 MST-EOF VALUE "E". 01 DISP-WK PIC X(30).
[e] に追加	PERFORM UNTIL MST-EOF READ MST-FILE AT END SET MST-EOF TO TRUE NOT AT END DISPLAY MST-NO ": NO-DATA" END-READ END-PERFORM.
行番号 54 と 55 の間に追加	DISPLAY MST-NO ": NO-DATA"
行番号 61 と 62 の間に追加	[f]. MOVE SPACE TO DISP-WK. IF MST-DATE(2) NOT = ZERO AND ([g]) THEN MOVE " WEIGHT" TO DISP-WK(CNT:) ADD 7 TO CNT END-IF. IF MST-HIGHPRES(1) >= 140 OR MST-HIGHPRES(1) <= 90 THEN MOVE " BLOOD-PRESSURE" TO DISP-WK(CNT:) ADD 15 TO CNT END-IF. IF MST-XRAY(1) = 1 THEN MOVE " XRAY" TO DISP-WK(CNT:) END-IF. DISPLAY MST-NO ":" DISP-WK.

eに関する解答群

ア　行番号34と35の間　　イ　行番号45と46の間

ウ　行番号51と52の間　　エ　行番号52と53の間

fに関する解答群

ア　MOVE 1 TO CNT

イ　MOVE 10 TO CNT

ウ　MOVE MST-DATE(1) TO NEW-DATE

エ　MOVE ZERO TO NEW-DATE

gに関する解答群

ア
```
MST-WEIGHT(1) >= MST-WEIGHT(2) + 4 OR
MST-WEIGHT(1) <= MST-WEIGHT(2) - 4
```

イ
```
MST-WEIGHT(1) >= MST-WEIGHT(CNT) + 4 OR
MST-WEIGHT(1) <= MST-WEIGHT(CNT) - 4
```

ウ
```
MST-WEIGHT(10) >= MST-WEIGHT(9) + 4 OR
MST-WEIGHT(10) <= MST-WEIGHT(9) - 4
```

エ
```
MST-WEIGHT(10) >= MST-WEIGHT(CNT) + 4 OR
MST-WEIGHT(10) <= MST-WEIGHT(CNT) - 4
```

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

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

〔プログラムの説明〕

整数値の加減乗除の演算をする電卓のプログラムである。この電卓は，数字キー，加減乗除の各演算キー，イコールキー及びクリアキーをもつ。プログラムは，キーが押されたとき，それぞれのキーに対応する処理を実行する。数値などの表示は，System.out.println を呼び出して行う。

(1) インタフェース Key は，電卓のキーが押されたときの処理を実行するメソッドを定義する。

メソッド operateOn は，引数で与えられたクラス java.util.Stack のインスタンス（以下，スタックという）に対して，キーに対応する処理を実行する。

(2) 列挙 DigitKey は，数字キーを表す定数 DIGIT0 ～ DIGIT9 を定義する。

メソッド operateOn は，キーを 10 進数の入力として処理する。引数で与えられたスタックの先頭に格納されている値は 0（初期値）又は入力中の数値であり，その値を更新する。

(3) 列挙 OperationKey は，加減乗除の各演算キー，イコールキー及びクリアキーを表す定数 ADD，SUBTRACT，MULTIPLY，DIVIDE，EQUAL 及び CLEAR を定義する。

メソッド operateOn は，加減乗除の各演算キーに対応する演算を，スタックの内容に対して実行する。

(4) クラス Calculator は，電卓本体を表す。フィールド stack は，電卓内部の数値の状態を表すスタックを保持する。フィールド pendingKey は，演算に必要な数値の入力が終わるまで演算キーを保持する。また，イコールキーが押されたときは，イコールキーを保持する。例えば，キーの定数 DIGIT2，ADD，DIGIT4 が順に処理されたとき，スタックに格納されている値は先頭から 4，2 であり，pendingKey の値は ADD である。次にキーの定数 EQUAL が処理されたとき，演算キーADD の加算処理が実行され，スタックに格納されている値は 6 となり，pendingKey の値は EQUAL となる。ここで，二つの数値に対する加減乗除の演算結果は，Java の int 型の演算結果に一致するものとする。

メソッド onKeyPressed は，電卓のキーが押されたときに呼び出される。押されたキーは，引数で与えられる。押されたキー及び電卓の内部状態に基づいて，処理を実行する。

(5) クラス CalculatorTest は，クラス Calculator をテストするプログラムである。

メソッド main は，まず，文字と電卓の各キーとの対応を作成し，クラス Calculator のインスタンスを生成する。次に，引数で与えられた文字列の各文字をキーの定数に変換し，そのキーの定数を引数としてクラス Calculator のインスタンスのメソッド onKeyPressed を呼び出す。例えば，メソッド main の引数として文字列“2*3=”が与えられたとき，それぞれの文字を，キーの定数 DIGIT2, MULTIPLY, DIGIT3, EQUAL に変換し，逐次それぞれのキーの定数を引数としてメソッド onKeyPressed を呼び出す。メソッド main を実行したときの出力を図1に示す。

```
DIGIT2
2
MULTIPLY
2
DIGIT3
3
EQUAL
6
```

図1　メソッド main を実行したときの出力

〔プログラム1〕

```
import java.util.Stack;

public interface Key {
   public void operateOn(Stack<Integer> stack);
}
```

〔プログラム2〕

```
import java.util.Stack;

enum DigitKey [ a ] Key {
   DIGIT0, DIGIT1, DIGIT2, DIGIT3, DIGIT4,
   DIGIT5, DIGIT6, DIGIT7, DIGIT8, DIGIT9;
```

```
    public void operateOn(Stack<Integer> stack) {
        stack.push( [ b ] * 10 + [ c ] );
    }
}
```

〔プログラム3〕

```
import java.util.Stack;

enum OperationKey [ a ] Key {
    ADD, SUBTRACT, MULTIPLY, DIVIDE, EQUAL, CLEAR;

    public void operateOn(Stack<Integer> stack) {
        if (this == EQUAL || this == CLEAR) {
            return;
        }
        int val2 = stack.pop();
        int val1 = stack.pop();
        stack.push(calculate(val1, val2));
    }

    private int calculate(int val1, int val2) {
        switch ( [ d ] ) {
        case ADD:
            return val1 + val2;
        case SUBTRACT:
            return val1 - val2;
        case MULTIPLY:
            return val1 * val2;
        case DIVIDE:
            return val1 / val2;
        default:
            throw new AssertionError(toString());
        }
    }
}
```

〔プログラム4〕

```
import java.util.Stack;

public class Calculator {
    private final Stack<Integer> stack = new Stack<Integer>();
```

```
    private Key pendingKey;

    public Calculator() {
        stack.push(0);
    }

    public void onKeyPressed(Key key) {
        System.out.println(key);
        if (key instanceof DigitKey) {
            if (pendingKey == OperationKey.EQUAL) {
                reset();
            }
            key.operateOn(stack);
            System.out.println(stack.peek());
        } else if (key == OperationKey.CLEAR) {
            reset();
            System.out.println(stack.peek());
        } else {
            try {
                if (pendingKey != null) {
                    pendingKey.operateOn(stack);
                }
                System.out.println(stack.peek());
                pendingKey = key;
                if (key != OperationKey.EQUAL) {
                    stack.push(0);
                }
            } catch (ArithmeticException e) {
                System.out.println("Error");
                reset();
            }
        }
    }

    private void reset() {
        stack.clear();
        stack.push(0);
        pendingKey = null;
    }
}
```

〔プログラム5〕

```
import java.util.HashMap;
import java.util.Map;

public class CalculatorTest {
   public static void main(String[] args) {
      Map<Character, [ e ]> map = new HashMap<Character, [ e ]>();
      // 文字と列挙OperationKeyの定数の対応をmapに格納する。
      for (OperationKey key : OperationKey.values())
         map.put("+-*/=C".charAt(key.ordinal()), key);
      // 数字と列挙DigitKeyの定数の対応をmapに格納する。
      for (DigitKey key : DigitKey.values())
         map.put("0123456789".charAt(key.ordinal()), key);

      Calculator calc = new Calculator();
      String chars = args[0];
      // charsの各文字をキーの定数に変換し，メソッドonKeyPressedを呼び出す。
      for (int i = 0; i < chars.length(); i++) {
         calc.onKeyPressed(map.get(chars.charAt(i)));
      }
   }
}
```

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　extends	イ　implements	ウ　imports
エ　inherits	オ　requires	カ　throws

b，c に関する解答群

ア　ordinal()	イ　stack.peek()	ウ　stack.pop()
エ　stack.push(0)	オ　stack.push(ordinal())	カ　values()

d に関する解答群

ア　DigitKey	イ　Key	ウ　stack.pop()
エ　this	オ　val1	カ　val2

e に関する解答群

ア Calculator	イ Character	ウ DigitKey
エ Integer	オ Key	カ OperationKey

設問2　表1は，文字列を引数としてメソッド main を実行したときの出力の最後の行（図1の場合は6）を表している。表中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，プログラム中の［　　］には，全て正しい答えが入っているものとする。

表1　文字列（引数）と出力（最後の行）

文字列（引数）	出力（最後の行）
2*6/3=	4
-2=	-2
2*4==	8
2*4C2=	f
8/2/=	g

f，g に関する解答群

ア 0	イ 2	ウ 4
エ 8	オ 16	カ 32
キ 64	ク ArithmeticException	ケ Error

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〕

リストに対して，要素を挿入又は削除する副プログラム LPROC である。リストの構造を図1に示す。

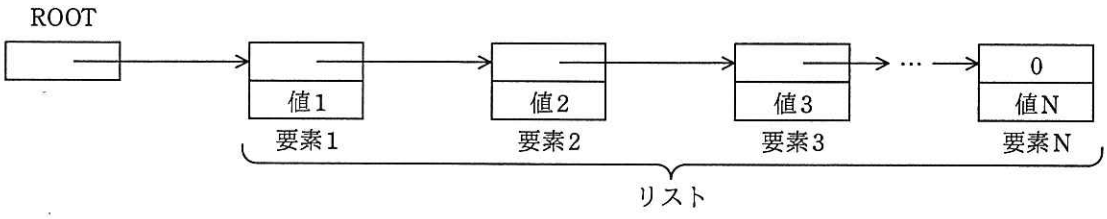


図1　リストの構造

(1)　リストは一つ以上の要素をもち，一つの要素は連続する2語から成る。1語目には次の要素の先頭アドレス（図中では矢印で表す）が，2語目にはその要素の保持する値が格納される。要素は主記憶の0番地以外に配置され，最後の要素の1語目には，次の要素がないことを表すために，0が設定される。

(2)　ROOT は1語から成り，最初の要素（要素1）の先頭アドレスが格納される。

(3)　主プログラムは，表1に示す値を各レジスタに設定して LPROC を呼ぶ。

①　n は操作対象とする要素が，リストの先頭から何番目かを指定する正の整数である。

②　要素の挿入の場合は，その要素が先頭から n 番目となる位置（要素 n の直前）に挿入される。削除の場合は，先頭から n 番目の要素（要素 n）が削除される。

③　nとして，リストの現在の要素の個数 N よりも大きい値が与えられたとき，LPROC は何もせずに呼出し元に戻る。

表1　LPROC 呼出し時のレジスタ設定

操作	GR0	GR1	GR2	GR3
挿入	挿入する値	ROOT のアドレス	n	0
削除	－	ROOT のアドレス	n	1

(4) LPROC は，要素を挿入又は削除するとき，それぞれ別に用意された副プログラム EGET 又は EFREE を呼ぶ。

(5) EGET は，挿入される要素に必要な領域を記憶管理領域から割り当てるために呼び出され，新たに割り当てられた，連続する 2 語から成る領域の先頭アドレスを GR2 に設定して呼出し元に返す。

(6) EFREE は，削除された要素の先頭アドレスを GR2 に設定して呼び出され，当該要素が使用していた領域を記憶管理領域に戻す。

(7) 副プログラム EGET から戻るとき，GR2 以外の汎用レジスタの内容は元に戻す。

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

(9) 副プログラム LPROC 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  LPROC  START
 2         RPUSH
 3         LD     GR4,GR2          ; GR4 ← n
 4  LP     LD     GR2,0,GR1        ; 次の要素をたどる
 5         JZE    FIN              ; n ＞ N か？
 6         SUBA   GR4,=1
 7         JZE    NEXT             ; 要素 n に達したか？
 8         |        a              |
 9         JUMP   LP
10  NEXT   LD     GR3,LTBL,GR3
11         JUMP   0,GR3            ; 指定された処理にジャンプ
12  LINS   LD     GR4,GR2
13         CALL   EGET             ; GR2 ← 挿入される要素の先頭アドレス
14         |        b              |
15         ST     GR4,0,GR2        ; 挿入される要素の 1 語目を設定
16         ST     GR0,1,GR2
17         JUMP   FIN
18  LDEL   |        c              |  ; GR4 ← 要素 n ＋ 1 の先頭アドレス
19         ST     GR4,0,GR1        ; 要素 n － 1 の 1 語目を再設定
20         CALL   EFREE            ; 削除された要素 n の領域を解放
21  FIN    RPOP
22         RET
23  LTBL   DC     LINS             ; 処理の分岐先アドレステーブル
24         DC     LDEL
25         END
```

アセンブラ

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に関する解答群

ア	JMI	FIN	イ	LAD	GR1,1,GR1	ウ	LAD	GR4,1,GR4
エ	LD	GR1,GR2	オ	LD	GR1,0,GR2			

bに関する解答群

ア	LD	GR4,0,GR1	イ	LD	GR4,0,GR2	ウ	ST	GR2,0,GR1
エ	ST	GR2,0,GR2	オ	ST	GR4,0,GR1	カ	ST	GR4,0,GR2

cに関する解答群

ア	LD	GR4,0,GR1	イ	LD	GR4,0,GR2	ウ	LD	GR4,2,GR1
エ	LD	GR4,2,GR2	オ	LD	GR4,GR1	カ	LD	GR4,GR2

設問2　図2に示す要素数が3個のリストについて，n＝3でLPROCを実行し，要素を挿入する。行番号12のラベルLINSの命令を実行するときGR2に格納されている値として，正しい答えを，解答群の中から選べ。

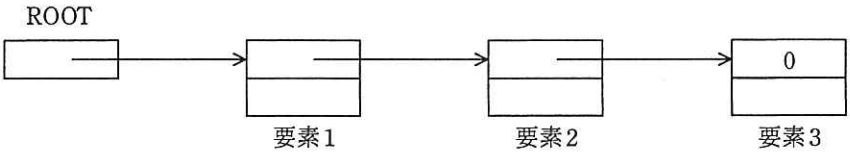


図2　要素数が3個のリスト

解答群

ア　挿入する要素の先頭アドレス	イ　要素2の先頭アドレス
ウ　要素3の先頭アドレス	エ　0

設問 3　ラベル LIST 以降に格納されている要素数が 4 個のリストについて，プログラム 2 の MAIN を実行した。実行後のラベル E3 で示される番地の内容として，正しい答えを，解答群の中から選べ。ここで，EGET で割り当てられた要素の先頭アドレスは，割り当てられた順に，α_1，α_2とする。

〔プログラム 2〕

```
MAIN    START
        LD    GR0,=1
        LD    GR2,=3        ; n = 3
        LD    GR3,=0
        LAD   GR1,ROOT
        CALL  LPROC
        LD    GR0,=2
        CALL  LPROC
        LD    GR3,=1
        CALL  LPROC
        RET
ROOT    DC    LIST
LIST    DC    E3,#000A      ; リストの最初の要素（要素 1）
E1      DC    E2,#000B
E2      DC    #0000,#000D   ; リストの最後の要素
E3      DC    E1,#000C
        END
```

解答群

ア　α_1　　イ　α_2　　ウ　E1　　エ　E2　　オ　0

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〕

A社では，建物の改築作業を効率的に遂行するスケジュールを決定するために表計算ソフトを用いている。

〔ワークシート：状態遷移〕

作業とは，建物の改築状態をある状態から別の状態にするための行為とする。20種類の状態があり，各状態には，1～20の番号（以下，状態IDという）が振られている。

改築作業の作業工程は，一つ以上の作業を順に施すことをいう。一連の作業による改築状態の遷移を，状態IDの列で表現する。これを，状態遷移列と呼ぶ。

はじめに，作業による改築状態の遷移の関係，及び各作業の作業日数を入力したワークシート“状態遷移”を作成した。ワークシート“状態遷移”を，図1に示す。

	A	B	C	D	E	…	U	V
1			作業開始時の状態ID					
2			1	2	3	…	19	20
3	作業終了時の状態ID	1	0	0	0	…	0	0
4		2	5	0	0	…	0	0
5		3	8	4	0	…	0	0
6		4	0	10	8	…	0	0
⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮
21		19	0	0	0	…	0	0
22		20	0	0	0	…	4	0

図1　ワークシート“状態遷移”

(1)　セルC2～V2には，作業開始時の状態IDを1から順に入力する。同様にセルB3～B22には，作業終了時の状態IDを1から順に入力する。

(2)　セルC3～V22には，対応する状態間を直接遷移する作業が存在すれば作業に掛かる日数（以下，作業日数という）を，存在しなければ0を入力する。作業日数は，100日以下の整数値である。例えば，図1のセルC4の値は，作業開始時の状態IDが1であって，作業終了時の状態IDが2となる作業が存在し，その作業は5日掛

かることを表している。また，セル C6 の値は，作業開始時の状態 ID が 1 であって，作業終了時の状態 ID が 4 となる作業が存在しないことを表している。

(3) 全ての作業について，作業開始時の状態 ID よりも作業終了時の状態 ID の方が大きい。

〔ワークシート：作業工程〕

状態遷移列を入力すると，その状態遷移列が存在するかどうかを判定し，存在する場合は作業工程の総日数を算出するワークシート“作業工程”を作成した。ワークシート“作業工程”において，状態遷移列が存在する場合の例と状態遷移列が存在しない場合の例を，図 2 に示す。

	A	B	C
1	状態遷移列		
2	の状態 ID	日数	総日数
3	3	0	0
4	4	8	8
5	6	8	16
⋮	⋮	⋮	⋮
8	11	5	35
9		0	35
⋮	⋮	⋮	⋮
22		0	35
23		計	35

状態遷移列が存在する場合の例

	A	B	C
1	状態遷移列		
2	の状態 ID	日数	総日数
3	3	0	0
4	4	8	8
5	7	×	×
⋮	⋮	⋮	⋮
8	15	×	×
9	17	×	×
⋮	⋮	⋮	⋮
22		0	×
23		計	×

状態遷移列が存在しない場合の例

図 2　ワークシート“作業工程”の例

(1) セル A3 ～ A22 には，状態遷移列を構成する状態 ID をセル A3 から順に入力する。最後の状態 ID が入力されているセルよりも下のセルには空値が入力されている。同じ状態 ID が 2 回以上入力されることはない。

(2) セル B3 には，0 を入力する。セル B4 ～ B22 には，同じ行の列 A の状態 ID へ 1 行上の列 A の状態 ID から直接遷移するのに必要な作業日数が表示される。直接遷移する作業が存在しない場合は，“×”が表示される。同じ行の列 A のセルの値が空値の場合は 0 が表示される。

(3) セル C3 には，0 を入力する。セル C4 ～ C22 には，セル A4 ～ A22 の各状態 ID へ，セル A3 に入力されている状態 ID から遷移するのに必要な作業の総日数が表示される。同じ行の列 A の状態 ID へ 1 行上の列 A の状態 ID から直接遷移する作業が存在しない場合は，“×”が表示される。一つ上のセルの値が“×”の場合は，“×”が表示される。同じ行の列 A のセルの値が空値の場合は，一つ上のセルと同じ値が表示される。

(4) セル C23 には，状態遷移列が存在する場合は，作業工程の総日数が表示され，存在しない場合は“×”が表示される。

設問 1　ワークシート“作業工程”の作成手順 (1)～(4) 中の［　　］に入れる正しい答えを，解答群の中から選べ。

(1) セル B3 及びセル C3 に 0 を入力する。

(2) 次の式をセル B4 に入力し，セル B5 ～ B22 に複写する。

IF(A4 ＝ null, 0, IF(［ a ］＞0,［ a ］, '×'))

(3) 次の式をセル C4 に入力し，セル C5 ～ C22 に複写する。

IF(［ b ］, '×',［ c ］)

(4) 次の式をセル C23 に入力する。

C22

a に関する解答群

ア　表引き(状態遷移!C$3:V$22, A$4, A3)

イ　表引き(状態遷移!C$3:V$22, A4, A$3)

ウ　表引き(状態遷移!C$3:V$22, A4, A3)

エ　表引き(状態遷移!C3:V$22, A$4, A3)

オ　表引き(状態遷移!C3:V$22, A4, A$3)

カ　表引き(状態遷移!C3:V$22, A4, A3)

b に関する解答群

ア　B4 ＝ '×'

イ　C3 ＝ '×'

ウ　論理積(B4 ＝ '×', C3 ＝ '×')

エ　論理積(B4 ≠ '×', C3 ≠ '×')

オ　論理和(B4 ＝ '×', C3 ＝ '×')

カ　論理和(B4 ≠ '×', C3 ≠ '×')

cに関する解答群

ア　B3＋B4　　イ　C3＋B4

ウ　C5－B4　　エ　C5－B5

〔ワークシート：作業工程（拡張）〕

状態遷移列の先頭になる状態 ID（以下，開始状態 ID という）と状態遷移列の最後になる状態 ID（以下，終了状態 ID という）を入力すると，作業工程の総日数が最小になる状態遷移列を求める機能を追加したワークシート“作業工程（拡張）”を作成し，マクロ CalculateMinimum とマクロ DisplayMinimumPath を格納した。作成したワークシート“作業工程（拡張）”の例を，図3に示す。

なお，ワークシート“作業工程”を拡張することによって，関数“表引き”の引数で与えられた行又は列の位置の値が空値になることがある。その場合は，関数“表引き”は空値を返す。

	A	B	C	D	E	F	G	H	I	J
1	状態遷移列									
2	の状態 ID	日数	総日数		開始状態 ID	3		状態 ID	最小日数	前状態 ID
3	3	0	0		終了状態 ID	13		1	9999	
4	4	8	8					2	9999	
5	6	8	16					3	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9	11	5	35					7	19	6
10	13	12	47					8	21	6
11		0	47					9	28	8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15		0	47					13	47	11
16		0	47					14	51	13
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
22		0	47					20	80	19
23		計	47							

図3　ワークシート“作業工程（拡張）”の例

(1)　セル F2 に開始状態 ID を，セル F3 に終了状態 ID を入力し，マクロ CalculateMinimum を実行した後，マクロ DisplayMinimumPath を実行する。

(2) マクロ CalculateMinimum の仕様は，次のとおりである。

① セル H3～H22 には，1から順に状態 ID を格納する。

② セル I3～I22 には，開始状態 ID から各行の列 H で示す状態 ID まで遷移するのに必要な作業の総日数の最小値を格納する。状態遷移列が存在しない場合は 9999 を格納する。

③ セル J3～J22 には，開始状態 ID から各行の列 H で示す状態 ID まで遷移するのに必要な作業の総日数が最小となる状態遷移列における，その状態 ID の直前の状態 ID を格納する。開始状態 ID に対応するセルの場合は 0 を格納する。また，状態遷移列が存在しない場合は空値を格納する。

(3) マクロ DisplayMinimumPath は，セル A3～A22 に作業工程の総日数が最小となる状態遷移列を格納する。終了状態 ID を格納したセルよりも下のセルには空値を格納する。開始状態 ID から終了状態 ID までの状態遷移列が存在しない場合は，セル A3～A22 には全て空値を格納する。

設問2　マクロ DisplayMinimumPath 中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
〔マクロ：DisplayMinimumPath〕
    ○マクロ: DisplayMinimumPath
    ○数値型: I, NumWork, Current
    ■I: 1, I ≦ 20, 1
    |・相対(A2, I, 0) ← null
    ■
    ▲相対(I2, F3, 0) ＜ 9999    /* 状態遷移列が存在する */
    |・NumWork ← 0
    |・Current ← F3
    |/* 作業工程の作業数 NumWork を算出する */
    |■［  d  ］
    ||・Current ← ［  e  ］
    ||・NumWork ← NumWork ＋ 1
    |■
    |/* 状態遷移列を格納する */
    |・Current ← F3
    |■［  f  ］
    ||・相対(A3, I, 0)  ← Current
    ||・Current ← ［  e  ］
    |■
    ▼
```

表計算

dに関する解答群

ア　Current ＜ F2　　イ　Current ＜ F3　　ウ　Current ＜ F3 － F2

エ　Current ＝ 0　　オ　Current ＝ F2　　カ　Current ≠ F2

eに関する解答群

ア　照合検索(Current, H3:H22, J3:J22)

イ　照合検索(Current, J3:J22, H3:H22)

ウ　照合検索(NumWork, H3:H22, J3:J22)

エ　照合検索(NumWork, J3:J22, H3:H22)

fに関する解答群

ア　I: 0, I ＜ NumWork, 1　　イ　I: 0, I ≦ NumWork, 1

ウ　I: 1, I ≦ NumWork, 1　　エ　I: NumWork － 1, I ≧ 0, －1

オ　I: NumWork , I ＞ 0, －1　　カ　I: NumWork, I ≧ 0, －1

■ Java プログラムで使用する API の説明

java.util

public interface Map<K, V>

型 K のキーに型 V の値を対応付けて保持するインタフェースを提供する。各キーは，一つの値としか対応付けられない。

メソッド

public V get(Object key)

指定されたキーに対応付けられた値を得る。

引数： key — キー

戻り値：指定されたキーに対応付けられた型 V の値
このキーと値の対応付けがなければ null

public V put(K key, V value)

指定されたキーに指定された値を対応付けて登録する。このキーが既にほかの値と対応付けられていれば，その値は指定された値に置き換えられる。

引数： key — キー
value — 値

戻り値：指定されたキーに対応付けられていた型 V の古い値
このキーと値の対応付けがなければ null

java.util

public class HashMap<K, V>

インタフェース Map のハッシュを用いた実装である。

コンストラクタ

public HashMap()

空の HashMap を作る。

java.util

public class Stack<E>

クラス Stack は，オブジェクトの後入れ先出し（LIFO）スタックを表す。

コンストラクタ

public Stack()

空のスタックを生成する。

メソッド

public E push(E item)

引数で指定されたオブジェクトをスタックの先頭にプッシュする。

引数： item — スタックにプッシュするオブジェクト

戻り値：引数で指定されたオブジェクト

public E pop()

スタックの先頭のオブジェクトをスタックから削除し，そのオブジェクトを返す。

戻り値：スタックの先頭から取り出したオブジェクト

public E peek()

スタックの先頭のオブジェクトをスタックから削除せずに返す。

戻り値：スタックの先頭のオブジェクト

public void clear()

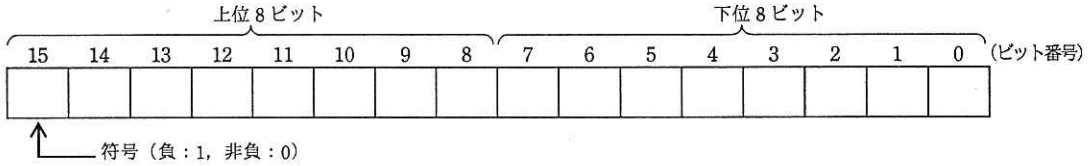
スタックの全オブジェクトをスタックから削除し，スタックを空にする。

■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。	

注記

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]

注記 [ ] [ ] 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。CASLⅡ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 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 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 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(−314.059,2) は，−314.06 を返す。 ［例2］切上げ(314.059,−2) は，400 を返す。 ［例3］切上げ(314.059,0) 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 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 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 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セル 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セル 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セル 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セル 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”の〔宣言，注釈及び処理〕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(E1, row, 1) ← 垂直照合(相対(E1, row, 0), A1:B10, 2, 0) * 10
```

例は，セル E1，E2，…，E5 の各値に対して，セル A1 ～ A10 の中で同じ値をもつセルが現れる最初の行を探し，見つけた行の列 B のセルの値を10倍し，セル F1，F2，…，F5 の順に代入する。

〔 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。ただし，問題冊子を切り離して利用することはできません。
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（時計型ウェアラブル端末は除く。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。