FE

# 平成24年度　春期
# 基本情報技術者試験
# 午後　問題

| 試験時間 | 13:00 ～ 15:30（2時間30分） |
|---|---|

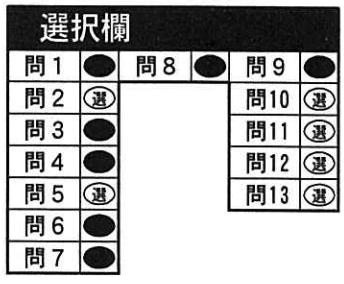
**注意事項**

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

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

5. 答案用紙の記入に当たっては，次の指示に従ってください。
   (1) 答案用紙は光学式読取り装置で読み取った上で採点しますので，B又はHBの黒鉛筆で答案用紙の**マークの記入方法**のとおりマークしてください。マークの濃度がうすいなど，**マークの記入方法**のとおり正しくマークされていない場合は読み取れません。特にシャープペンシルを使用する際には，マークの濃度に十分注意してください。訂正の場合は，あとが残らないように消しゴムできれいに消し，消しくずを残さないでください。
   (2) **受験番号欄**に**受験番号**を，**生年月日欄**に**受験票の生年月日**を記入及びマークしてください。答案用紙の**マークの記入方法**のとおり記入及びマークされていない場合は，採点されないことがあります。生年月日欄については，受験票の生年月日を訂正した場合でも，訂正前の生年月日を記入及びマークしてください。

〔問1，問3，問4，問6，問7，問9を選択した場合の例〕




   (3) **選択した問題**については，右の例に従って，**選択欄**の**問題番号**の (選) をマークしてください。答案用紙の**マークの記入方法**のとおりマークされていない場合は，採点されません。問1～問7について6問以上マークした場合は，はじめの5問を採点します。問9～問13について2問以上マークした場合は，はじめの1問を採点します。
   (4) **解答**は，次の例題にならって，**解答欄**にマークしてください。答案用紙の**マークの記入方法**のとおりマークされていない場合は，採点されません。

〔例題〕　次の［　　］に入れる正しい答えを，解答群の中から選べ。

春の情報処理技術者試験は，［ a ］月に実施される。

解答群　　ア　2　　イ　3　　ウ　4　　エ　5

正しい答えは“ウ　4”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。



C　COBOL　Java　アセンブラ　表計算

〔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 | 浮動小数点数 |
| 問 2 | ソフトウェア | コンパイラの最適化 |
| 問 3 | データベース | 社員食堂の利用記録データベースの設計と運用 |
| 問 4 | ネットワーク | データ転送時のフロー制御 |
| 問 5 | ソフトウェア設計 | 受験者数の集計リスト作成 |
| 問 6 | プロジェクトマネジメント | 設計工程での進捗管理 |
| 問 7 | 経営・関連法規 | 正味現在価値による投資採算性の評価 |

●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 | ビットの検査 |

●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） | 会議時間の調整 |
| 問 10 | ソフトウェア開発（COBOL） | 遊園地の入園者情報の集計 |
| 問 11 | ソフトウェア開発（Java） | 試験の成績管理 |
| 問 12 | ソフトウェア開発（アセンブラ） | 数字列の加算 |
| 問 13 | ソフトウェア開発（表計算） | 図書管理及び図書推薦 |

## 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**<br>▼ **処理** | 単岐選択処理を示す。<br>**条件式**が真のときは**処理**を実行する。 |
| | ▲ **条件式**<br>**処理 1**<br>▼ **処理 2** | 双岐選択処理を示す。<br>**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**<br>■ **処理** | 前判定繰返し処理を示す。<br>**条件式**が真の間，**処理**を繰り返し実行する。 |
| | ■ **処理**<br>■ **条件式** | 後判定繰返し処理を示す。<br>**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**<br>■ **処理** | 繰返し処理を示す。<br>開始時点で**変数**に**初期値**（式で与えられる）が格納され，**条件式**が真の間，**処理**を繰り返す。また，繰り返すごとに，**変数**に**増分**（式で与えられる）を加える。 |

〔演算子と優先順位〕

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

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

〔論理型の定数〕

```
true, false
```

次の**問1**から**問7**までの7問については，この中から5問を選択し，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**　浮動小数点数に関する次の記述を読んで，設問1，2に答えよ。

(1)　$\alpha \times 2^{\beta}$の形で表記される浮動小数点数を，図1に示す32ビット単精度浮動小数点形式（以下，単精度表現という）で表現する。ここで，$\alpha$と$\beta$は次の条件を満たすものとする。

$\alpha = 0$，又は $1 \leqq |\alpha| < 2$

$-126 \leqq \beta \leqq 127$

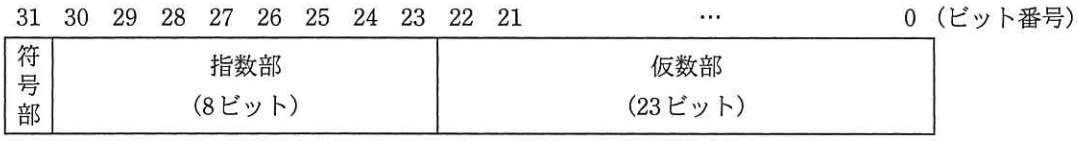



**図1　32ビット単精度浮動小数点形式**

①　符号部（ビット番号31）

$\alpha$の値が正のとき0，負のとき1が入る。

②　指数部（ビット番号30～23）

$\beta$の値に127を加えた値が2進数で入る。

③　仮数部（ビット番号22～0）

$|\alpha|$の整数部分1を省略し，残りの小数部分が，ビット番号22に小数第1位が来るような2進数で入る。

ただし，$\alpha$の値が0の場合，符号部，指数部，仮数部ともに0とする。

(2)　例えば，10進数の0.75を2進数で表すと，$(0.11)_2$となる。これは$(1.1)_2 \times 2^{-1}$と表記でき，単精度表現では，図2のとおり，符号部は$(0)_2$，指数部は$-1$に127を加えて$(01111110)_2$となり，仮数部は$(1.1)_2$の小数部分が入るので，$(100\cdots0)_2$となる。ここで，00…0は0が連続していることを表す。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | … | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | … | 0 |

**図２　0.75 の単精度表現**

**設問１**　次の単精度表現が表す数値として正しい答えを，解答群の中から選べ。

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | … | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | … | 0 |

解答群

ア　$3\times2^{-125}$　　イ　$3\times2^{-122}$　　ウ　$3\times2^{5}$　　エ　$3\times2^{132}$

オ　$11\times2^{-125}$　　カ　$11\times2^{-122}$　　キ　$11\times2^{5}$　　ク　$11\times2^{132}$

**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。

二つの浮動小数点数 A と B の減算と乗算を行う。

A の単精度表現

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | … | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | … | 0 |

B の単精度表現

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | … | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | … | 0 |

(1)　減算 A－B を，次の手順①～③で行う。

①　指数部の値を大きい方に合わせる。A が $(1.01)_2\times2^5$ であることから，B を $(\boxed{\text{a}})_2\times2^5$ とする。

②　減算を行う。

$((1.01)_2-(\boxed{\text{a}})_2)\times2^5=(1.0)_2\times2^{\boxed{\text{b}}}$

③　②の結果を単精度表現する。その結果は［　c　］となる。

(2)　乗算 A×B の結果は $(\boxed{\text{d}})_2\times2^9$ となる。

aに関する解答群

ア　0.011　　イ　0.101　　ウ　0.11　　エ　1.01

オ　1.1

bに関する解答群

ア　3　　イ　4　　ウ　5　　エ　6

オ　131　　カ　132

cに関する解答群

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | … | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ア | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| イ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| ウ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| エ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| オ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| カ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| キ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | … | 0 |

dに関する解答群

ア　1.0　　イ　1.11　　ウ　1.1101　　エ　1.111

オ　1.1111

**問2**　コンパイラの最適化に関する次の記述を読んで，設問1～3に答えよ。

コンパイラとは，プログラム言語で記述された原始プログラムを翻訳して目的プログラムを生成するためのソフトウェアである。コンパイラの機能の一つに最適化がある。最適化では，原始プログラムを翻訳する過程で，プログラムの実行時間を短くするために原始プログラムの構造を変換する。最適化の方法の例を表1に示す。

**表1　最適化の方法の例**

| 最適化の方法 | 内容 |
|---|---|
| 関数のインライン展開 | 関数を呼び出す箇所に，呼び出される関数のプログラムを展開する。 |
| 共通部分式の削除 | 同じ式が複数の箇所に存在し，それらの式で使用している変数の値が変更されず，式の値が変化しないとき，その式の値を作業用変数に格納する文を追加し，複数の同じ式をその作業用変数で置き換える。 |
| 定数の畳込み | プログラム中の定数同士の計算式を，その計算結果で置き換える。 |
| 定数伝播 | 変数を定数で置き換える。 |
| 無用命令の削除 | プログラムの実行結果に影響しない文を削除する。 |
| ループ内不変式の移動 | ループ中で値の変化しない式があるとき，その式をループの外に移動する。 |
| ループのアンローリング | ループ中の繰返しの処理を展開する。 |

擬似言語の形式で記述したプログラムの一部（以下，プログラム1という）に対して，表1の最適化の方法を複数組み合わせて最適化した例を，表2に示す。

〔プログラム1〕

```
i: 0, i ≦ 1, 1
  • x[i] ← y[i] + m + n
  • v[i] ← w[i] + m + n
```

表2　プログラム1の最適化の例

| 最適化の方法 | プログラム1の変換 |
|---|---|
| ① | ■ i: 0, i ≦ 1, 1<br>　• t ← m + n<br>　• x[i] ← y[i] + t<br>　• v[i] ← w[i] + t<br>■ |
| ② | • t ← m + n<br>■ i: 0, i ≦ 1, 1<br>　• x[i] ← y[i] + t<br>　• v[i] ← w[i] + t<br>■ |
| ③ | • t ← m + n<br>• x[0] ← y[0] + t<br>• v[0] ← w[0] + t<br>• x[1] ← y[1] + t<br>• v[1] ← w[1] + t<br>• i ← 2 |

**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。

表2において，最適化の方法を①，②，③の順で適用するとき，②で適用される最適化の方法は［　a　］であり，③で適用される最適化の方法は［　b　］である。

a，bに関する解答群

ア　関数のインライン展開
イ　共通部分式の削除
ウ　定数の畳込み
エ　定数伝播
オ　無用命令の削除
カ　ループ内不変式の移動
キ　ループのアンローリング

**設問2**　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

関数のインライン展開の例を図1に示す。図1の関数 func をインライン展開した場合，実引数 10 の値が仮引数 p に代入され，その値 10 が関数 func 内の変数 p の値になる（図1①）。次に，［　c　］の方法を適用することによって，条件式 p ≧ 10 の判定結果や p を含んだ計算式の計算結果がコンパイル時に確定する（図1②）。その結果，func(10)を値15に置き換えることができる（図1③）。

なお，#w はコンパイラが最適化をしたときに生成した整数型の変数である。

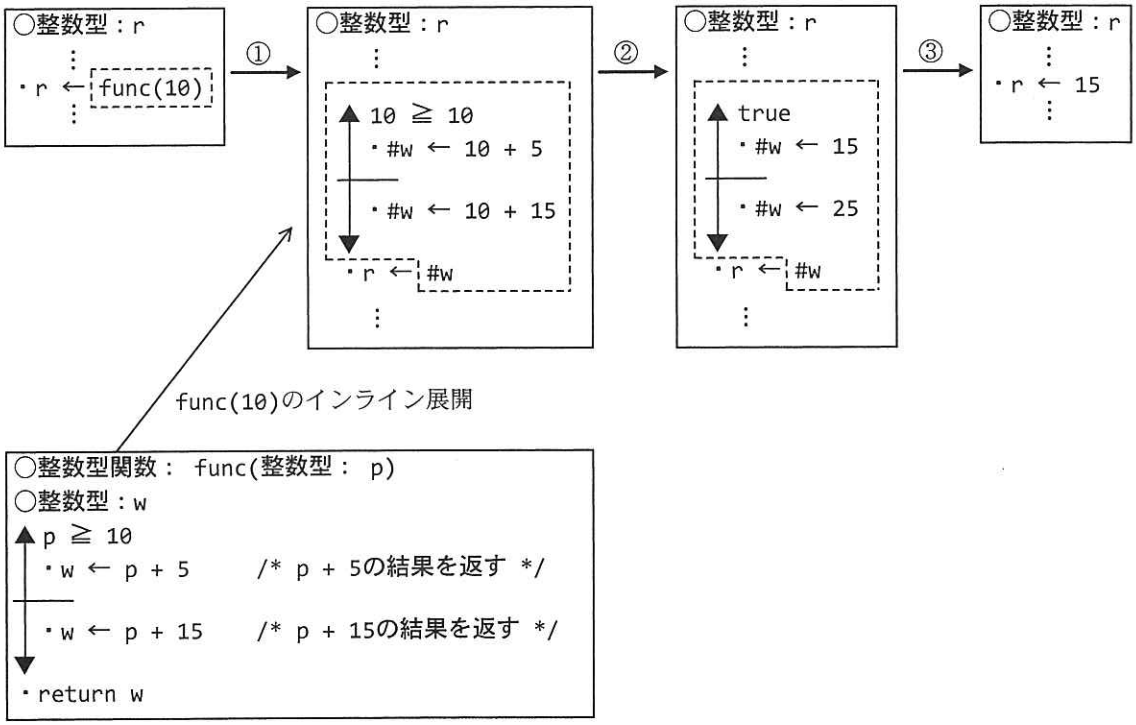



**図1　関数のインライン展開の例**

cに関する解答群

ア　共通部分式の削除
イ　定数の畳込み
ウ　定数伝播
エ　無用命令の削除
オ　ループ内不変式の移動
カ　ループのアンローリング

**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。

最適化をしないときと最適化をしたときとで浮動小数点数の演算の結果が異なる場合がある。プログラム1に対して表2の最適化をしなかったものと，最適化をしたものとに，y[0]＝307000000.0，y[1]＝305000000.0，m＝−303000000.0，n＝4.0を与えて実行した。その結果のx[0]，x[1]が次のとおりになった。ここで，同一優先順位の算術演算子は，左から順に演算する。

最適化をしないとき：　x[0]＝4000004.0，x[1]＝2000004.0
最適化をしたとき：　x[0]＝4000000.0，x[1]＝2000000.0

この実行結果が異なる原因としては，［　d　］の方法の適用によって演算順序が変化したことで，［　e　］が発生したからである。ここで，浮動小数点数の演算は単精度（仮数部は23ビット）で計算しているものとする。

dに関する解答群

ア　関数のインライン展開
イ　共通部分式の削除
ウ　定数の畳込み
エ　定数伝播
オ　無用命令の削除
カ　ループ内不変式の移動
キ　ループのアンローリング

eに関する解答群

ア　桁あふれ　　イ　桁落ち　　ウ　情報落ち　　エ　丸め誤差

**問3**　社員食堂の利用記録データベースの設計と運用に関する次の記述を読んで，設問1～4に答えよ。

A社では，社員証のICカード化に伴い，社員証を用いた社員食堂の精算システムを構築することにした。トレーに載せた料理を精算機の前に置くと，料理皿に埋め込まれたICチップのデータから料金が計算され，合計金額が表示される。合計金額を確認した後に社員証をかざすと，精算ができる。精算データはデータベースに記録され，1か月分の精算額が，まとめて翌月の給料から引き落とされる。

A社のシステム部門では，精算データを記録するデータベースとして，当初，図1に示す表を設計した。

精算表

| 社員番号 | 日付 | 精算額 |
|---|---|---|
| 050221 | 20120310 | 380 |

**図1　表構成とデータの格納例**

精算システムの機能に関して関係者にヒアリングした結果，給料からの引落し額の算出以外にも次に示す四つの要望が挙がった。

要望1：ある社員の，ある日の精算の明細を表示できること

要望2：ある日の売上合計額（精算額の合計）を算出できること

要望3：料理の一覧を表示できること

要望4：ある日の，ある料理の販売皿数を算出できること

そこで，四つの要望に対応できるように図2のとおり，1回の精算に対して一つの精算コードを割り当てた三つの表で構成するように設計を変更した。下線付きの項目は主キーを表す。

精算表

| 精算コード | 社員番号 | 日付 | 精算額 |
|---|---|---|---|
| 03100186 | 050221 | 20120310 | 380 |

明細表

| 精算コード | 料理コード | 皿数 |
|---|---|---|
| 03100186 | 0001 | 1 |
| 03100186 | 0002 | 1 |

料理表

| 料理コード | 料理名 | 単価 |
|---|---|---|
| 0001 | ごはん | 100 |
| 0002 | 肉じゃが | 280 |

**図２　変更後の表構成とデータの格納例**

**設問１**　図１に示した表構成のままでも対応できる要望として正しい答えを，解答群の中から選べ。

解答群

ア　要望1　　イ　要望2　　ウ　要望3　　エ　要望4

**設問２**　料理名が“肉じゃが”の単価に誤りがあることが判明したので，購入者に差額を返金することになった。“肉じゃが”購入者の社員番号と購入皿数を求める。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 精算表.社員番号, SUM(明細表.皿数) AS 購入皿数
  FROM 料理表, 精算表, 明細表
  WHERE [                                    ]
```

解答群

ア
```
精算表.精算コード = 明細表.精算コード AND
明細表.料理コード = (SELECT 料理表.料理コード FROM 料理表
                      WHERE 料理表.料理名 = '肉じゃが')
```

イ
```
精算表.精算コード = 明細表.精算コード AND
明細表.料理コード = 料理表.料理コード
GROUP BY 精算表.社員番号
HAVING 料理表.料理名 = '肉じゃが'
```

ウ
```
精算表.精算コード = 明細表.精算コード AND
明細表.料理コード = 料理表.料理コード AND
料理表.料理名 = '肉じゃが'
```

エ
```
精算表.精算コード = 明細表.精算コード AND
明細表.料理コード = 料理表.料理コード AND
料理表.料理名 = '肉じゃが'
GROUP BY 精算表.社員番号
```

**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。

A社の健康管理部門から，精算時に料理の合計カロリーを表示する機能と，ある社員の，ある期間における1回の精算当たりの平均カロリーを求める機能の追加を要望された。このため，料理表にカロリーの列を追加することにした。

しかし，平均カロリーを求めるには［　a　］を結合しなければならないので，多くの検索と計算の処理が必要となることが予想できた。精算時に合計カロリーを計算するので，その情報を記録しておけば，検索及び計算量は少なくなる。そこで，［　b　］に，精算単位の合計カロリーの列を追加することにした。これによって，前述の平均カロリーを求める場合は，［　b　］だけを参照すればよいので，処理の高速化が期待できる。

解答群

ア　精算表　　イ　精算表と明細表　　ウ　精算表と料理表
エ　精算表と明細表と料理表　　オ　明細表　　カ　料理表
キ　料理表と明細表

**設問4**　設問3のカロリーに関する機能を追加した後，食堂利用者にカロリーを意識して料理を選んでもらうために，人気料理とそのカロリーを掲示することにした。販売皿数の多い順に，料理名，カロリー及び販売皿数を求める。正しい SQL 文を，解答群の中から選べ。

解答群

ア
```
SELECT 料理表.料理名, 料理表.カロリー, COUNT(明細表.皿数) AS 販売皿数
    FROM 料理表, 明細表 WHERE 料理表.料理コード = 明細表.料理コード
    GROUP BY 料理表.料理名, 料理表.カロリー
    ORDER BY 販売皿数 DESC
```

イ
```
SELECT 料理表.料理名, 料理表.カロリー, COUNT(明細表.皿数) AS 販売皿数
    FROM 料理表, 明細表 WHERE 料理表.料理コード IN (SELECT
      明細表.料理コード FROM 明細表 WHERE 明細表.皿数 IS NOT NULL)
    GROUP BY 料理表.料理名, 料理表.カロリー
    ORDER BY 販売皿数 DESC
```

ウ
```
SELECT 料理表.料理名, 料理表.カロリー, SUM(明細表.皿数) AS 販売皿数
    FROM 料理表, 明細表 WHERE 料理表.料理コード = 明細表.料理コード
    GROUP BY 料理表.料理名, 料理表.カロリー
    ORDER BY 販売皿数 DESC
```

エ
```
SELECT 料理表.料理名, 料理表.カロリー, SUM(明細表.皿数) AS 販売皿数
    FROM 料理表, 明細表 WHERE 料理表.料理コード IN (SELECT
      明細表.料理コード FROM 明細表 WHERE 明細表.皿数 IS NOT NULL)
    GROUP BY 料理表.料理名, 料理表.カロリー
    ORDER BY 販売皿数 DESC
```

**問4**　データ転送時のフロー制御に関する次の記述を読んで，設問に答えよ。

端末Aから端末Bにデータを転送する。端末Aでは，データを1kバイト単位に分割し，分割したデータをそれぞれ一つずつのパケットに格納して送信する。パケット一つの大きさは1.2kバイトである。端末Bでは，受信したパケットを受信バッファに格納し，受信処理として，パケットの整合性の検査とパケットからのデータ抽出を行う。その後，パケットを正しく受信したことを知らせるために，端末AにACKを送信する。受信処理後は直ちに受信バッファの再利用が可能となる。端末Bの受信バッファの大きさは1.2kバイトである。端末Aでは，端末Bの受信バッファの大きさは判明しているが，端末Bでの各処理に掛かる時間は分からない。

端末Aは，送信したパケットに対応する端末BからのACKを受信することで，端末Bの受信バッファに空きができたことを検知し，次のパケットを送信する。

端末Aで，パケット1個の送信に掛かる時間は10ミリ秒，ACK受信に掛かる時間は0.5ミリ秒，ACKの受信を完了してから次のパケットが送信可能になるまでに掛かる時間は0.5ミリ秒である。

端末Aと端末Bとの間の通信の様子と，端末Aでの各処理に掛かる時間を，図1に示す。

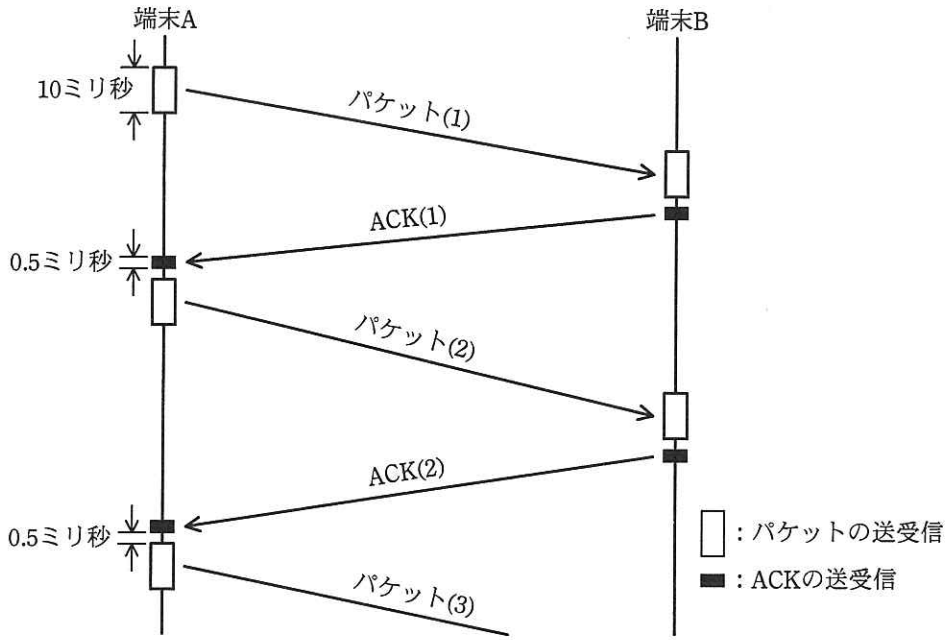



**図1　端末Aと端末Bとの間の通信の様子**

パケット 1 個の送信を完了してから，対応する ACK の受信を開始するまでに掛かる時間は一定で，その時間が 100 ミリ秒であったとすると，データ 1M バイトを送信し，次のパケットが送信可能となるまでの時間（以下，データ 1M バイト当たりの転送時間という）は［ a ］秒である。ここで，1M バイトは 1,000k バイトとし，パケットや ACK は確実に相手に届くものとする。

データの転送に掛かる時間を短縮するために，端末 B の受信バッファを 2.4k バイトに拡大した。

端末 A は，端末 B の受信バッファに空きがあることが確かなときは，送信したパケットに対応する ACK の受信を待たずに次のパケットを送信することができる。すなわち，“送信済みのパケット数 － 受信済みの ACK 数”が 1 以下であれば，端末 A は次のパケットを送信できる。パケット 1 個の送信を完了してから次のパケットが送信可能になるまでに掛かる時間は 0.5 ミリ秒であり，その他に掛かる時間は受信バッファの大きさが 1.2k バイトのときと同じである。

パケットの送信と ACK の受信，及びパケットの受信と ACK の送信は，並行して行うことができる。このときの通信の様子を，図 2 に示す。

このとき，データ 1M バイト当たりの転送時間は［ b ］秒である。

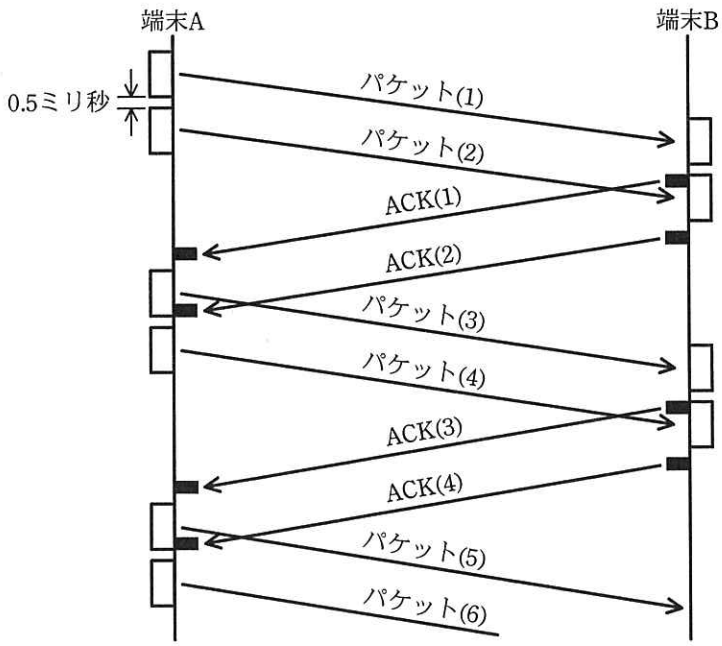



図 2　受信バッファを拡大したときの通信の様子

受信バッファを更に拡大することで，データの転送に掛かる時間を短縮することを考える。受信バッファの大きさを（1.2 × n）k バイトとすると，“送信済みのパケット数 − 受信済みの ACK 数”が［ c ］ときは，端末 A は端末 B の受信バッファにパケット 1 個分以上の空きがあることが分かるので，次のパケットを送信することができる。

各処理に掛かる時間が図 1 及び図 2 のとおりであり，パケット 1 個の送信を完了してから，対応する ACK の受信を開始するまでに掛かる時間は一定で，その時間が 100 ミリ秒であったとすると，データ 1M バイト当たりの転送時間は［ d ］秒まで短くすることができ，このときの最小の受信バッファの大きさは［ e ］k バイトである。

**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，d に関する解答群

| | | | | |
|---|---|---|---|---|
| ア　8.3 | イ　8.8 | ウ　10 | エ　10.5 | オ　46.3 |
| カ　50 | キ　55.5 | ク　92.5 | ケ　100 | コ　111 |

c に関する解答群

| | | | |
|---|---|---|---|
| ア　n 以下の | イ　n 以上の | ウ　n 未満の | エ　n より大きい |

e に関する解答群

| | | | |
|---|---|---|---|
| ア　9 | イ　10 | ウ　10.8 | エ　11 |
| オ　12 | カ　13.2 | キ　14.4 | |

**問5**　受験者数の集計リスト作成に関する次の記述を読んで，設問1～3に答えよ。

C大学では，今年度と前年度の受験者データを用いて，出身校ごとの受験者数を集計した出身校別受験者数リスト（以下，出身校リストという）を作成する。

〔受験者データ及び出身校リストの説明〕

(1)　今年度と前年度の受験者データは，それぞれ今年度受験者ファイルと前年度受験者ファイルの二つの順ファイルに保存されている。これらのレコード様式を図1に示す。

| 受験番号 | 姓 | 名 | 学校コード | 高校卒業年度 |
|---|---|---|---|---|

**図1　今年度受験者ファイル及び前年度受験者ファイルのレコード様式**

(2)　今年度受験者ファイル及び前年度受験者ファイルのレコードは，どちらも学校コードの昇順に並んでいる。

(3)　図2に示す例のように，出身校リストの印字項目は，順位，学校名，今年度受験者数，前年度受験者数である。印字の順序は，今年度受験者数の降順とし，今年度受験者数が同数の場合は，前年度受験者数の降順，それも同数の場合は，学校コードの昇順とする。

順位は，今年度受験者数の多い方から1，2，3，…と採番した番号である。今年度受験者数が同じ場合は同順位とし，次の順位は，同順位の学校数分だけ加算した順位とする。

学校名は，学校名ファイルから得る。学校名ファイルのレコード様式を図3に示す。

出身校別受験者数リスト

| 順位 | 学校名 | 今年度受験者数 | 前年度受験者数 |
|---|---|---|---|
| 1 | 東南高校 | 82 | 62 |
| 2 | 東西高校 | 80 | 96 |
| 2 | 西北高校 | 80 | 68 |
| 4 | 南西高校 | 73 | 64 |
| 5 | 北北学園 | 60 | 70 |
| 5 | 南南学園 | 60 | 0 |
| | その他計 | 2666 | 2723 |
| | 合計 | 3101 | 3083 |

**図２　出身校リストの例**

| 学校コード | 学校名 |
|---|---|

**図３　学校名ファイルのレコード様式**

(4)　図２に示す例のように，出身校リストには，１位から指定した順位（以下，指定順位という）までを印字し，指定順位よりも下位の学校の受験者数は，それらを合計して“その他計”として印字する。

〔出身校リスト作成処理の説明〕

図４に，出身校リスト作成処理の流れと各ファイルのレコード様式を示す。

(1)　集計処理では，今年度受験者ファイルと前年度受験者ファイルから，それぞれ学校ごとに受験者数を集計したファイル１及びファイル２を作成する。

(2)　突合せ処理では，学校コードをキーとして，ファイル 1 とファイル 2 の突合せを行い，ファイル３を作成する。この突合せ処理の中では，整列は行わない。

(3)　整列処理では，ファイル３を整列し，ファイル４を作成する。

(4)　順位付け処理では，ファイル 4 の各レコードの順位付けを行い，ファイル 5 を作成する。

(5)　リスト作成処理では，ファイル 5 から，出身校リストを作成する。学校ごとの明細行の印字は，ファイル５の１レコードの内容から１行ずつ，指定順位まで印字する。このとき，学校名は，学校コードをキーにして学校名ファイルから読み込む。指定順位よりも下位のレコードは，それらの受験者数を合計して“その他計”として印字する。

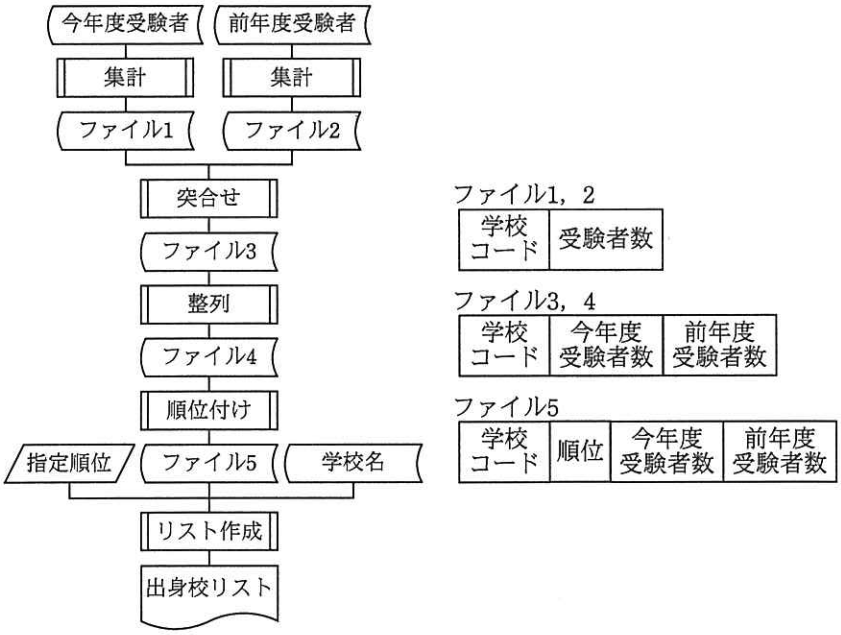



**図４　出身校リスト作成処理の流れと各ファイルのレコード様式**

**設問１**　表１は，図４中の突合せ処理における条件に合致するレコードの有無と出力レコードの各項目の内容である。表１中の［　　］に入れる正しい答えを，解答群の中から選べ。

**表１　突合せ処理での出力レコードの各項目の内容**

| レコードの有無 | | 学校コード | 今年度受験者数 | 前年度受験者数 |
|---|---|---|---|---|
| ファイル１ | ファイル２ | | | |
| あり | あり | | | |
| あり | なし | | a | |
| なし | あり | b | | |

**注記**　網掛けの部分は表示していない。

a，bに関する解答群

| | | |
|---|---|---|
| ア | 0 | 0 |
| イ | 0 | ファイル1の当該項目 |
| ウ | 0 | ファイル2の当該項目 |
| エ | ファイル1の当該項目 | 0 |
| オ | ファイル1の当該項目 | ファイル2の当該項目 |
| カ | ファイル2の当該項目 | 0 |
| キ | ファイル2の当該項目 | ファイル1の当該項目 |

**設問2**　図4中の整列処理に最低限必要な整列キー項目及び整列順（昇順又は降順）の組の並びとして正しい答えを，解答群の中から選べ。ここで，キー項目及び整列順の組は，（［整列キー項目］；［整列順］）で表す。また，コンマで区切られた組の並びは，左の方が整列の優先度が高いことを表す。

解答群

ア　（学校コード；昇順），（今年度受験者数；昇順），（前年度受験者数；昇順）

イ　（学校コード；昇順），（前年度受験者数；降順），（今年度受験者数；降順）

ウ　（今年度受験者数；降順），（前年度受験者数；降順），（学校コード；降順）

エ　（今年度受験者数；降順），（前年度受験者数；降順），（学校コード；昇順）

オ　（今年度受験者数；降順），（前年度受験者数；降順）

カ　（今年度受験者数；昇順），（学校コード；昇順）

**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。

〔出身校リストへの印字項目追加の説明〕

図5に示す例のような，出身校リストに前年度順位を追加した出身校別受験者数リスト2（以下，出身校リスト2という）を作成することになり，処理の追加と変更を行うことになった。前年度に受験者がいなかった高校の前年度順位は空欄とする。

図６に出身校リスト２作成処理の流れと主なファイルのレコード様式を示す。

出身校別受験者数リスト2

| 順位 | 学校名 | 今年度受験者数 | 前年度順位 | 前年度受験者数 |
|---|---|---|---|---|
| 1 | 東南高校 | 82 | 8 | 62 |
| 2 | 東西高校 | 80 | 1 | 96 |
| 2 | 西北高校 | 80 | 4 | 68 |
| 4 | 南西高校 | 73 | 5 | 64 |
| 5 | 北北学園 | 60 | 3 | 70 |
| 5 | 南南学園 | 60 | | 0 |
| | その他計 | 2666 | | 2723 |
| | 合計 | 3101 | | 3083 |

**図５　出身校リスト２の例**

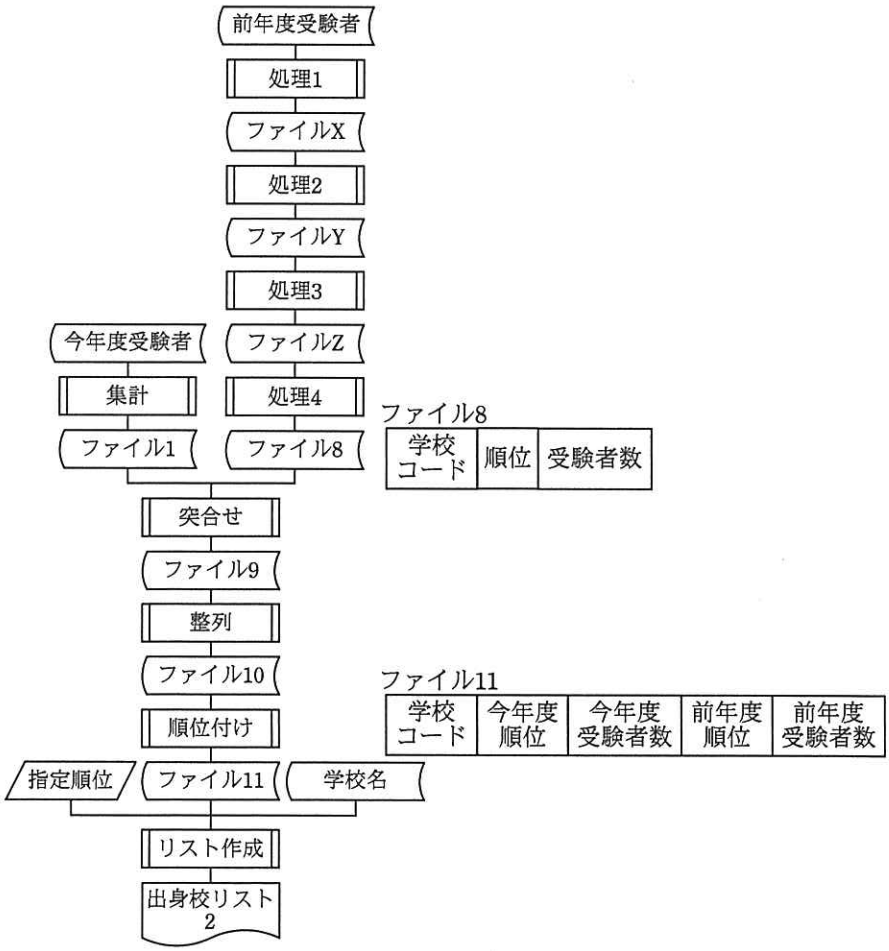



**図６　出身校リスト２作成処理の流れと主なファイルのレコード様式**

〔出身校リスト2作成処理の説明〕

(1) 図 6 の処理は，図 4 の同名の処理に，入出力のレコード様式への前年度順位の追加に対する変更を加えたものを用いる。

(2) 前年度受験者ファイルからファイル 8 を作成するまでの処理に用いられる処理 1～4 のうち，処理 2 は，［ c ］処理である。また，処理 4 は，［ d ］処理である。

(3) 変更になる突合せ処理の出力であるファイル 9 及びそれを整列した出力であるファイル 10 のレコード項目は，［ e ］である。

c，d に関する解答群

ア 学校コードの降順への整列　　イ 学校コードの昇順への整列

ウ 学校ごとの人数の集計　　エ 受験者数の降順への整列

オ 受験者数の昇順への整列　　カ 順位付け

キ リスト作成

e に関する解答群

ア 学校コード，今年度受験者数，前年度受験者数

イ 学校コード，今年度受験者数，前年度順位

ウ 学校コード，今年度受験者数，前年度順位，前年度受験者数

エ 学校コード，今年度順位，今年度受験者数，前年度受験者数

オ 学校コード，今年度順位，今年度受験者数，前年度順位，前年度受験者数

カ 学校コード，今年度順位，前年度受験者数

キ 学校コード，今年度順位，前年度順位

ク 学校コード，今年度順位，前年度順位，前年度受験者数

**問6**　設計工程で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。

チーム X は，あるシステム開発プロジェクトにおけるユーザインタフェース設計（以下，UI 設計という）を担当している。このプロジェクトでは，週単位でプロジェクトの進捗状況を把握し，計画値と実績値の比較分析によって，スケジュール遅延などに早期対応を行っている。

〔設計工程での進捗管理の説明〕

(1)　就業規則は，次のとおりである。

①　1日の勤務時間は8時間である。

②　週のうちで勤務が認められている日は，月曜日から金曜日までの 5 日間である。土曜日と日曜日は休日であり，勤務は禁止されている。

(2)　設計工程の開始前に，各メンバの UI 設計に要する工数を見積もり，設計工程の各週に分配する。分配した工数を計画工数といい，単位は時間である。

(3)　週単位で進捗状況を把握するための指標の一つとして，計画進捗率を用いる。計画進捗率は，次の計算式で求め，小数第2位を四捨五入する。

$$\text{計画進捗率} = \frac{\text{計画工数の累積}}{\text{計画工数の総合計}} \times 100$$

“計画工数の累積”とは，計画工数を設計の開始時点からその週まで累積した値である。

〔チームの状況〕

(1)　チーム X のメンバは A，B，C，D の 4 名であり，メンバ別の設計分担，計画工数及び計画進捗率は表1のとおりである。

表１　設計工程におけるメンバ別の設計分担，計画工数及び計画進捗率

| 設計分担 | メンバ | 第１週 | | 第２週 | | 第３週 | | 計画工数の総合計（時間） |
|---|---|---|---|---|---|---|---|---|
| | | 計画工数（時間） | 計画進捗率（%） | 計画工数（時間） | 計画進捗率（%） | 計画工数（時間） | 計画進捗率（%） | |
| UI設計１ | A | 0 | 0.0 | 25 | 50.0 | 25 | 100.0 | 50 |
| UI設計２ | B | 20 | 20.0 | 40 | 60.0 | 40 | 100.0 | 100 |
| UI設計３ | C | 20 | 20.0 | 40 | 60.0 | 40 | 100.0 | 100 |
| UI設計４ | D | 20 | 20.0 | 40 | 60.0 | 40 | 100.0 | 100 |
| 合計 | | 60 | 17.1 | 145 | 58.6 | 145 | 100.0 | 350 |

(2)　設計工程の期間は３週間であり，現在，第２週が終了したところである。

(3)　設計工程での第１週及び第２週における，各メンバが実際に作業した時間数（以下，実績工数という）は，表２のとおりである。第１週では，他業務の都合で実績工数が計画工数と異なるメンバがいたが，第２週は全メンバが計画工数どおりの工数をUI設計に充てた。

表２　各メンバの実績工数及び残り工数

単位　時間

| 設計分担 | メンバ | 計画工数の総合計 | 第１週の週末時点 | | 第２週の週末時点 | |
|---|---|---|---|---|---|---|
| | | | 実績工数の累積 | 残り工数 | 実績工数の累積 | 残り工数 |
| UI設計１ | A | 50 | 0 | 50 | 25 | 20 |
| UI設計２ | B | 100 | 25 | 75 | 65 | 35 |
| UI設計３ | C | 100 | 20 | 80 | 60 | 45 |
| UI設計４ | D | 100 | 15 | 85 | 55 | 40 |
| 合計 | | 350 | 60 | 290 | 205 | 140 |

①　“計画工数の総合計”は，表１の右端の欄の値に等しい。

②　“実績工数の累積”は，各週末時点で，各週の実績工数をその週まで累積した値である。

③　残り工数は，各週末時点で，各メンバの今後の作業に要する工数を見積もりし直したものである。

**設問1**　次の計算式による指標を実績進捗率といい，各週末時点で算出する。ここで，実績進捗率は，小数第2位を四捨五入した値である。計画進捗率と実績進捗率との対比によるチーム X の進捗判定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。

$$\text{実績進捗率} = \frac{\text{実績工数の累積}}{\text{実績工数の累積} + \text{残り工数}} \times 100$$

実績進捗率と計画進捗率が等しい場合は，計画どおりの進捗であり，実績進捗率が上回っている場合は，計画よりも進んでいる状況である。いずれの場合も，設計工程の進捗に問題がないと判定する。一方，実績進捗率が計画進捗率を下回っている場合は，スケジュール遅延などの問題が懸念される。

第1週の週末時点では，UI 設計作業に着手しているメンバの中で，実績進捗率が計画進捗率を下回っているメンバは［ a ］。第2週の週末時点では，［ b ］の2名であり，メンバ個別に判定した場合には，やや進捗遅れの懸念がある。次に，表2の最下行にある合計値によるチーム X 全体の実績進捗率を用いて，［ c ］と判定した。

a に関する解答群

ア　B である　　イ　C である　　ウ　D である　　エ　いない

b に関する解答群

ア　A と B　　イ　A と C　　ウ　A と D
エ　B と C　　オ　B と D　　カ　C と D

cに関する解答群

ア　週ごとに実績進捗率が計画進捗率を下回る傾向にあり，このままではスケジュール遅延が生じる

イ　実績進捗率が，第1週の週末時点では計画進捗率と同じであり，第2週の週末時点では計画進捗率を上回っており，チームX全体では進捗に問題はない

ウ　実績進捗率が，第1週の週末時点では計画進捗率と同じであるが，第2週の週末時点では計画進捗率を下回っており，このままではスケジュール遅延が懸念される

エ　実績進捗率が，第1週の週末時点は計画進捗率を下回っているが，第2週の週末時点では計画進捗率を上回っており，チーム X 全体では進捗が回復しているので問題はない

**設問2**　次の計算式による指標を計画実績工数比といい，各週末時点で算出する。ここで，計画実績工数比は，小数第3位を四捨五入した値である。計画実績工数比を用いた今後のスケジュール分析に関する次の記述中の［　　］に入れる適切な答えを，解答群の中から選べ。

$$\text{計画実績工数比} = \frac{\text{実績工数の累積＋残り工数}}{\text{計画工数の総合計}}$$

計画実績工数比は，プロジェクトメンバの増減がないとすると，今後のスケジュールの予測に利用できる。計画実績工数比が 1.00 の場合は，今後も計画どおりのスケジュールで進捗可能と判定できる。1.00 よりも小さい場合は，スケジュール遅延の懸念はないと判定できる。1.00 よりも大きい場合は，今後のスケジュールに遅延が生じる可能性があると判定できる。

チーム X 全体の計画実績工数比は，第 1 週及び第 2 週の週末時点のそれぞれの値が［ d ］であり，チーム X 全体では計画した期間内に設計工程を終えることが可能であると判定した。次に，メンバ個別の計画実績工数比を週ごとに比較してみると，第 2 週の週末時点の値が第 1 週の週末時点の値よりも増加しているメンバ［ e ］名については，スケジュール遅延が懸念されると判定した。

d に関する解答群

| | | |
|---|---|---|
| ア 0.99 及び 0.95 | イ 0.99 及び 1.00 | ウ 1.00 及び 0.99 |
| エ 1.00 及び 1.00 | オ 1.01 及び 0.99 | カ 1.01 及び 1.00 |

e に関する解答群

| | | |
|---|---|---|
| ア 1 | イ 2 | ウ 3 |

**設問 3** 次の文は，スケジュール遅延の懸念に対する解決策について記述したものである。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。

第 2 週の週末時点での残り工数が第 3 週の計画工数を下回っているメンバのうち，その時点での実績進捗率が最も高い［ f ］に，スケジュール遅延の懸念があるメンバの UI 設計の一部を分担させることによって，スケジュール遅延を回避し，計画した期間内に全メンバの UI 設計を完了させることにした。

解答群

| | | | |
|---|---|---|---|
| ア A | イ B | ウ C | エ D |

**問7**　正味現在価値（Net Present Value）による投資採算性の評価に関する次の記述を読んで，設問1～3に答えよ。

日用品メーカのD社では，業務効率の向上及び2016年に予定されている新生産設備の稼働への対応を目的として，新たに生産管理システムの導入を検討している。生産管理システムの導入における投資採算性の評価を，企画課で行うことになった。企画課で試算したシステム導入に掛かる投資と効果は次のとおりである。

(1)　投資

生産管理システムの導入の方法として，次の2案を検討した。

①　業務効率向上のための投資と新生産設備の稼働対応の投資の両方を2012年に行う（以下，一括投資という）場合，7億5千万円が必要となる。

②　業務効率向上のための投資を2012年に，新生産設備の稼働対応の投資を2015年に行う（以下，分割投資という）場合，2012年に5億円，2015年に3億円が必要となる。

(2)　効果

(1)のどちらの場合でも，生産管理システムの導入によって，2013年からは年間2億円，2016年からは年間3億円のコスト削減が見込まれる。

D社では，投資採算性の評価に正味現在価値（以下，NPVという）を用いている。NPVは，キャッシュの現在価値という概念を使っている。現在価値の考え方を用いると，現在の100万円は1年後の100万円よりも価値が高い。なぜなら，現在の100万円を1年間預金すると利子がついて1年後には100万円よりも多くなるからである。利子率を0.05として複利計算すると，1年後の100万円は現在の約95万円，2年後の100万円は現在の約91万円と同じ価値ということになる。これがキャッシュの現在価値という概念である。

なお，NPVの計算では，利子率を割引率と呼ぶこともある。

**設問1**　NPVを使った投資採算性の評価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。

NPV を計算するときは，各年に増減するキャッシュの現在価値を計算する必要がある。割引率を r で複利計算し，n 年後のキャッシュを C と表すとき，その現在価値は，［ a ］と表せる。

なお，投資年については，n＝0 で計算する。

将来の各年におけるキャッシュの増減に対して，その現在価値を計算し，次に各年の現在価値を合計することで NPV が求められる。NPV が正のときは，その投資案は採算性があり，NPV が大きいほど採算性が高いと評価される。NPV が負のときは，その投資案は採算性がなく，投資するべきでないと評価される。NPV が正の投資案が二つ以上あるときは，NPV の最も大きな投資案を選択するべきである。

解答群

ア $\frac{C}{(1+r)^n}$　　イ $\frac{C}{(1-r)^n}$　　ウ $\frac{C}{r^n}$

エ $C\times(1+r)^n$　　オ $C\times(1-r)^n$　　カ $C\times r^n$

**設問 2**　生産管理システムの投資採算性の評価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。

企画課では，次の手順で生産管理システム導入の投資採算性の評価を行うことにした。一括投資する場合の生産管理システム導入の NPV を計算する表 1 と，分割投資する場合の NPV を計算する表 2 を作成した。

(1)　2012 年から 2020 年までの期間を対象に NPV を計算する。

(2)　各年のコスト削減額から減価償却費を差し引いて，各年の課税対象効果を求める。

(3)　減価償却費は，投資した翌年から 4 年間の定額償却（残存価額は 0 円）として求める。

(4)　法人税率を 40％とし，各年の課税対象効果から法人税として控除される法人税額を求める。

(5) 各年のコスト削減額から法人税額を差し引いて税引き後効果を求め，更に税引き後効果から各年の投資額を差し引いて各年の投資後効果（キャッシュ増減額）を求める。

(6) 割引率を 0.08 とし，各年の投資後効果の現在価値を計算する。

(7) (6)で計算した各年の現在価値を合計し，投資案の NPV を計算する。

表 1　一括投資する場合の生産管理システム導入の NPV 計算

単位　百万円

| 年 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 |
|---|---|---|---|---|---|---|---|---|---|
| 投資額 | 750 | | | | | | | | |
| 減価償却費 | 0 | 187.5 | 187.5 | 187.5 | 187.5 | 0 | 0 | 0 | 0 |
| コスト削減額 | 0 | 200 | 200 | 200 | 300 | 300 | 300 | 300 | 300 |
| 法人税額 | 0 | 5 | 5 | 5 | 45 | 120 | 120 | 120 | 120 |
| 税引き後効果 | 0 | 195 | 195 | 195 | 255 | 180 | 180 | 180 | 180 |
| 投資後効果 | −750 | b | | | | | | | |
| 割引率 | 0.08 | | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 |
| 現在価値 | −750 | | | | | | | | |
| NPV | 378.1 | | | | | | | | |

注記　網掛けの部分は表示していない。
現在価値は小数第 2 位を四捨五入している。

表 2　分割投資する場合の生産管理システム導入の NPV 計算

単位　百万円

| 年 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 |
|---|---|---|---|---|---|---|---|---|---|
| 投資額 | 500 | | | 300 | | | | | |
| 減価償却費 | 0 | 125 | 125 | c | 200 | 75 | 75 | 75 | 0 |
| コスト削減額 | 0 | 200 | 200 | | 300 | 300 | 300 | 300 | 300 |
| 法人税額 | 0 | 30 | 30 | | 40 | 90 | 90 | 90 | 120 |
| 税引き後効果 | 0 | 170 | 170 | | 260 | 210 | 210 | 210 | 180 |
| 投資後効果 | −500 | | | | | | | | |
| 割引率 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 | 0.08 |
| 現在価値 | −500 | 157.4 | 145.7 | −103.2 | 191.1 | 142.9 | 132.3 | 122.5 | 97.2 |
| NPV | | | | | | | | | |

注記　網掛けの部分は表示していない。
現在価値は小数第 2 位を四捨五入している。

企画課では，表1と表2の結果から［ d ］と判断した。

bに関する解答群

| ア | イ | ウ | エ |
|---|---|---|---|
| −195 | −195 | −195 | −195 |
| 0.08 | 0.08 | 0.08 | 0.08 |
| −210.6 | −195 | −180.6 | 0 |

| オ | カ | キ | ク |
|---|---|---|---|
| 195 | 195 | 195 | 195 |
| 0.08 | 0.08 | 0.08 | 0.08 |
| 0 | 180.6 | 195 | 210.6 |

cに関する解答群

| ア | イ | ウ | エ |
|---|---|---|---|
| 125 | 125 | 200 | 200 |
| 200 | 300 | 200 | 300 |
| 30 | 70 | 0 | 40 |
| 170 | 230 | 16 | 260 |

dに関する解答群

ア　一括投資，分割投資のどちらの投資案にしてもよい

イ　一括投資，分割投資のどちらの投資案もするべきでない

ウ　一括投資するべきである

エ　分割投資するべきである

**設問3**　生産管理システムの NPV の再計算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。

なお，解答は重複して選んでもよい。

企画課で作成した表 1，表 2 を社内の関係部署に説明したところ，生産課から，生産管理システムに新たな機能を追加したいとの要望があった。新たな機能の追加には，2012 年に 5 千万円の追加投資が必要となる。その場合，生産管理システムの機能が向上し，追加効果は年間で5百万円の見込みである。

機能を追加した場合の NPV を企画課で計算した結果，一括投資の場合の NPV は 3 億 6,190 万円，分割投資の場合の NPV は 3 億 7,000 万円であった。企画課では，これらの結果と表 1，表 2 から，生産課の要望については［ e ］と判断した。

また，経理課から，D 社では情報システムの減価償却期間を 4 年間ではなく 5 年間で行っているので，表 1，表 2 もそのように修正するべきであるとのコメントがあった。減価償却期間を 5 年に変更すると，表 1 の各年の減価償却費は表 3 のように変わる。

**表 3　減価償却期間を 5 年に変更したときの表 1 の各年の減価償却費**

**単位**　百万円

| 年 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 |
|---|---|---|---|---|---|---|---|---|---|
| 減価償却費 | 0 | 150 | 150 | 150 | 150 | 150 | 0 | 0 | 0 |

表 3 によると，2013 年から 2016 年までの各年の減価償却費は表 1 に比べて下がり，2017 年の減価償却費は表 1 に比べて上がり，2018 年以降の減価償却費は表 1 と同じであることが分かる。したがって，2013 年から 2016 年までの税引き後効果は［ f ］ことになり，2017 年の税引き後効果は［ g ］ことになる。2020 年までの減価償却費の総額は変わらないものの，現在価値の考え方を考慮すると，NPV は［ h ］と考えられる。これは，表 2 についても同様である。

企画課では，経理課のコメントに従って表 1 と表 2 を修正して，再度投資採算性を評価した。

e に関する解答群

ア　一括投資，分割投資のどちらの場合も受け入れるべきではない

イ　一括投資，分割投資のどちらの場合も受け入れるべきである

ウ　一括投資の場合だけ受け入れるべきである

エ　分割投資の場合だけ受け入れるべきである

f～h の解答群

ア　表 1 と変わらない　　イ　表 1 に比べて上がる　　ウ　表 1 に比べて下がる

次の**問8**は必須問題です。必ず解答してください。

**問8**　次のプログラムの説明及びプログラムを読んで，設問1～3に答えよ。

整数型関数 BitTest は，8 ビットのデータ中の指定したビット位置にあるビットの値を検査して，結果を返す。整数型関数 BitCount は，8 ビットのデータ中にある 1 のビットの個数を返す。

なお，本問において，演算子“&”，“|”は，二つの 8 ビット論理型データの対応するビット位置のビット同士について，それぞれ論理積，論理和を求め，8 ビット論理型で結果を得るものとする。また，"～"B という表記は，8 ビット論理型定数を表す。

〔プログラム 1 の説明〕

整数型関数 BitTest を，次のとおりに宣言する。

**○整数型関数：BitTest（8 ビット論理型：Data，8 ビット論理型：Mask）**

検査される 8 ビットのデータは入力用の引数 Data に，検査をするビット位置の情報は入力用の引数 Mask に，それぞれ格納されている。Mask 中のビットの値が 1 であるビット位置に対応した Data 中のビットを検査して，次の返却値を返す。ここで，Mask 中には 1 のビットが 1 個以上あるものとする。

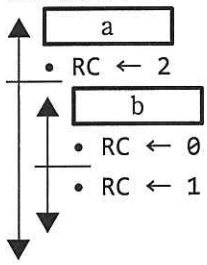
返却値　0：検査した全てのビットが 0
　　1：検査したビット中に 0 と 1 が混在
　　2：検査した全てのビットが 1

例えば，図 1 の例 1 では，Mask のビット番号 7～5 の 3 ビットが 1 であるので，Data のビット番号 7～5 の 3 ビットの値を検査し，0 と 1 が混在しているので返却値 1 を返す。例 2 では，Mask のビット番号 4 と 0 の 2 ビットが 1 であるので，Data のビット番号 4 と 0 の 2 ビットの値を検査し，どちらも 1 であるので返却値 2 を返す。


**図 1　BitTest の実行例**

〔プログラム1〕

```
○整数型関数：BitTest（8ビット論理型：Data，8ビット論理型：Mask）
○整数型：RC                    /* 返却値 */
```




```
• return RC                    /* RCを返却値として返す */
```

〔プログラム2，3の説明〕

整数型関数BitCountを，次のとおりに宣言する。

```
○整数型関数：BitCount（8ビット論理型：Data）
```

検査される8ビットのデータは入力用の引数Dataに格納されている。

このためのプログラムとして，基本的なアルゴリズムを用いたプログラム2と，処理効率を重視したプログラム3を作成した。

プログラム2，3中の各行には，ある処理系を想定して，プログラムの各行を1回実行するときの処理量（1, 2, …）を示してある。選択処理と繰返し処理の終端行の処理量は，それぞれの開始行の処理量に含まれるものとする。

なお，演算子“−”は，両オペランドを8ビット符号なし整数とみなして，減算を行うものとする。

〔プログラム2〕

(処理量)

```
        ○整数型関数：BitCount（8ビット論理型：Data）
        ○8ビット論理型：Work
        ○整数型：Count，Loop
1       • Work ← Data
1       • Count ← 0
4       ■ Loop: 0, Loop ＜ 8, 1
3         ▲ Workの最下位ビットが1
1           • Count ← Count + 1
1         • Workを右へ1ビット論理シフトする
2       • return Count                  /* Countを返却値として返す */
```

〔プログラム3〕

(処理量)

```
        ○整数型関数：BitCount（8ビット論理型：Data）
        ○8ビット論理型：Work
        ○整数型：Count
1       • Work ← Data
1       • Count ← 0
2       ■ Work 中に1のビットがある
1         • Count ← Count + 1
3         • Work ← Work & (Work − 1)   ←――― α
          ■
2       • return Count                  /* Count を返却値として返す */
```

**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　(Data & Mask) = "00000000"B　　イ　(Data & Mask) = Data

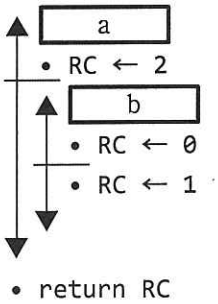
ウ　(Data & Mask) = Mask　　エ　(Data | Mask) = "00000000"B

オ　(Data | Mask) = Mask

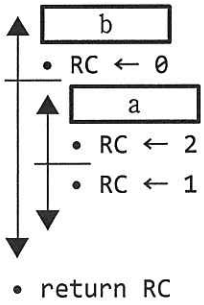
**設問2**　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

プログラム1は，Mask 中に1のビットが1個以上あることを前提としている。ここで，この前提を取り除いて，Mask 中の1のビットが0個の場合は返却値0を返すようにしたい。そのために，プログラム1の処理部分について，次の修正案①～③を考えた。ここで，修正案①は，プログラム1のままで何も変更しない。また，［ a ］と［ b ］には，設問1の正しい答えが入っているものとする。

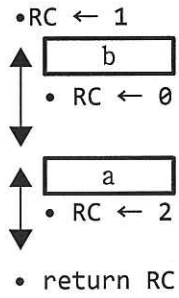
修正案①(変更なし)




修正案②




修正案③




これらの修正案のうち，正しく動作するのは［ c ］である。

解答群

ア　修正案①　　イ　修正案②　　ウ　修正案③

エ　修正案①及び②　　オ　修正案①及び③　　カ　修正案②及び③

**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。

プログラム2，3の処理効率について考えてみる。表1にプログラム2，3の処理量の比較結果を示す。

**表1　プログラム2，3の処理量の比較**

| | 最小 | 最大 |
|---|---|---|
| プログラム2 | 72 | ［ d ］ |
| プログラム3 | ［ e ］ | 54 |

プログラム3では，αの行での変数 Work の更新において効率の良いアルゴリズムが使われている。例えば，プログラム3で引数 Data の内容が "01101010"B であったとき，繰返し処理においてαの行の2回目の実行が終了した時点で変数 Work の内容は，"［ f ］"B になっている。このようなビット変換の処理によって，繰返し処理の繰返し回数は，検査されるデータ中の1のビットの個数と同じになる。

dに関する解答群

ア　80　　イ　88　　ウ　104　　エ　112

eに関する解答群

ア　6　　イ　10　　ウ　20　　エ　22

fに関する解答群

ア　00000011　　イ　00000110　　ウ　00001010

エ　01010000　　オ　01100000　　カ　10100000

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〕

S社では，通常勤務時間帯（8時30分～18時）中の各社員の予定を管理している。関数 allmem_free_slot は，通常勤務時間帯に会議を計画する場合に，出席して欲しい社員（以下，出席社員という）全員の予定が入っていない時間帯を調べるプログラムである。

(1) S社では，各社員の予定を30分単位の時間帯ごとに図1に示すスケジュール表で管理している。

時間帯番号：

| 0 | 1 | … | 17 | 18 |
|---|---|---|---|---|
| 8:30～9:00の予定 | 9:00～9:30の予定 | … | 17:00～17:30の予定 | 17:30～18:00の予定 |

**図1　スケジュール表**

① スケジュール表は大きさ19の整数型配列で，各要素の添字の値は時間帯番号に対応している。

② スケジュール表の各要素には，予定が入っている場合には0以外の値が，予定が入っていない場合には0が入る。

(2) 会議時間は30分以上であり，開始時刻と終了時刻は時間帯の区切りと一致する。

(3) 関数 allmem_free_slot 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。

num　　出席社員数

time_tbl　出席社員全員のスケジュール（スケジュール表へのポインタの配列）

slot_num　会議時間（30分単位の時間帯数）

free　　会議開催の候補時間帯（SLOT型の配列）

(4) 構造体を使った SLOT 型の定義は次のとおりである。

```
typedef struct {
    int s_start;      /* 開始時間帯番号 */
    int s_num;        /* 時間帯数 */
} SLOT;
```

(5) 候補時間帯 free には，出席社員全員の予定が入っていない，会議時間以上の連続する時間帯（以下，可能時間帯という）を全て格納する。最後に格納した要素の次の要素の開始時間帯番号には，−1 を格納する。例えば，出席社員 5 名（社員 A～E）の予定が図 2 のとおりであった場合，会議時間が 1 時間（slot_num = 2）のとき，可能時間帯の一覧は表 1 になる。

| 時刻： | 8:30 | 9:00 | 9:30 | 10:00 | 10:30 | 11:00 | 11:30 | 12:00 | 12:30 | 13:00 | 13:30 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 時間帯番号： | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 社員A | ○ | ○ | ○ | ○ | × | × | ○ | × | × | ○ | |
| 社員B | ○ | ○ | ○ | × | × | ○ | ○ | × | × | ○ | |
| 社員C | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × | ○ | |
| 社員D | ○ | ○ | ○ | × | × | ○ | ○ | × | × | ○ | |
| 社員E | ○ | ○ | ○ | × | × | ○ | ○ | × | × | ○ | |

| 時刻： | 13:30 | 14:00 | 14:30 | 15:00 | 15:30 | 16:00 | 16:30 | 17:00 | 17:30 | 18:00 |
|---|---|---|---|---|---|---|---|---|---|---|
| 時間帯番号： | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | |
| 社員A | × | ○ | ○ | × | ○ | × | ○ | ○ | ○ | |
| 社員B | × | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | |
| 社員C | × | ○ | ○ | × | ○ | × | ○ | ○ | ○ | |
| 社員D | ○ | ○ | ○ | × | ○ | × | ○ | ○ | ○ | |
| 社員E | × | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | |

注記　“○”は予定が入っていないことを表し，“×”は予定が入っていることを表す。

**図 2　出席社員全員のスケジュールの例**

**表 1　可能時間帯の一覧の例**

| 可能時間帯 | 開始時間帯番号 | 時間帯数 |
|---|---|---|
| 8:30 ～ 10:00 | 0 | 3 |
| 14:00 ～ 15:00 | 11 | 2 |
| 16:30 ～ 18:00 | 16 | 3 |

(6) 関数 allmem_free_slot で使用している関数 search_free_slot の仕様は次のとおりである。

機能：スケジュール表から，予定が入っていない，時間帯数以上の連続した時間帯を全て探し出し，空き時間帯に格納する。最後に格納した空き時間帯の要素の次の要素の開始時間帯番号には，−1を格納する。

引数：sch　　スケジュール表

slot_num　　時間帯数

free　　空き時間帯（SLOT型の配列）

〔プログラム1〕

```
#define NUM_DAY 19      /* 1日の時間帯数 */

typedef struct {
    int s_start;        /* 開始時間帯番号 */
    int s_num;          /* 時間帯数 */
} SLOT;

void allmem_free_slot(int, int *[], int, SLOT []);
void search_free_slot(int [], int, SLOT []);

void allmem_free_slot(int num, int *time_tbl[], int slot_num,
                      SLOT free[]) {
    int sch[NUM_DAY], i, j;

    for (i = 0; i < NUM_DAY; i++) {
        sch[i] = [ a ];
        for (j = 1; j < num; j++) {
            if (time_tbl[j][i] != 0) {
                sch[i] = 1;
            }
        }
    }
    search_free_slot(sch, slot_num, free);
}

void search_free_slot(int sch[], int slot_num, SLOT free[]) {
    int cnt = 0, free_num = 0, i;

    for (i = 0; i < NUM_DAY; i++) {
        if (sch[i] != 0){
            if (cnt >= slot_num) {
                free[free_num].s_start = [ b ];
                free[free_num].s_num   = cnt;
                free_num++;
            }
            [ c ];
```

```
            } else {
                cnt++;
            }
        }
        if (    d    ) {
            free[free_num].s_start = NUM_DAY - cnt;
            free[free_num].s_num   = cnt;
            free_num++;
        }
        free[free_num].s_start = -1;
    }
```

**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　0　　イ　1　　ウ　time_tbl[0][0]

エ　time_tbl[0][i]

bに関する解答群

ア　i　　イ　i - cnt　　ウ　i - cnt + 1

エ　i - cnt - 1　　オ　NUM_DAY - cnt

cに関する解答群

ア　cnt = 0　　イ　cnt++　　ウ　cnt--

エ　free_num = 0　　オ　free_num++　　カ　free_num--

dに関する解答群

ア　cnt == 0　　イ　cnt > 0　　ウ　cnt >= slot_num

エ　cnt < slot_num　　オ　free_num == 0　　カ　free_num > 0

**設問2**　出席社員が増え，出席社員全員での会議の計画ができない（可能時間帯がない）場合がでてきた。そこで，出席社員を出席が必須の社員（以下，必須出席社員という）とそれ以外の社員（以下，任意出席社員という）に分け，必須出席社員全員が出席可能な時間帯で会議の計画をすることにした。さらに，任意出席社員について順位付けを行い，順位の高い方から最多の社員が出席可能な時間帯に絞込みを行うことにした。この条件で会議が計画できる時間帯を調べるプログラムを作成した。作成したプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。ここで，［　a　］には，設問1の正しい答えが入っているものとする。

〔プログラム2の説明〕

(1)　関数 maxmem_free_slot 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。

num　　出席社員数
must_num　必須出席社員数
time_tbl　出席社員全員のスケジュール(スケジュール表へのポインタの配列)
slot_num　会議時間（30分単位の時間帯数）
free　　会議開催の候補時間帯（SLOT型の配列）

①　必須出席社員のスケジュール表へのポインタは，time_tbl の先頭から人数分が格納されている。

②　任意出席社員のスケジュール表へのポインタは，time_tbl の必須出席社員のスケジュールに続けて，順位の高い方から順に格納されている。

(2)　候補時間帯 free には，必須出席社員全員が出席可能な時間帯の中で，任意出席社員の順位の高い方から最多の社員が出席可能な時間帯を全て格納する。最後に格納した要素の次の要素の開始時間帯番号には，−1を格納する。

〔プログラム2〕

```
#define NUM_DAY 19      /* 1日の時間帯数 */

typedef struct {
    int s_start;        /* 開始時間帯番号 */
    int s_num;          /* 時間帯数 */
} SLOT;

void maxmem_free_slot(int, int, int *[], int, SLOT []);
void search_free_slot(int [], int, SLOT []);

void maxmem_free_slot(int num, int must_num, int *time_tbl[],
                      int slot_num, SLOT free[]) {
    int sch[NUM_DAY], i, j, k;
    SLOT wk_free[2][NUM_DAY + 1];

    /* 必須出席社員全員が出席可能な時間帯を探し出す */
    for (i = 0; i < NUM_DAY; i++) {
        sch[i] = [ a ];
        for ([ e ]; j++) {
            if (time_tbl[j][i] != 0) {
                sch[i] = 1;
            }
        }
    }
    search_free_slot(sch, slot_num, wk_free[0]);

    /* 任意出席社員の順位の高い方から
       最多の社員が出席可能な時間帯に絞り込む */
    for (j = must_num; j < num; j++) {
        for (i = 0; i < NUM_DAY; i++) {
            if (time_tbl[j][i] != 0) {
                sch[i] = 1;
            }
        }
        search_free_slot(sch, slot_num, wk_free[1]);
        if ([ f ]) {
            break;
        }
        k = 0;
        do {
            wk_free[0][k] = wk_free[1][k];
        } while (wk_free[1][k++].s_start != -1);
    }

    /* 候補時間帯を格納する */
    k = 0;
    do {
        free[k] = wk_free[0][k];
```

```
    } while (wk_free[0][k++].s_start != -1);
}
```

eに関する解答群

ア　j = 1; j < must_num　　イ　j = 1; j < num

ウ　j = 1; j < num - must_num　　エ　j = must_num; j < num

オ　j = num - must_num; j < num

fに関する解答群

ア　wk_free[1][0].s_num == 0

イ　wk_free[1][0].s_num == -1

ウ　wk_free[1][0].s_num > slot_num

エ　wk_free[1][0].s_start == 0

オ　wk_free[1][0].s_start == -1

カ　wk_free[1][0].s_start > slot_num

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

〔プログラムの説明〕

ある遊園地では，IC カードを利用した精算システムを導入した。入園時に，入園者から入園料を受け取ると，入園者番号を記録した IC カードを渡し，入園者番号と年齢層の情報を入園者ファイルに格納する。各施設では，提示された IC カードから入園者番号を読み取り，施設利用実績をイベントファイルに格納する。退園時に，入園者から IC カードが返却されたときに，施設利用料金を一括で徴収する。

このプログラムは，入園者ファイル及びイベントファイルに格納されたデータを読み込み，入園者に関する情報を集計して表示する。



(1)　入園者ファイルは，図 1 に示すレコード様式の，入園者番号を主キーとする索引ファイルである。ファイルは，営業日ごとに作成する。

| 入園者番号<br>4 桁 | 年齢層コード<br>1 桁 |
|---|---|

**図 1　入園者ファイルのレコード様式**

①　入園者番号は，営業日ごとに一意となる，1 から始まる 4 桁の数字である。

②　年齢層コードには，4～12 歳は 1，13～18 歳は 2，19～29 歳は 3，30～49 歳は 4，50歳以上は 5を格納する。4 歳未満は入園料及び遊戯施設利用料金が無料となるので，IC カードは発行せず，入園者ファイルにも格納しない。

(2)　イベントファイルは，図 2 に示すレコード様式の順ファイルである。入園者の IC カード利用状況（以下，イベントという）を格納する。ファイルは，営業日ごとに作成する。

| 入園者番号<br>4 桁 | イベントコード<br>2 桁 | 時刻<br>6 桁 | 金額<br>6 桁 |
|---|---|---|---|

**図 2　イベントファイルのレコード様式**

① イベントコードには，発生したイベントに応じて，表1に示すイベントコードを格納する。

表1　イベントとイベントコード

| イベント | イベントコード | 補足 |
|---|---|---|
| 入園 | 01 | IC カード発行 |
| 退園 | 02 | IC カード返却 |
| 観覧車 | 03 | 遊戯施設利用 |
| ゴーカート | 04 | 遊戯施設利用 |
| ジェットコースター | 05 | 遊戯施設利用 |
| メリーゴーランド | 06 | 遊戯施設利用 |
| ウォータースライダー | 07 | 遊戯施設利用 |
| お化け屋敷 | 08 | 遊戯施設利用 |
| レストラン | 09 | 飲食 |
| ショップ | 10 | 購入 |

② 時刻には，イベントが発生した時刻の時，分，秒を24時間表記で格納する。時系列で格納するので，レコードは時刻の昇順になっている。

なお，この遊園地は10時に開園し，全ての入園者は 20 時になるまでに退園する。20時台の時刻が格納されることはない。

③ 金額には，該当するイベントを利用した際の金額を格納する。

なお，入退園（イベントコード01，02）の場合は，0を格納する。

④ 一旦退園した入園者が，その日のうちに同じ入園者番号で再入園することはない。

(3) 集計結果の表示例を図3に示す。

```
TOTAL GUESTS  2012
STAY GUESTS
         ----+----1----+----2----+----3----+----4----+----5
10:00-59 @@@*++*==-
11:00-59 @@@@*****++==-
     ⋮
19:00-59 *++++=
```

図3　集計結果の表示例

① TOTAL GUESTSには，入園者数を表示する。1日の入園者数が5,000人を超えることはない。

② STAY GUESTS には，左端に表示した時間帯の滞在者数を，年齢層別に棒グラフで表示する。例えば，10時ちょうどに入園し，11時ちょうどに退園した者は，10時台と11時台の滞在者として数える。

③ 棒グラフを構成する各記号は，表2に示すとおり滞在者の年齢層と対応している。記号一つは100人を表し，100人未満の端数は切り捨てる。

**表2　年齢層と対応する記号**

| 年齢層 | 記号 |
|---|---|
| 4～12歳 | @ |
| 13～18歳 | * |
| 19～29歳 | + |
| 30～49歳 | = |
| 50歳以上 | - |

〔プログラム〕

```
(行番号)
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD GUEST.
 4  01 G-REC.
 5     02 G-NO            PIC X(4).
 6     02 G-AGE           PIC 9(1).
 7  FD EVENT.
 8  01 E-REC.
 9     02 E-NO            PIC 9(4).
10     02 E-CODE          PIC 9(2).
11        88 E-ENTER         VALUE 1.
12        88 E-EXIT          VALUE 2.
13        88 E-FAC           VALUE 3 THRU 8.
14        88 E-MEAL          VALUE 9.
15        88 E-SHOP          VALUE 10.
16     02 E-TIME.
17        03 E-HH             PIC 9(2).
18        03 E-MMSS           PIC 9(4).
19     02 E-SALES         PIC 9(6).
20  WORKING-STORAGE SECTION.
21  77 READ-FLAG        PIC X(1) VALUE SPACE.
22     88  EOF             VALUE "E".
```

```
77 CR-HH           PIC 9(2) VALUE 10.
77 I               PIC 9(2).
77 J               PIC 9(2).
77 MARK-CNT        PIC 9(2).
77 MARK-POS        PIC 9(2).
77 TOTAL-GUEST     PIC 9(4) VALUE 0.
77 TOTAL-GUEST-X  REDEFINES TOTAL-GUEST PIC X(4).
01 STAY-GUEST.
   02 STAY-CNT        PIC 9(4) OCCURS 5 VALUE ZERO.
01 PRT-GUEST.
   02 PRT-TIME        OCCURS 10.
      03 PRT-AGE         PIC 9(4) OCCURS 5 VALUE ZERO.
01 HD-SCALE     PIC X(50) VALUE
     "----+----1----+----2----+----3----+----4----+----5".
01 LHEADER.
   02 LHD-HH       PIC 9(2).
   02              PIC X(7)  VALUE ":00-59".
   02 LHD-MARK     PIC X(50).
01 AGE-MARK.
   02 KIDS-M       PIC X(50) VALUE ALL "@".
   02 STUDENT-M    PIC X(50) VALUE ALL "*".
   02 YOUNG-M      PIC X(50) VALUE ALL "+".
   02 ADULT-M      PIC X(50) VALUE ALL "=".
   02 OLD-M        PIC X(50) VALUE ALL "-".
01 REDEFINES AGE-MARK.
   02 MARK         PIC X(50) OCCURS 5.
PROCEDURE DIVISION.
MAIN-PROC.
    OPEN INPUT GUEST EVENT.
    PERFORM UNTIL EOF
       READ EVENT
          AT END
             SET EOF TO TRUE
          NOT AT END
             IF CR-HH NOT = E-HH THEN
                PERFORM VARYING CR-HH FROM CR-HH BY 1
                        UNTIL CR-HH = E-HH
                   a
                END-PERFORM
             END-IF
             PERFORM SUM-PROC
       END-READ
    END-PERFORM.
    CLOSE GUEST EVENT.
    PERFORM PRT-PROC.
    STOP RUN.
SUM-PROC.
    MOVE E-NO TO G-NO.
    READ GUEST.
    EVALUATE TRUE
       WHEN E-ENTER
```

COBOL

```
74            ADD 1 TO [ b ] [ c ] [ d ]
75          WHEN E-EXIT
76             SUBTRACT 1 FROM STAY-CNT(G-AGE)
77       END-EVALUATE.
78   PRT-PROC.
79       DISPLAY "TOTAL GUESTS  " TOTAL-GUEST-X.
80       DISPLAY "STAY GUESTS".
81       DISPLAY "         " HD-SCALE.
82       PERFORM VARYING I FROM 1 BY 1 UNTIL I > 10
83          MOVE 1 TO MARK-POS
84          COMPUTE LHD-HH = I + 9
85          MOVE SPACE TO LHD-MARK
86          PERFORM VARYING J FROM 1 BY 1 UNTIL J > 5
87             COMPUTE MARK-CNT = PRT-AGE(I, J) / 100
88             IF MARK-CNT > 0 THEN
89                MOVE MARK(J)(1:MARK-CNT) TO LHD-MARK(MARK-POS:)
90                                [        e        ]
91             END-IF
92          END-PERFORM
93          DISPLAY LHEADER
94       END-PERFORM.
```

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に関する解答群

ア　MOVE STAY-CNT(CR-HH) TO PRT-AGE(CR-HH - 8, E-HH)

イ　MOVE STAY-CNT(CR-HH - 8) TO PRT-TIME(CR-HH)

ウ　MOVE STAY-GUEST TO PRT-TIME(CR-HH - 8)

エ　MOVE ZERO TO PRT-AGE(CR-HH - 8, CR-HH)

b～dに関する解答群

ア　CR-HH

イ　I

ウ　J

エ　PRT-AGE(CR-HH - 9, G-AGE)

オ　PRT-AGE(G-AGE, CR-HH - 9)

カ　STAY-CNT(CR-HH)

キ　STAY-CNT(G-AGE)

ク　TOTAL-GUEST

eに関する解答群

ア　ADD J TO MARK-CNT

イ　ADD J TO MARK-POS

ウ　ADD MARK-CNT TO MARK-POS

エ　ADD MARK-POS TO MARK-CNT

**設問2**　各遊戯施設の延べ利用者数を集計し，図4に示すグラフを追加表示するようにプログラムを変更した。表3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)　左端に表示したイベントコードに対応する遊戯施設を利用した延べ人数を表示する。記号#一つは100人を表し，100人未満の端数は切り捨てる。一つの遊戯施設の延べ利用者数が5,000人を超えることはない。

(2)　延べ利用者数が100人未満の遊戯施設の行は表示しない。

```
RECREATIONAL FACILITIES
    ----+----1----+----2----+----3----+----4----+----5
03: ################
04: #######
 :
08: ##########
```

**図4　遊戯施設利用状況の表示例**

表3　プログラムの変更

| 処置 | 変更内容 |
|---|---|
| 行番号48と49の間に追加 | 01 .<br>　02 FAC-CNT　　PIC 9(4) OCCURS 6 VALUE ZERO.<br>01 FAC-HEADER.<br>　02 FAC-NO　　PIC 9(2).<br>　02　　PIC X(2) VALUE ": ".<br>01 FAC-MARK　　PIC X(50) VALUE ALL "#". |
| 行番号76と77の間に追加 | 　　WHEN E-FAC<br>　　[ f ] |
| 行番号94の後ろに追加 | 　DISPLAY "RECREATIONAL FACILITIES".<br>　DISPLAY "　　" HD-SCALE.<br>　PERFORM [ g ]<br>　　COMPUTE MARK-CNT = FAC-CNT(FAC-NO - 2) / 100<br>　　IF MARK-CNT NOT = 0 THEN<br>　　　DISPLAY FAC-HEADER FAC-MARK(1:MARK-CNT)<br>　　END-IF<br>　END-PERFORM. |

COBOL

f に関する解答群

ア　ADD 1 TO FAC-CNT(E-CODE)

イ　ADD 1 TO FAC-CNT(E-CODE - 2)

ウ　ADD G-AGE TO FAC-CNT(E-CODE)

エ　ADD G-NO TO FAC-CNT(E-CODE)

オ　ADD STAY-CNT(G-AGE) TO FAC-CNT(E-CODE)

g に関する解答群

ア　6 TIMES

イ　UNTIL FAC-NO > 6 AND MARK-CNT > 50

ウ　UNTIL FAC-NO > 6 OR MARK-CNT > 50

エ　VARYING FAC-NO FROM 1 BY 1 UNTIL FAC-NO > 6

オ　VARYING FAC-NO FROM 3 BY 1 UNTIL FAC-NO > 8

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〕

試験の成績を管理するプログラムである。

(1) クラス ScoreManager は，後述するクラス ValueSortedMap を継承したクラスで，試験の成績を管理する。点数の高い順に学籍番号（英数字で構成される固定長の文字列）と点数を出力するメソッド print をもつ。

(2) クラス ValueSortedMap は，キーと値の対応付けを値の昇順，又はコンストラクタで指定したコンパレータに従った順に保持する。キーと値の対応付けをキーの昇順，又はコンパレータに従った順に保持するクラス TreeMap を利用している。主なメソッドは次のとおりである。

```
public V put(K key, V value)
```

key に value を対応付けて登録する。key 又は value が null のときは例外 NullPointerException を投げる。key が既に他の値と対応付けられていれば，その値を value で置き換え，置き換えられる前の値を返す。key に値が対応付けられていなければ，null を返す。

```
public V get(K key)
```

key に対応付けられた値を返す。key と値の対応付けがなければ，null を返す。

```
public V remove(K key)
```

key と値の対応付けを削除し，対応付けられていた値を返す。key と値の対応付けがなければ，null を返す。

```
public Iterator<K> iterator()
```

対応付けられた値の昇順，又はコンストラクタで指定したコンパレータに従った順に，キーを返すための反復子を返す。

フィールド map は，キーと値の対応付けを保持する。

フィールド reverseMap は，値にキーを対応付けて保持する。異なるキーに同じ値が対応付けられることがあるので，値に対応付けられるのはキーのリストである。

(3) クラス ScoreManagerTester は，成績管理プログラムのテストプログラムである。メソッド main の実行結果を，図 1 に示す。

```
数学の成績
 FE0002 : 90 点
 FE0005 : 90 点
 FE0003 : 85 点
 FE0001 : 85 点
null
```

図 1　メソッド main の実行結果

〔プログラム 1〕

```
import java.util.Comparator;

public class ScoreManager extends ValueSortedMap<  a  > {
   private final String subject;

   public ScoreManager(String subject) {
      super(new Comparator<Integer>() {
            public int compare(Integer a, Integer b) {
               return b.compareTo(a);
            }
         });
      this.subject = subject;
   }

   public void print() {
      System.out.println(subject + "の成績");
      for (String name : this) {
         System.out.printf(" %s : %d点%n", name, get(name));
      }
   }
}
```

〔プログラム 2〕

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.TreeMap;
```

```
public class ValueSortedMap<K, V> implements Iterable<K> {
   Map<K, V> map = new HashMap<K, V>();
   Map<V, List<K>> reverseMap;

   public ValueSortedMap() {
      reverseMap = new TreeMap<V, List<K>>();
   }

   public ValueSortedMap(Comparator<? super V> c) {
      reverseMap = new TreeMap<V, List<K>>(c);
   }

   public V put(K key, V value) {
      if ( [ b ] ) {
         throw new NullPointerException();
      }
      V old = remove(key);
      map.put(key, value);
      List<K> keys = reverseMap.get(value);
      if ( [ c ] ) {
         keys = new ArrayList<K>();
         reverseMap.put(value, keys);
      }
      keys.add(key);
      return old;
   }

   public V get(K key) {
      return map.get(key);
   }

   public V remove(K key) {
      V value = [ d ] (key);
      if (value != null) {
         List<K> keys = reverseMap.get(value);
         keys.remove(key);
         if (keys.isEmpty()) {
            [ e ] (value);
         }
      }
      return value;
   }

   public int size() {
      return map.size();
   }

   public Iterator<K> iterator() {
      return new Iterator<K>() {
         Iterator<V> vi = reverseMap.keySet().iterator();
```

```
        Iterator<K> ki = new ArrayList<K>().iterator();

        public boolean hasNext() {
            return vi.hasNext() || ki.hasNext();
        }

        public K next() {
            if (hasNext()) {
                if (!ki.hasNext()) {
                    ki = reverseMap.get(vi.next()).iterator();
                }
                return ki.next();
            }
            throw new NoSuchElementException();
        }

        public void remove() {
            throw new UnsupportedOperationException();
        }
    };
  }
}
```

〔プログラム3〕

```
public class ScoreManagerTester {
   public static void main(String[] args) {
      ScoreManager sm = new ScoreManager("数学");
      try {
         sm.put("FE0001", 70);
         sm.put("FE0002", 90);
         sm.put("FE0003", 85);
         sm.put("FE0004", 95);
         sm.put("FE0005", 90);
         sm.put("FE0001", 85);
         sm.remove("FE0004");
         sm.put(null, 90);
         sm.put("FE0004", 90);
      } catch (NullPointerException e) {}
      sm.print();
      System.out.println(sm.get("FE0004"));
   }
}
```

**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　Integer　　イ　Integer, String

ウ　String　　エ　String, Integer

b に関する解答群

ア　key != null && value != null　　イ　key != null || value != null

ウ　key == null && value == null　　エ　key == null || value == null

c に関する解答群

ア　!keys.isEmpty()　　イ　keys != null

ウ　keys == null　　エ　keys.isEmpty()

d, e に関する解答群

ア　Map.get　　イ　map.get

ウ　Map.remove　　エ　map.remove

オ　remove　　カ　reverseMap.get

キ　reverseMap.remove

**設問2**　点数が同じ場合には，学籍番号の文字列としての自然順序付けに従って出力するように変更する。クラス ValueSortedMap で使用しているクラスやインタフェースの変更だけで実現する場合，変更内容として適切なものを，解答群の中から選べ。

解答群

ア　ArrayList を TreeSet に変更する。

イ　List と ArrayList を Set に変更する。

ウ　List を Set に変更し，ArrayList を TreeSet に変更する。

エ　List を TreeSet に変更し，ArrayList を Set に変更する。

オ　List を TreeSet に変更する。

**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〕

主プログラムから渡された二つの数字列（数字列 1 と数字列 2）をそれぞれ正の 10 進数とみなして加算し，結果を数字列で返す副プログラム ADDC である。二つの数字列の加算の例を図 1 に示す。

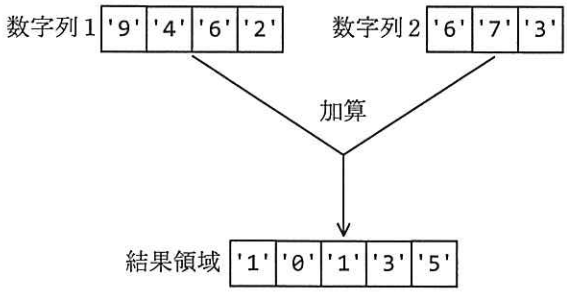



**図 1　二つの数字列の加算の例**

(1)　汎用レジスタ GR1 ～ GR5 には，それぞれ次の内容が設定されて，主プログラムから渡される。

GR1：数字列 1 の先頭アドレス

GR2：数字列 2 の先頭アドレス

GR3：結果領域の先頭アドレス

GR4：数字列 1 の長さ（文字数）

GR5：数字列 2 の長さ（文字数）

(2)　数字は 1 語に 1 文字格納する。

(3)　結果の数字列は結果領域に格納し，長さ（文字数）は GR0 に設定して，主プログラムに返す。

(4)　数字列 1 と数字列 2 の長さ（文字数）は，それぞれ 1 以上とする。

(5)　数字列 1 と数字列 2 の左端の文字は“0”でないものとする。

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

〔プログラム1〕

(行番号)

```
 1  ADDC     START
 2           RPUSH
 3           CPA     GR4,GR5
 4           JPL     CONT
 5           PUSH    0,GR4          ;
 6           LD      GR4,GR5        ;
 7           POP     GR5            ;  GR4≧GR5 となるように
 8           PUSH    0,GR1          ;  数字列1と数字列2のポインタ
 9           LD      GR1,GR2        ;  を入替え
10           POP     GR2            ;
11  CONT     ADDA    GR1,GR4
12           ADDA    GR2,GR5
13           LD      GR0,=0         ; 桁上げフラグを初期化
14           PUSH    0              ; スタックデータの終わりの印
15  LOOP1    LAD     GR2,-1,GR2
16           LD      GR6,0,GR2
17           AND     GR6,=#000F
18  LOOP2    LAD     GR1,-1,GR1
19           ADDA    GR6,0,GR1      ; 1桁加算
20           ADDA    GR6,GR0        ; 桁上げフラグを加算
21           CPA     GR6,='9'       ; 桁上げ?
22           JPL     CARRY
23           LD      GR0,=0         ; 桁上げフラグをクリア
24           JUMP    NEXT
25  CARRY    LD      GR0,=1         ; 桁上げフラグを設定
26                   a
27  NEXT     PUSH    0,GR6          ; 加算結果1桁を保存
28                   b
29           JZE     END1
30                   c
31           JPL     LOOP1
32           LD      GR6,=0
33           JUMP    LOOP2
34  END1     LD      GR0,GR0        ; 左端を桁上げ?
35           JZE     NOCARRY
36           LD      GR1,='1'       ; 左端に'1'を追加
37  LOOP3    ST      GR1,0,GR3
38           LAD     GR3,1,GR3
39  NOCARRY  POP     GR1            ; 加算結果1桁取出し
40           LD      GR1,GR1
41           JZE     END2
42           ADDA    GR0,=1
43           JUMP    LOOP3
44  END2     RPOP
45           RET
46           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。

解答群

| | | |
|---|---|---|
| ア　SUBA　GR4,=1 | イ　SUBA　GR4,=9 | ウ　SUBA　GR4,=10 |
| エ　SUBA　GR5,=1 | オ　SUBA　GR5,=9 | カ　SUBA　GR5,=10 |
| キ　SUBA　GR6,=1 | ク　SUBA　GR6,=9 | ケ　SUBA　GR6,=10 |

**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。

図1の数字列1と数字列2を引数として副プログラム ADDC を実行した場合，行番号25の LD 命令は［　d　］回実行される。

解答群

ア　0　　イ　1　　ウ　2　　エ　3　　オ　4

**設問3**　副プログラム ADDC を使用して，n個（n≧1）の数字列を入力し，総和を求めて数字列で出力するプログラム SUMC を作成した。n=5の場合の例を図2に示す。

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。

なお，入力される数字列，総和は256桁以内に収まるものとする。

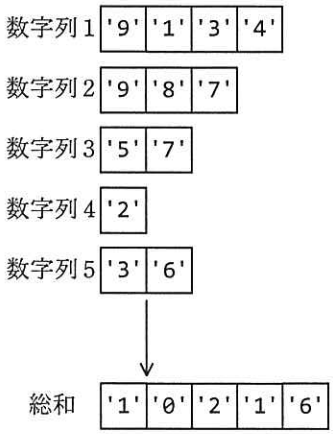



**図2　数字列1～5の総和の例**

アセンブラ

〔プログラム2〕

```
SUMC    START
        LAD     GR1,BUF1        ; 数字列1のポインタを設定
        IN      BUF1,LEN1       ; 最初の数字列を入力し数字列1とする
        LD      GR4,LEN1        ; 数字列1の長さを設定
        LAD     GR3,BUF3        ; BUF3を総和の領域として設定
        LAD     GR2,BUF2        ; 数字列2のポインタを設定
LOOP    IN      BUF2,LEN2       ; 次の数字列を入力し数字列2とする
        LD      GR5,LEN2        ; 数字列2の長さを設定
        JMI     FIN1            ; 入力終了
        CALL    ADDC
        [    e    ]
        PUSH    0,GR1           ;┐ 総和の領域のポインタと
        LD      GR1,GR3         ;┘ 数字列1のポインタを入替え
        POP     GR3             ;
        JUMP    LOOP
FIN1    [    f    ]
        CPA     GR1,GR3         ; 総和の領域を判定
        JZE     FIN2
        ST      GR4,LEN1
        OUT     BUF1,LEN1
        JUMP    FIN3
FIN2    ST      GR4,LEN3
        OUT     BUF3,LEN3
FIN3    RET
BUF1    DS      256             ; 数字列1又は総和
BUF2    DS      256             ; 数字列2
BUF3    DS      256             ; 総和又は数字列1
LEN1    DS      1
LEN2    DS      1
LEN3    DS      1
        END
```

アセンブラ

解答群

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ア | LAD | GR3,BUF1 | イ | LAD | GR3,BUF2 | ウ | LAD | GR3,BUF3 |
| エ | LD | GR3,GR0 | オ | LD | GR4,GR0 | カ | LD | GR5,GR0 |

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

〔表計算の説明〕

表計算ソフトを用いて，ある大学学部所有の図書を管理するプロトタイプシステムを作成した。プロトタイプシステムは“図書情報”と“利用者情報”の二つのワークシートから成る。

〔ワークシート：図書情報〕

学部で所有している図書のうち 200 冊を抽出し，図 1 のワークシート“図書情報”を作成した。

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 図書 ID | 図書名 | 利用者 ID | 貸出日 | 返却期日 | 延滞日数 | | 日付 | 11-06-15 |
| 2 | 1 | プログラミング入門 | | | | | | | |
| 3 | 2 | ネットワーク | 4 | 11-05-08 | 11-06-05 | 10 | | | |
| 4 | 3 | 情報科学入門 | 4 | 11-05-08 | 11-06-05 | 10 | | | |
| 5 | 4 | データベース | | | | | | | |
| 6 | 5 | 統計入門 | 37 | 11-06-13 | 11-06-20 | 0 | | | |
| 7 | 6 | 情報セキュリティ | 2 | 11-05-28 | 11-06-25 | 0 | | | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | | |
| 201 | 200 | 論理学応用 | 2 | 11-05-28 | 11-06-25 | 0 | | | |
| 202 | | | | | | | | | |
| 203 | 図書 ID | 170 | 利用者 ID | 40 | | | | | |

注記　行 203 については設問 2 で説明する。

**図 1　ワークシート“図書情報”**

(1)　セル B2～B201 には，該当する図書名を入力する。セル A2～A201 には，1 から始まる連番で図書 ID を入力する。

(2)　セル C2～C201 には，図書が貸出し中であれば，借りた利用者の利用者 ID を入力する。そうでなければ空値である。

(3)　セル D2～D201 には，図書が貸出し中であれば，貸し出した日付を入力する。そうでなければ空値である。

(4)　セル E2～E201 には，返却期日を算出して格納する。貸出し中でなければ空値

である。

(5) セル F2～F201 には，返却期日を過ぎても返却されていない図書に関して，延滞日数を求める式を入力する。この式は，返却期日を過ぎていなければ 0 に，貸出し中でなければ空値になる。

(6) セル I1 には，本日の日付を求める式を入力する。

(7) 日付の表記は yy-mm-dd で表示されるが，表計算ソフトの内部では 1970 年 1 月 1 日からの経過日数を整数値で保持している。計算にはこの値を利用する。

〔ワークシート：利用者情報〕

利用対象者を 50 名抽出し，利用者ごとの情報から図 2 に示すワークシート“利用者情報”を作成した。

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 利用者 ID | 氏名 | 属性 | 残り貸出冊数 | 延滞状態 | | 属性 | 貸出上限冊数 | 貸出日数 |
| 2 | 1 | 情報一郎 | 学部生 | 4 | * | | 教員 | 20 | 28 |
| 3 | 2 | 情報二郎 | 教員 | 8 | | | 大学院生 | 10 | 28 |
| 4 | 3 | 情報三郎 | 大学院生 | 10 | | | 学部生 | 5 | 14 |
| 5 | 4 | 情報四郎 | 教員 | 17 | * | | その他 | 3 | 7 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | | | |
| 51 | 50 | 情報花子 | その他 | 3 | | | | | |

**図 2　ワークシート“利用者情報”**

(1) セル B2～B51 には，氏名を入力する。セル A2～A51 には，1 から始まる連番で利用者 ID を入力する。

(2) セル C2～C51 には，属性（教員，大学院生，学部生，その他）を入力する。

(3) セル D2～D51 には，貸出上限までの現在の残り冊数（以下，残り貸出冊数という）を求める式を入力する。

(4) セル E2～E51 には，延滞状態の図書が 1 冊でもあれば“*”に，延滞状態の図書が無いときは空値になる式を入力する。

(5) セル G2～G5 には属性の名称を，セル H2～H5 とセル I2～I5 には，それぞれの属性に応じた貸出上限冊数と貸出日数を入力する。

**設問1**　ワークシート“図書情報”及び“利用者情報”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。

返却期日を算出するために，次の式をワークシート“図書情報”のセル E2 に入力し，セル E3～E201 に複写する。

IF(C2 ≠ null, ［ a ］, null)

延滞日数を算出するために，次の式をワークシート“図書情報”のセル F2 に入力し，セル F3～F201 に複写する。

IF(C2 ≠ null, ［ b ］, null)

残り貸出冊数を算出するために，次の式をワークシート“利用者情報”のセル D2 に入力し，セル D3～D51 に複写する。

［ c ］

延滞状態を表示するために，次の式をワークシート“利用者情報”のセル E2 に入力し，セル E3～E51 に複写する。

［ d ］

a に関する解答群

ア　D2＋垂直照合(C2, 利用者情報!G$2～I$5, 3, 0)

イ　D2＋垂直照合(C2, 利用者情報!A$2～C$51, 1, 0)

ウ　D2＋垂直照合(C2, 利用者情報!A$2～C$51, 3, 0)

エ　D2＋垂直照合(垂直照合(C2, 利用者情報!A$2～C$51, 1, 0), 利用者情報!G$2～I$5, 3, 0)

オ　D2＋垂直照合(垂直照合(C2, 利用者情報!A$2～C$51, 3, 0), 利用者情報!G$2～I$5, 3, 0)

カ　D2＋水平照合(C2, 利用者情報!A$2～C$51, 1, 0)

キ　D2＋水平照合(垂直照合(C2, 利用者情報!A$2～C$51, 1, 0), 利用者情報!G$2～I$5, 3, 0)

ク　D2＋水平照合(垂直照合(C2, 利用者情報!A$2～C$51, 3, 0), 利用者情報!G$2～I$5, 3, 0)

bに関する解答群

ア　E2－I$1　　イ　I$1－E2

ウ　IF(E2≧I$1, 0, I$1－E2)　　エ　IF(E2≧I$1, E2, I$1)

オ　IF(E2≧I$1, I$1, E2)　　カ　IF(E2≧I$1, I$1－E2, 0)

cに関する解答群

ア　垂直照合(C2, G$2～I$5, 2, 0)

イ　垂直照合(C2, G$2～I$5, 2, 0)＋条件付個数(図書情報!C$2～C$201, ＝A2)

ウ　垂直照合(C2, G$2～I$5, 2, 0)－条件付個数(図書情報!C$2～C$201, ＝A2)

エ　垂直照合(C2, G$2～I$5, 3, 0)

オ　垂直照合(C2, G$2～I$5, 3, 0)＋条件付個数(図書情報!C$2～C$201, ＝A2)

カ　垂直照合(C2, G$2～I$5, 3, 0)－条件付個数(図書情報!C$2～C$201, ＝A2)

dに関する解答群

ア　IF(D2＞0, '＊', null)

イ　IF(D2＝0, '＊', null)

ウ　IF(合計(図書情報!C$2～C$201)＝C2, '＊', null)

エ　IF(合計(図書情報!C$2～C$201)≠C2, '＊', null)

オ　IF(合計(図書情報!F$2～F$201)＝F2, '＊', null)

カ　IF(合計(図書情報!F$2～F$201)≠F2, '＊', null)

キ　IF(条件付合計(図書情報!C$2～C$201, ＝A2, 図書情報!F$2～F$201)＞0, '＊', null)

ク　IF(条件付合計(図書情報!C$2～C$201, ＝A2, 図書情報!F$2～F$201)＝0, '＊', null)

表計算

**設問2**　利用者が図書を借りる際，その情報を登録するマクロ Borrowing をワークシート“図書情報”に格納した。ワークシート“図書情報”のセルB203に図書IDを，セルD203に利用者IDを入力して，マクロ Borrowing を実行すると，次に示す三つの条件を満たすとき，貸出登録される。

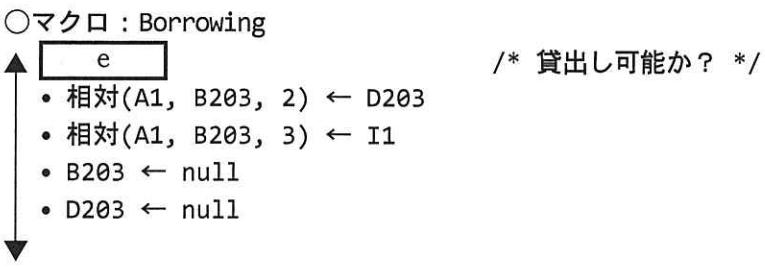
(1) 対象図書が貸出し中でない。

(2) 対象利用者の残り貸出冊数が１以上である。

(3) 対象利用者が現在借りている全ての図書が延滞状態でない。

［　　］に入れる正しい答えを，解答群の中から選べ。

〔マクロ：Borrowing〕




eに関する解答群

ア　論理積(相対(A1, B203, 2) ＝ null, 相対(利用者情報!A1, D203, 3) ＞ 0,
相対(利用者情報!A1, D203, 4) ≠ '*' )

イ　論理積(相対(A1, B203, 2) ＝ null, 相対(利用者情報!A1, D203, 3) ＞ 0,
相対(利用者情報!A1, D203, 4) ＝ '*' )

ウ　論理積(相対(A1, B203, 2) ＝ null, 相対(利用者情報!A1, D203, 3) ＝ 0,
相対(利用者情報!A1, D203, 4) ≠ '*' )

エ　論理積(相対(A1, B203, 2) ＝ null, 相対(利用者情報!A1, D203, 3) ＝ 0,
相対(利用者情報!A1, D203, 4) ＝ '*' )

オ　論理積(相対(A1, B203, 2) ≠ null, 相対(利用者情報!A1, D203, 3) ＞ 0,
相対(利用者情報!A1, D203, 4) ≠ '*' )

カ　論理積(相対(A1, B203, 2) ≠ null, 相対(利用者情報!A1, D203, 3) ＞ 0,
相対(利用者情報!A1, D203, 4) ＝ '*' )

キ　論理積(相対(A1, B203, 2) ≠ null, 相対(利用者情報!A1, D203, 3) ＝ 0,
相対(利用者情報!A1, D203, 4) ≠ '*' )

ク　論理積(相対(A1, B203, 2) ≠ null, 相対(利用者情報!A1, D203, 3) ＝ 0,
相対(利用者情報!A1, D203, 4) ＝ '*' )

**設問3**　全利用者の過去の貸出履歴に基づき図書を推薦する機能について検討するために，ワークシート“貸出履歴”及びマクロ RecommendBooks を作成した。

〔ワークシート：貸出履歴〕

| | A | B | C | D | E | … | AW | AX | AY | AZ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 図書 ID＼利用者 ID | 1 | 2 | 3 | 4 | … | 48 | 49 | 50 | 推薦度 |
| 2 | 1 | 0 | 0 | 0 | 0 | … | 1 | 0 | 0 | 0.63 |
| 3 | 2 | 0 | 0 | 0 | 1 | … | 0 | 1 | 0 | 0 |
| 4 | 3 | 1 | 1 | 0 | 1 | … | 0 | 0 | 0 | 0 |
| 5 | 4 | 0 | 0 | 0 | 0 | … | 0 | 0 | 1 | 0.56 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 201 | 200 | 1 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 |
| 202 | 類似度 | 0.05 | 0 | 0.06 | 0.04 | … | 0.07 | 0.055 | 0.065 | |
| 203 | | | | | | | | | | |
| 204 | 利用者 ID | 2 | | | | | | | | |

**図3　ワークシート“貸出履歴”**

(1)　図書 ID をセル A2～A201 に，利用者 ID をセル B1～AY1 に入力する。セル B2～AY201 には，利用者が図書を借りたことがあるならば 1 を，無ければ 0 を入力する。

(2)　セル B204 に利用者 ID を入力してマクロ RecommendBooks を実行すると，セル AZ2～AZ201 にその利用者に対する個々の図書の推薦度の値を表示する。

(3)　利用者 ID に $i$ を指定したとき，各図書の推薦度は次の方法で算出する。

①　指定した利用者と残りの全ての利用者間の類似度を数値で表現し，セル B202～AY202 に求める。利用者 ID が $i$，$j$ の利用者間の類似度は，次式で定義する $s_{ij}$ で表現する。

$$s_{ij} = \begin{cases} \dfrac{\sum_{k=1}^{200} (x_{ki} \times x_{kj})}{200} & (i \neq j) \\ 0 & (i = j) \end{cases}$$

ここで，$x_{ki}$ は，利用者 ID $i$ の利用者が，図書 ID $k$ の図書を借りたことがあるならば 1，そうでなければ 0 である。

②　利用者 ID $i$ の図書 ID $k$ に対する推薦度 $r_{ki}$ は類似度 $s_{ij}$ を用いて表現し，セ

表計算

ル AZ2～AZ201 に表示する。

$$r_{ki} = \begin{cases} 0 & (x_{ki}=1 \text{ 又は図書 ID } k \text{ の図書が貸出し中の場合}) \\ \sum_{j=1}^{50} (s_{ij} \times x_{kj}) & (\text{それ以外}) \end{cases}$$

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

〔マクロ：RecommendBooks〕

○マクロ：RecommendBooks
○数値型：J, K, NumBook, NumUser
• NumBook ← 200
• NumUser ← 50
■ J：1, J ≦ NumUser, 1
　• 相対(A1, 201, J) ← 0
　▲ B204 ≠ J
　　■ K：1, K ≦ NumBook, 1
　　　• 相対(A1, 201, J) ← 相対(A1, 201, J) ＋ ［ f ］
　　■
　• 相対(A1, 201, J) ← 相対(A1, 201, J) ／ NumBook
■
■ K：1, K ≦ NumBook, 1
　• 相対(A1, K, 51) ← 0
　▲ ［ g ］
　　■ J：1, J ≦ NumUser, 1
　　　• 相対(A1, K, 51) ← 相対(A1, K, 51) ＋ ［ h ］
　　■
■

表計算

f, hに関する解答群

ア　相対(A1, B204, K) * 相対(A1, J, K)

イ　相対(A1, B204, K) * 相対(A1, K, J)

ウ　相対(A1, J, K) * 相対(A1, J, K)

エ　相対(A1, J, K) * 相対(A1, K, 51)

オ　相対(A1, J, K) * 相対(A1, K, J)

カ　相対(A1, K, B204) * 相対(A1, K, J)

キ　相対(A1, K, J) * 相対(A1, 201, J)

ク　相対(A1, K, J) * 相対(A1, 201, K)

gに関する解答群

ア　論理積(相対(A1, K, B204) = 0, 相対(図書情報!A1, K, 2) = null)

イ　論理積(相対(A1, K, B204) = 0, 相対(図書情報!A1, K, 2) ≠ null)

ウ　論理積(相対(A1, K, B204) = 1, 相対(図書情報!A1, K, 2) = null)

エ　論理積(相対(A1, K, B204) = 1, 相対(図書情報!A1, K, 2) ≠ null)

オ　論理和(相対(A1, K, B204) = 0, 相対(図書情報!A1, K, 2) = null)

カ　論理和(相対(A1, K, B204) = 0, 相対(図書情報!A1, K, 2) ≠ null)

キ　論理和(相対(A1, K, B204) = 1, 相対(図書情報!A1, K, 2) = null)

ク　論理和(相対(A1, K, B204) = 1, 相対(図書情報!A1, K, 2) ≠ null)

## ■ 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 Set<K> keySet()**

登録されているキーの集合を得る。

戻り値：登録されているキーの集合

---

**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，その値を指定された値に置き換える。

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の値
このキーと値の対応付けがなければnull

---

**public V remove(Object key)**

指定されたキーの対応付けが登録されていれば，削除する。

引数： key — キー

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

---

**public int size()**

マップ内のキーと値の対応付けの個数を調べる。

戻り値：キーと値の対応付けの個数

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。

メソッド

public V get(Object key)
インタフェース Map のメソッド get と同じ

public V put(K key, V value)
インタフェース Map のメソッド put と同じ

public V remove(Object key)
インタフェース Map のメソッド remove と同じ

public int size()
インタフェース Map のメソッド size と同じ

java.util

**public class TreeMap<K, V>**

インタフェース Map の実装である。マップはキーの昇順又はコンストラクタで指定されたコンパレータに従った順に整列される。

コンストラクタ

public TreeMap()
空の TreeMap を作る。マップはキーの自然順序付けに従って整列される。

public TreeMap(Comparator<? super K> c)
空の TreeMap を作る。マップは指定されたコンパレータに従って整列される。

メソッド

public V get(Object key)
インタフェース Map のメソッド get と同じ

public Set<K> keySet()
インタフェース Map のメソッド keySet と同じ

public V put(K key, V value)
インタフェース Map のメソッド put と同じ

public V remove(Object key)
インタフェース Map のメソッド remove と同じ

java.util

**public interface List<E>**

リスト（順序付けられたコレクション）のためのインタフェースを提供する。

メソッド

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

public boolean remove(Object obj)
指定された要素がこのリストにあれば，その最初の要素を削除する。
引数： obj — リストから削除する要素
戻り値：リストから要素が削除されれば true
それ以外は false

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

public Iterator<E> iterator()
リスト内の要素についての反復子を返す。
戻り値：リスト内の要素についての反復子

java.util

**public class ArrayList<E>**

インタフェース List の配列による実装である。

コンストラクタ

public ArrayList()
空のリストを作る。

メソッド

public boolean add(E e)
インタフェース List のメソッド add と同じ

public boolean remove(Object obj)
インタフェース List のメソッド remove と同じ

public boolean isEmpty()
インタフェース List のメソッド isEmpty と同じ

public Iterator<E> iterator()
インタフェース List のメソッド iterator と同じ

java.util

**public interface Set<E>**

型 E の要素を集合（セット）として管理するインタフェースを提供する。

メソッド

public boolean add(E e)
指定された要素が集合に含まれていなければ，集合に追加する。
引数： e — 集合に追加する要素
戻り値：指定された要素が集合に含まれていなければ true
それ以外は false

public boolean remove(Object obj)
指定された要素が集合に含まれていれば，集合から削除する。
引数： obj — 集合から削除する要素
戻り値：指定された要素が集合に含まれていれば true
それ以外は false

public boolean isEmpty()
空集合か否かを判定する。
戻り値：空集合ならば true
それ以外は false

public Iterator<E> iterator()
集合内の要素についての反復子を返す。
戻り値：集合内の要素についての反復子

java.util

**public class TreeSet<E>**

インタフェース Set の実装である。要素の昇順に整列される。

コンストラクタ

public TreeSet()
空の集合を作る。集合は要素の自然順序付けに従って整列される。

メソッド

public boolean add(E e)
インタフェース Set のメソッド add と同じ

public boolean remove(Object obj)
インタフェース Set のメソッド remove と同じ

public boolean isEmpty()
インタフェース Set のメソッド isEmpty と同じ

public Iterator<E> iterator()
集合内の要素を昇順で返す反復子を返す。

java.util

**public interface Comparator<T>**

あるオブジェクトの集合に対して完全な順序を規定する関数を提供するインタフェースである。

メソッド

---

`public int compare(T o1, T o2)`

引数で与えられた型 T の二つのオブジェクトを比較し，大小関係を整数値で返す。

引数： o1 — 1 番目のオブジェクト
o2 — 2 番目のオブジェクト

戻り値：o1 が o2 より小さいときは負の値
o1 と o2 が等しいときは 0
o1 が o2 より大きいときは正の値

java.util

**public interface Iterator<E>**

反復子を提供するインタフェースである。

メソッド

---

`public boolean hasNext()`

繰返し処理でさらに要素がある場合に true を返す。

戻り値：反復子がさらに要素をもつ場合は true
それ以外は false

---

`public E next()`

繰返し処理で次の要素を返す。

戻り値：繰返し処理で次の要素

例外： NoSuchElementException — 繰返し処理でそれ以上要素がない場合

---

`public void remove()`

基になるコレクションから，反復子によって最後に返された要素を削除する。

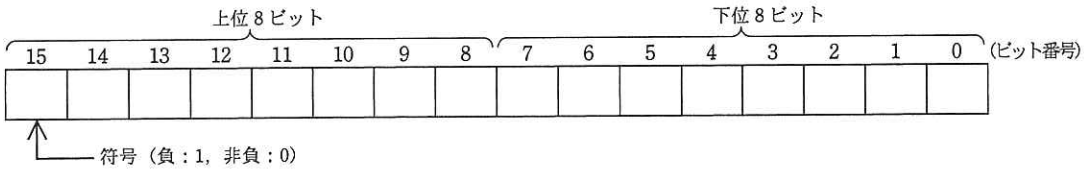
例外： UnsupportedOperationException — Iterator の実装クラスが remove オペレーションをサポートしない場合

# ■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<br>LoaD | LD | r1,r2 | r1 ← (r2) | ○*1 |
| | | r,adr [,x] | r ← (実効アドレス) | |
| ストア<br>STore | ST | r,adr [,x] | 実効アドレス ← (r) | － |
| ロードアドレス<br>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<br>ADD Arithmetic | ADDA | r1,r2 | r1 ← (r1) + (r2) | ○ |
| | | r,adr [,x] | r ← (r) + (実効アドレス) | |
| 論理加算<br>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<br>SUBtract Arithmetic | SUBA | r1,r2 | r1 ← (r1) − (r2) | |
| | | r,adr [,x] | r ← (r) − (実効アドレス) | |
| 論理減算<br>SUBtract Logical | SUBL | r1,r2 | r1 ← (r1) $-_L$ (r2) | |
| | | r,adr [,x] | r ← (r) $-_L$ (実効アドレス) | |
| 論理積<br>AND | AND | r1,r2 | r1 ← (r1) AND (r2) | ○*1 |
| | | r,adr [,x] | r ← (r) AND (実効アドレス) | |
| 論理和<br>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<br>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<br>ComPare Arithmetic | CPA | r1,r2 | (r1) と (r2)，又は (r) と (実効アドレス) の算術比較又は論理比較を行い，比較結果によって，FR に次の値を設定する。 | ○*1 |
| | | r,adr [,x] | | |
| 論理比較<br>ComPare Logical | CPL | r1,r2 | | |
| | | r,adr [,x] | | |

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

(4) シフト演算命令

| | | | | |
|---|---|---|---|---|
| 算術左シフト<br>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。<br>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<br>Shift Right Arithmetic | SRA | r,adr [,x] | | |
| 論理左シフト<br>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。<br>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<br>Shift Right Logical | SRL | r,adr [,x] | | |

(5) 分岐命令

| | | | | |
|---|---|---|---|---|
| 正分岐<br>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<br>Jump on MInus | JMI | adr [,x] | | |
| 非零分岐<br>Jump on Non Zero | JNZ | adr [,x] | | |
| 零分岐<br>Jump on ZEro | JZE | adr [,x] | | |
| オーバフロー分岐<br>Jump on OVerflow | JOV | adr [,x] | | |
| 無条件分岐<br>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<br>PUSH | PUSH adr [,x] | SP ← (SP) $-_L$ 1,<br>(SP) ← 実効アドレス | — |
| ポップ<br>POP | POP r | r ← ( (SP) ),<br>SP ← (SP) $+_L$ 1 | |

(7) コール，リターン命令

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

(8) その他

| | | | |
|---|---|---|---|
| スーパバイザコール<br>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<br>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<br>プログラムの実行開始番地を定義<br>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。<br>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。<br>リテラルは，一つの 10 進定数，16 進定数又は文字定数の前に等号（=）を付けて記述する。CASL Ⅱ は，等号の後の定数をオペランドとする DC 命令を生成し，そのアドレスを adr の値とする。 |

### 2.6 その他

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

(2) 生成された命令語，領域は，主記憶上で連続した領域を占める。

## 3. プログラム実行の手引

### 3.1 OS

プログラムの実行に関して，次の取決めがある。

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

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

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

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

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

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

### 3.2 未定義事項

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

# 表計算ソフトの機能・用語（基本情報技術者試験用）

表計算ソフトの機能，用語などは，原則として次による。

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

## 1. ワークシート

(1) 列と行とで構成される升目の作業領域をワークシートという。ワークシートの大きさは 256 列，10,000 行とする。

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

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

## 2. セルとセル範囲

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

［例］列 A 行 1 にあるセルのセル番地は，A1 と表す。

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

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

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

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

## 3. 値と式

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

(2) 文字列は一重引用符“ ’ ”で囲って表す。

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

(3) 論理値の真を true，偽を false と表す。

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

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

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

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

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

**4. 演算子**

(1) 単項演算子は，正符号“＋”及び負符号 “－”とする。

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

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

(4) 括弧は丸括弧“(”及び“)”を使う。

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

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

**5. セルの複写**

(1) セルの値又は式を，他のセルに複写することができる。

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

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

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

［例］セル B1 に式 $A$1 ＋ $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 $A$1 + $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。<br>［例］合計 (A1 ～ B5) は，　セル範囲 A1～B5 に含まれる数値の合計を返す。 |
| 平均 (セル範囲[1]) | セル範囲に含まれる数値の平均を返す。 |
| 標本標準偏差 (セル 範囲[1]) | セル範囲に含まれる数値を標本として計算した標準偏差を返す。 |
| 母標準偏差 (セル範囲[1]) | セル範囲に含まれる数値を母集団として計算した標準偏差を返す。 |
| 最大 (セル範囲[1]) | セル範囲に含まれる数値の最大値を返す。 |
| 最小 (セル範囲[1]) | セル範囲に含まれる数値の最小値を返す。 |
| IF ( 論理式 , 式1, 式2) | 論理式の値が true のとき式 1 の値を，false のとき式 2 の値を返す。<br>［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。<br>［例1］条件付個数 (H5 ～ L9, ＞ A1) は，セル範囲 H5 ～ L9 のセルのうち，セル A1 の値より大きな値をもつセルの個数を返す。<br>［例2］条件付個数 (H5 ～ L9, ＝ 'A4') は，セル範囲 H5 ～ L9 のセルのうち，文字列 “A4 ” をもつセルの個数を返す。 |
| 整数部 ( 算術式 ) | 算術式の値以下で最大の整数を返す。<br>［例1］整数部 (3.9) は，3 を返す。<br>［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。<br>［例1］剰余 (10,3) は，1 を返す。<br>［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）<br>四捨五入（算術式，桁位置）<br>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。<br>［例1］切上げ(−314.159,2) は，−314.16 を返す。<br>［例2］切上げ(314.159,−2) は，400 を返す。<br>［例3］切上げ(314.159,0) は，315 を返す。 |
| 結合(式1,式2,…)[2] | 式1，式2，… のそれぞれの値を文字列として扱い，それらを引数の順につないでできる一つの文字列を返す。<br>［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。<br>［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。<br>・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。<br>・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。<br>［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。<br>・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。<br>・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。<br>［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。<br>［例］照合検索(15,A1 ～ A8,C6 ～ C13) は，セル範囲 A1 ～ A8 をセル A1，A2，… と探す。このとき，セル A5 で 15 を最初に見つけたとすると，セル範囲 C6 ～ C13 の上端から数えて 5 番目のセル C10 の値を返す。 |

| | |
|---|---|
| 照合一致(式,セル範囲,検索の指定) | 1行又は1列を対象とするセル範囲に対して，セル範囲の左端又は上端から走査し，検索の指定によって指定される条件を満たす最初のセルを探す。見つかったセルの位置を，セル範囲の左端又は上端から 1，2，… と数えた値とし，その値を返す。<br>・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。<br>・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。<br>・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。<br>［例］照合一致(15,B2 ～ B12,−1) は，セル範囲 B2 ～ B12 をセル B2，B3，…と探す。このとき，15 以上の最小値をセルB9 で最初に見つけたとすると，セル B2 から数えた値 8 を返す。 |
| 条件付合計(検索のセル範囲,検索条件の記述,合計のセル範囲[1)]) | 行数及び列数が共に同じ検索のセル範囲と合計のセル範囲に対して，検索と合計を行う。検索のセル範囲に含まれるセルのうち，検索条件の記述で指定される条件を満たすセルを全て探す。検索条件の記述を満たした各セルについての左上端からの位置と，合計のセル範囲中で同じ位置にある各セルの値を合計して返す。<br>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。<br>［例1］条件付合計(A1 ～ B8,＞ E1,C2 ～ D9) は，検索のセル範囲である A1 ～ B8 のうち，セル E1 の値より大きな値をもつ全てのセルを探す。このとき，セル A2，B4，B7 が見つかったとすると，合計のセル範囲である C2 ～ D9 の左上端からの位置が同じであるセル C3，D5，D8 の値を合計して返す。<br>［例2］条件付合計(A1 ～ B8,＝ 160,C2 ～ D9) は，検索のセル範囲である A1 ～ B8 のうち，160 と一致する値をもつ全てのセルを探す。このとき，セル A2，B4，B7 が見つかったとすると，合計のセル範囲である C2 ～ D9 の左上端からの位置が同じであるセル C3，D5，D8 の値を合計して返す。 |

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

[2)] 引数として渡すことができる式の個数は，1 以上である。

## 7. マクロ

### (1) ワークシートとマクロ

ワークシートには複数のマクロを格納することができる。

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

［例］○**マクロ**：Pro

例は，マクロ Pro の宣言である。

### (2) 変数とセル変数

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

［例］○**数値型**：row，col

例は，数値型の変数 row，col の宣言である。

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

セル変数の表現方法には，絶対表現と相対表現とがある。

セル変数の絶対表現は，セル番地で表す。

セル変数の相対表現は，次の書式で表す。

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

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

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

(3) 配列

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

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

［例］○**文字列型**：table[100, 200]

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

(4) 宣言，注釈及び処理

宣言，注釈及び処理の記述は，“共通に使用される擬似言語の記述形式”に従う。

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

［例］○**数値型**：row

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

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

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

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

| 退室可能時間 | 13:40 ～ 15:20 |
|---|---|

7. **問題に関する質問にはお答えできません。**文意どおり解釈してください。
8. 問題冊子の余白などは，適宜利用して構いません。
9. Java プログラムで使用する API の説明，アセンブラ言語の仕様及び表計算ソフトの機能・用語は，この冊子の末尾を参照してください。
10. 試験時間中，机上に置けるもの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，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。