

# 平成 18 年度 秋期 基本情報技術者 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。
2. この注意事項は、問題冊子の裏表紙に続きます。必ず読んでください。
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 5	問 6 ~ 問 9	問 10 ~ 問 13
選択方法	全問必須	1 問選択	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) HB の黒鉛筆又はシャープペンシルを使用してください。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) 答案用紙は光学式読取り装置で処理しますので、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。
  - (3) 受験番号欄に、受験番号を記入及びマークしてください。正しくマークされていない場合、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。
  - (4) 生年月日欄に、受験票に印字されているとおりの生年月日を記入及びマークしてください。正しくマークされていない場合は、採点されないことがあります。
  - (5) 選択した問題については、次の例に従って、〔問 6 と問 10 を選択した場合の例〕  
 選択欄の問題番号の (選) をマークしてください。  
 マークがない場合は、採点の対象になりません。
  - (6) 解答は、次の例題にならって、解答欄にマークしてください。

選択欄					
問 1	<input type="radio"/>	問 6	<input type="radio"/>	問 10	<input type="radio"/>
問 2	<input type="radio"/>	問 7	<input checked="" type="radio"/>	問 11	<input checked="" type="radio"/>
問 3	<input type="radio"/>	問 8	<input checked="" type="radio"/>	問 12	<input checked="" type="radio"/>
問 4	<input type="radio"/>	問 9	<input checked="" type="radio"/>	問 13	<input checked="" type="radio"/>
問 5	<input type="radio"/>				

〔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。


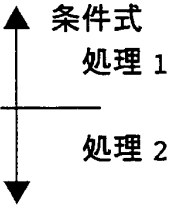
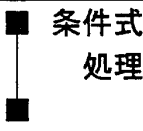
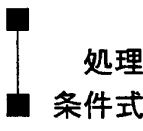
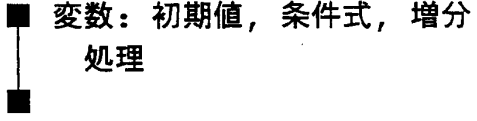
例題	a	(ア)	(イ)	<input checked="" type="radio"/>	(エ)
----	---	-----	-----	----------------------------------	-----

裏表紙の注意事項も、  
必ず読んでください。

## 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。
・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。
	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。
	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。
	前判定繰返し処理を示す。 条件式が真の間, 処理を繰返し実行する。
	後判定繰返し処理を示す。 処理を実行し, 条件式が真の間, 処理を繰返し実行する。
	繰返し処理を示す。 開始時点で変数に初期値 (定数又は式で与えられる) が格納され, 条件式が真の間, 処理を繰返す。また, 繰返すごとに, 変数に増分 (定数又は式で与えられる) 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 ↑ ↓ 低
乗除演算	×, ÷, %	
加減演算	+, -	
関係演算	>, <, ≥, ≤, =, ≠	
論理積	and	
論理和	or	

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

〔論理型の定数〕

true, false

次の問1から問5までの5問については、全問解答してください。

問1 ハードディスク装置に関する次の記述を読んで、設問に答えよ。

X社は、同型のハードディスク装置1,200台で構成されるデータセンタを運営している。このうち、1年当たり平均24台の装置が故障し、置き換えられる（以下、交換という）。故障によるハードディスク装置の交換には、1台当たり10万円の交換費用がかかるが、装置購入費（装置価格）はかからない。

また、データセンタを構成するハードディスク装置は、耐用年数に達する前に、毎年200台ずつ計画的に置き換えられる（以下、入替えという）。この入替台数は、故障の発生数とは無関係である。入替費用は、次式で計算される。作業費は1万円である。

$$\text{入替費用} = (\text{装置価格} + \text{作業費}) \times \text{入替台数}$$

X社は、来年ハードディスク装置の総入替えを行うことになり、新たに利用するハードディスク装置を、表に示す機種の中から選定する。各機種は、MTTF（Mean Time To Failure：故障までの平均時間）と装置価格が異なる。表中の $y$ は現在使用しているハードディスク装置のMTTFと同じ値である。MTTFは、次式で計算される。

$$\text{MTTF} = \text{延べ稼働時間} \div \text{故障台数}$$

表 選定候補機種リスト

機種	装置価格(円)	MTTF(年)
A	50,000	$y$
B	52,500	$1.2y$
C	54,000	$1.6y$
D	56,000	$2y$

MTTFが $y$ のときの1年当たりの平均故障台数が24台であることから、データセンタで利用するハードディスク装置を、MTTFが $1.2y$ である機種Bに総入替えすると、1年当たりの平均故障台数は $24 \div 1.2 = 20$ （台）になると見込まれる。

設問 次の記述中の  に入れる正しい答えを、解答群の中から選べ。

X 社のデータセンタが現在利用しているハードディスク装置の MTTF は、 a  年である。また、ハードディスク装置を、MTTF が 100 年の機種に総入替えすると、1 年当たりの平均故障台数は  b  台になると見込まれる。

データセンタで利用するハードディスク装置を、表中のいずれかの機種に総入替えした後の、ハードディスク装置の年間入替え及び故障による交換にかかる総費用が最も安くなると見込まれる機種は  c  であり、最も高くなる機種は  d  である。総費用は、機種の違いによって、次式で計算される額だけ差が出る。

$$\begin{aligned} \text{機種の違いによる総費用の差額} &= (\text{装置価格の差} \times \text{入替台数}) \\ &\quad + (\text{交換費用} \times \text{故障台数の差}) \\ &= (\text{装置価格の差} \times 200) \\ &\quad + (100,000 \times \text{故障台数の差}) \end{aligned}$$

a, b に関する解答群

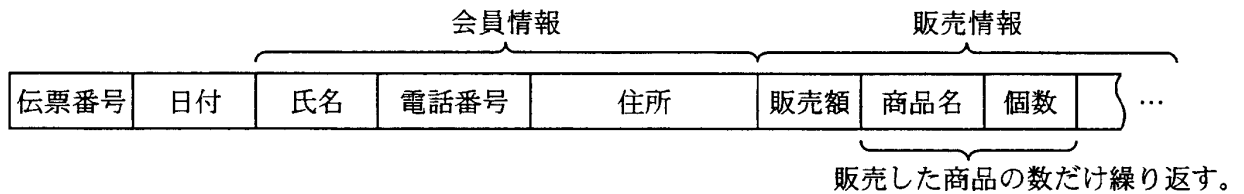
ア 0	イ 1	ウ 2	エ 5
オ 6	カ 12	キ 24	ク 50

c, d に関する解答群

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

問2 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。

ある酒屋では、次に示すファイル様式で管理していた会員への販売データをデータベース化することにした。



伝票番号は、1回の支払いに対して一つ割り振られる。

管理すべき情報を精査し、情報を追加するとともに正規化を実施して、次に示す四つの表からなるデータベースを設計した。網掛けした項目は主キーである。

会員表

会員ID	氏名	電話番号	住所
020001	山田太郎	03-0000-0001	東京都文京区…
020002	中村二郎	03-0000-0002	東京都新宿区…
⋮	⋮	⋮	⋮

販売表

伝票番号	日付	会員ID	販売額
00000001	20061002	020002	4490
00000002	20061002	040027	2000
00000003	20061002	020004	12700
⋮	⋮	⋮	⋮

販売明細表

伝票番号	商品コード	個数
00000001	BE0001	3
00000001	NI0002	1
00000002	WI0001	1
⋮	⋮	⋮

商品表

商品コード	分類	商品名	単価
BE0001	ビール	Aビール	230
BE0002	ビール	Bビール	230
WI0001	ワイン	Cワイン	2000
⋮	⋮	⋮	⋮

設問1 データベース設計における第1正規化の考えに基づいて実施した操作を、解答群の中から選べ。

解答群

- ア 会員表を作成し、データの重複をなくした。
- イ 商品表を作成し、情報の独立性を高めた。
- ウ 販売明細表を作成し、販売情報から繰返し要素を排除した。

設問2 データベース設計における正規化による直接の効果として適切なものを、解答群の中から選べ。

解答群

- ア 会員情報の集約によって、セキュリティが強化できる。
- イ 表間の独立性が高まり、会員情報や商品情報を更新する際の影響を局所化できる。
- ウ 将来、OSやDBMSなどを変更する際、移行が容易になる。
- エ 会員の購入歴など、複数の表を連結してデータを検索する際、処理性能が向上する。

設問3 作成したデータベースを利用して、販売促進のためのデータを抽出する。1回の支払いが1万円以上の買物をした会員の氏名と住所を抽出したい。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 DISTINCT 会員表.氏名, 会員表.住所  
FROM 会員表, 販売表 WHERE 
```

解答群

- ア MAX(販売表.販売額) >= 10000
- イ SUM(販売表.販売額) >= 10000
- ウ 販売表.販売額 >= 10000
- エ 会員表.会員ID = 販売表.会員ID AND 販売表.販売額 >= 10000
- オ 会員表.会員ID = 販売表.会員ID AND MAX(販売表.販売額) >= 10000
- カ 会員表.会員ID = 販売表.会員ID AND SUM(販売表.販売額) >= 10000

設問4 過去にワインの購入歴がある会員の氏名と住所を抽出したい。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 DISTINCT 会員表.氏名, 会員表.住所  
FROM 会員表, 販売表, 販売明細表, 商品表 WHERE 
```

解答群

- ア 会員表.会員 ID IN (SELECT 販売表.会員 ID FROM 販売表, 商品表  
WHERE 商品表.分類 = 'ワイン')
- イ 会員表.会員 ID = (SELECT 販売表.会員 ID FROM 販売表, 商品表  
WHERE 商品表.分類 = 'ワイン')
- ウ 会員表.会員 ID = 販売表.会員 ID  
AND 販売明細表.商品コード = 商品表.商品コード  
AND 商品表.分類 = 'ワイン'
- エ 会員表.会員 ID = 販売表.会員 ID  
AND 販売表.伝票番号 = 販売明細表.伝票番号  
AND 販売明細表.商品コード = 商品表.商品コード  
AND 商品表.分類 = 'ワイン'



問3 次のプログラムの説明及びプログラムを読んで、設問に答えよ。

[プログラムの説明]

次の  $n$  元連立 1 次方程式の解、すなわち  $x_i (i = 1, 2, \dots, n)$  の値を求める副プログラム Gauss である。

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

(1) Gauss は、前進消去と後退代入の二つの段階からなる。

① 前進消去

次の計算を  $k = 1, 2, \dots, n-1$  の順に行う。 $k$  は前進消去の計算回数であり、変数の右肩に括弧付きの添字で示す。ここで、 $a_{ij}^{(0)} = a_{ij}$ 、 $b_i^{(0)} = b_i$  とする。 $a_{kk}^{(k-1)}$  をピボット (Pivot) と呼び、計算の途中で 0 にならないものとする。

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{kj}^{(k-1)} \times (a_{ik}^{(k-1)} / a_{kk}^{(k-1)}) \quad (i = k+1, \dots, n; j = k+1, \dots, n)$$

$$b_i^{(k)} = b_i^{(k-1)} - b_k^{(k-1)} \times (a_{ik}^{(k-1)} / a_{kk}^{(k-1)}) \quad (i = k+1, \dots, n)$$

この計算の結果、 $n$  元連立 1 次方程式は次の形になる。

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + \dots + a_{1n}^{(0)}x_n = b_1^{(0)} \\ a_{22}^{(1)}x_2 + \dots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \vdots \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

② 後退代入

①の結果から、 $n$  元連立 1 次方程式の解  $x_i$  を  $i = n, n-1, \dots, 1$  の順に求める。

$$x_n = b_n^{(n-1)} / a_{nn}^{(n-1)}$$

$$x_i = \left( b_i^{(i-1)} - \sum_{j=i+1}^n (a_{ij}^{(i-1)} \times x_j) \right) / a_{ii}^{(i-1)} \quad (i = n-1, \dots, 1)$$

(2) 次の連立方程式を例として、前進消去と後退代入の計算を示す。

$$\begin{cases} 2x_1 + x_2 + 2x_3 = 2 \\ 3x_1 + 2x_2 + x_3 = 6 \\ 2x_1 + 4x_2 + x_3 = 9 \end{cases}$$

① 前進消去の計算を  $k = 1, 2$  の順に行う。

$k = 1$  のとき、 $a_{22}^{(1)}$  は次のとおりに計算される。

$$\begin{aligned} a_{22}^{(1)} &= a_{22}^{(0)} - a_{12}^{(0)} \times (a_{21}^{(0)} / a_{11}^{(0)}) \\ &= 2 - 1 \times (3 / 2) = 0.5 \end{aligned}$$

同様に  $a_{23}^{(1)}$ ,  $b_2^{(1)}$ ,  $a_{32}^{(1)}$ ,  $a_{33}^{(1)}$ ,  $b_3^{(1)}$  を計算すると、連立方程式は次のとおりになる。

$$\begin{cases} 2x_1 + x_2 + 2x_3 = 2 \\ 0.5x_2 - 2x_3 = 3 \\ 3x_2 - x_3 = 7 \end{cases}$$

$k = 2$  のとき、 $a_{33}^{(2)}$ ,  $b_3^{(2)}$  を計算すると、連立方程式は次のとおりになる。

$$\begin{cases} 2x_1 + x_2 + 2x_3 = 2 \\ 0.5x_2 - 2x_3 = 3 \\ 11x_3 = -11 \end{cases}$$

② 後退代入によって、 $x_3$ ,  $x_2$ ,  $x_1$  の値を得る。

$$\begin{cases} x_3 = -11 / 11 = -1 \\ x_2 = (3 + 2 \times (-1)) / 0.5 = 2 \\ x_1 = (2 - 1 \times 2 - 2 \times (-1)) / 2 = 1 \end{cases}$$

(3) Gauss の引数の仕様を表に示す。

なお、各配列の添字は 1 から始まる。

表 Gauss の引数の仕様

引数	データ型	入力/出力	意味
n	整数型	入力	連立方程式の元数 (未知数の個数)
a[, ]	実数型	入力	係数 $a_{ij}$ が格納されている 2 次元配列
b[ ]	実数型	入力	定数 $b_i$ が格納されている 1 次元配列
x[ ]	実数型	出力	解 $x_i$ の値を格納する 1 次元配列

[プログラム]

○Gauss(整数型: n, 実数型: a[, ], 実数型: b[, ], 実数型: x[ ])

○整数型: k, i, j

○実数型: Pivot, Factor, Sum

/\* ----- 前進消去 ----- \*/

```
■ k: 1, k ≤ n-1, 1
  ■ a
    ■ i: k+1, i ≤ n, 1
      ■ Factor ← a[i,k] ÷ Pivot
        ■ j: k+1, j ≤ n, 1
          ■ a[i,j] ← a[i,j] - a[k,j] × Factor
            ■ b[i] ← b[i] - b[k] × Factor
        ■
      ■
    ■
```

/\* ----- 後退代入 ----- \*/

```
■ x[n] ← b[n] ÷ a[n,n]
  ■ b
    ■ Sum ← 0.0
      ■ c
        ■ Sum ← Sum + a[i,j] × x[j]
          ■
        ■
      ■
    ■ x[i] ← (b[i] - Sum) ÷ a[i,i]
  ■
```

設問 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の解答群

ア Pivot ← a[1,1]

イ Pivot ← a[1,k]

ウ Pivot ← a[k,k]

エ Pivot ← a[n,n]

b, cの解答群

ア i: 1, i ≤ n, 1

イ i: n-1, i ≥ 1, -1

ウ j: 1, j ≤ i, 1

エ j: 1, j ≤ n-1, 1

オ j: i+1, j ≤ n, 1

問4 次のプログラムの説明及びプログラムを読んで、設問1, 2に答えよ。

[プログラム1の説明]

副プログラム TabSpcl は、タブ文字を展開するプログラムである。

(1) TabSpcl は、文字型配列 Src を先頭から調べ、Src 中のすべてのタブ文字をそれぞれ一つ以上の間隔文字（スペース）に置換して、文字型配列 Dst に格納する。タブ文字以外の文字は、そのまま Dst に格納する。

(2) 文字型配列の各要素には、文字を1文字ずつ順に格納し、最後の文字の次の要素にはシステム定数である EOS を格納する。

なお、配列の添字は1から始まり、添字の値を文字位置と呼ぶ。

(3) Src 中にタブ文字が出現した場合、次の文字が最も近い右のタブ位置に格納されるように、タブ文字を一つ以上の間隔文字に置換して、Dst に格納する。

ここで、タブ位置とは、引数で渡されるタブ間隔 ( $\geq 2$ ) を用いた次の式で計算される文字位置である。

$$\text{タブ位置} = \text{タブ間隔} \times n + 1 \quad (n = 1, 2, \dots)$$

(4) タブ間隔が4のときの実行例を図1に示す。

“j” を Dst のタブ位置である文字位置 13 ( $= 4 \times 3 + 1$ ) に格納したのでは、タブ文字が間隔文字に置き換わらないので、最も近い右のタブ位置である文字位置 17 ( $= 4 \times 4 + 1$ ) に格納する。

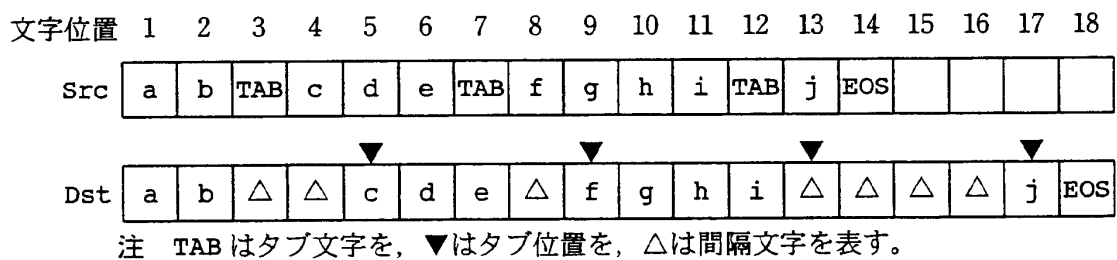


図1 タブ間隔が4のときの TabSpcl の実行例

(5) Dst は十分に大きいものとする。

(6) 副プログラム TabSpcl の引数の仕様を表1に示す。

表 1 TabSpcl の引数の仕様

引数	データ型	入力/出力	意味
Src[]	文字型	入力	対象となる文字型配列
Dst[]	文字型	出力	出力する文字型配列
TabGap	整数型	入力	タブ間隔 ( $\geq 2$ )

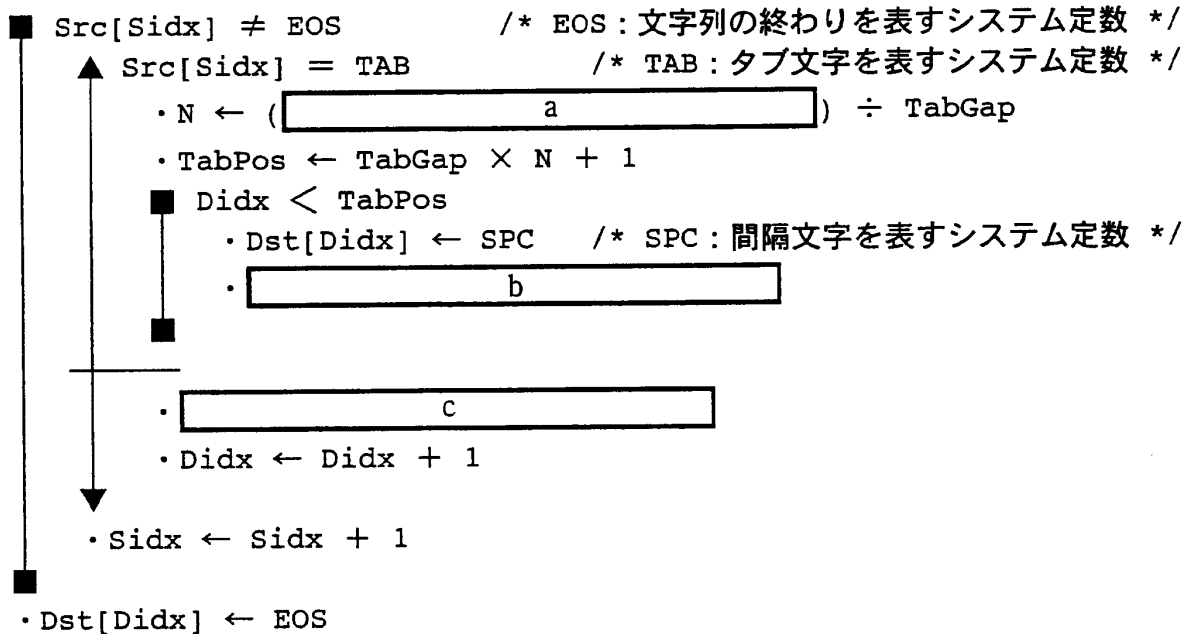
[プログラム 1]

○TabSpcl(文字型: Src[], 文字型: Dst[], 整数型: TabGap)

○整数型: Sidx, Didx, N, TabPos

・Sidx ← 1

・Didx ← 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 Didx + 1

イ Didx - 1

ウ Didx + TabGap + 1

エ Didx + TabGap - 1

b, c に関する解答群

ア Didx ← Didx + 1

イ Dst[Didx] ← Src[Sidx]

ウ Dst[Didx + 1] ← Src[Sidx]

エ Dst[Didx] ← Src[Sidx + 1]

設問 2 プログラム 1 を改造して、タブ位置を任意の文字位置に指定できるようなプログラム 2 を作成した。ただし、タブ位置は 2 以上の値で、隣り合うタブ位置の間隔も 2 以上とする。タブ位置は引数で渡される整数型配列 TabSet に昇順に格納され、最後のタブ位置が格納された次の配列要素には -1 が格納されている。

なお、最後のタブ位置以降にタブ文字が出現した場合は、一つの間隔文字に置換する。

図 1 と同じ動作をさせる場合の TabSet の内容を図 2 に、プログラム 2 に示す副プログラム TabSpc2 の引数の仕様を表 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。ただし、 b ,  c  には設問 1 の正しい答えが入っているものとする。

TabSet[1]	5
TabSet[2]	9
TabSet[3]	13
TabSet[4]	17
⋮	⋮
TabSet[m]	-1

図 2 整数型配列 TabSet の内容 (図 1 と同じ動作をさせる場合)

表 2 TabSpc2 の引数の仕様

引数	データ型	入力/出力	意味
Src[]	文字型	入力	対象となる文字型配列
Dst[]	文字型	出力	出力する文字型配列
TabSet[]	整数型	入力	タブ位置を格納した整数型配列

[プログラム2]

○TabSpc2(文字型: Src[], 文字型: Dst[], 整数型: TabSet[])

○整数型: Sidx, Didx, Tidx, TabPos

○論理型: Loop

・Sidx ← 1

・Didx ← 1

・Tidx ← 1

■ Src[Sidx] ≠ EOS

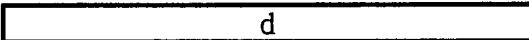
▲ Src[Sidx] = TAB

・TabPos ← Didx + 1

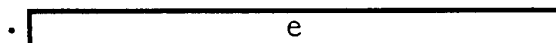
・Loop ← true

■ (TabSet[Tidx] ≠ -1) and (Loop = true)

▲ Didx < TabSet[Tidx]

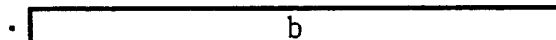
・  d

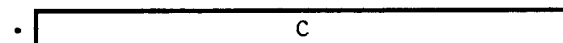
・Loop ← false

・  e

■ Didx < TabPos

・Dst[Didx] ← SPC

・  b

・  c

・Didx ← Didx + 1

・Sidx ← Sidx + 1

■ Dst[Didx] ← EOS

d, e に関する解答群

ア TabPos ← TabSet[Tidx]

イ TabPos ← TabSet[Tidx] + Didx

ウ TabPos ← TabSet[Tidx] - Didx

エ Tidx ← Didx + 1

オ Tidx ← TabPos + 1

カ Tidx ← Tidx + 1

問5 プログラム設計に関する次の記述を読んで、設問1～3に答えよ。

B社は、バター、チーズ、ヨーグルトなど、牛乳を主な原料とする乳製品を製造している。

牛乳は、J牧場、K牧場及びL牧場から仕入れ、個別のタンクに保存されている。製造計画で、ロット番号と呼ばれるコードで識別される製造単位ごとに、単一の牧場の牛乳を使うか、又は標準の混合割合で混合して使うかを定める。ロット番号は、同一製品の同一製造日の製造単位ごとに付けられるコードである。

B社では、製造計画で決められた、製造日ごとの各牧場の牛乳の必要量を集計するプログラムを作成することになった。

[必要量ファイルの説明]

製造計画で決められた、製造日、製品コード、ロット番号ごとに、原料牛乳の“必要量”（正の整数値）と、単一か混合かを表す“牧場区分”が格納されている。

牧場区分は、J牧場、K牧場及びL牧場を表す“J”、“K”、“L”、又は混合を表す“M”のどれかである。

必要量ファイルの様式を図1に示す。レコードは、“製造日”、“製品コード”、“ロット番号”を、この順で整列キーとして、昇順に整列されている。

製造日	製品コード	ロット番号	牛乳の必要量	牧場区分
-----	-------	-------	--------	------

図1 必要量ファイルの様式

[標準混合割合ファイルの説明]

“製品コード”をキーとする索引ファイルである。製品ごとに各牧場の牛乳の標準混合割合が、合計で100%となるように、百分率（0～100の整数値）で格納されている。

標準混合割合ファイルの様式を図2に示す。

製品コード	J牧場の割合	K牧場の割合	L牧場の割合
-------	--------	--------	--------

図2 標準混合割合ファイルの様式



[集計ファイルの説明]

製造日ごとの各牧場の牛乳の必要量を集計して、総必要量として格納する。

集計ファイルの様式を図3に示す。

製造日	J牧場の総必要量	K牧場の総必要量	L牧場の総必要量
-----	----------	----------	----------

図3 集計ファイルの様式

[プログラムの説明]

必要量ファイルを読み、レコードごとに、次の処理①～③のどれかを行う。

- ① 牧場区分が“M”のときは、“混合”であり、標準混合割合ファイルと必要量ファイルの情報から、J牧場、K牧場及びL牧場のそれぞれの必要量を計算する。この計算は、小数点以下切捨てとし、切捨て誤差は、L牧場の牛乳の必要量で補正する。
- ② 牧場区分が“J”、“K”及び“L”のときは、“単一”であり、必要量の全量を、“J”のときはJ牧場、“K”のときはK牧場、“L”のときはL牧場の牛乳とする。
- ③ 牧場区分が“J”、“K”、“L”及び“M”以外のときは、エラーであり、このレコードの内容をエラーファイルに出力する。

エラーとなったものを除き、製造日ごとにJ牧場、K牧場及びL牧場の牛乳の必要量をそれぞれ集計し、集計ファイルに出力する。

入出力関連図を図4に、モジュール構造図を図5に示す。

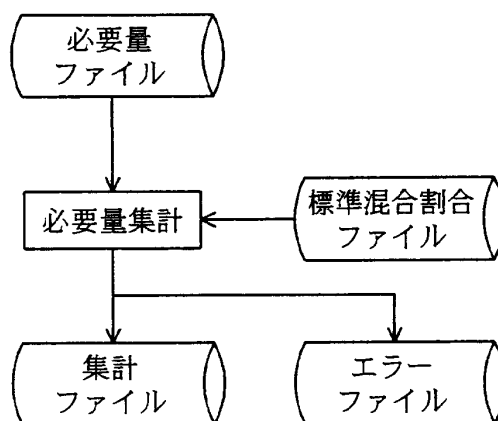


図4 入出力関連図

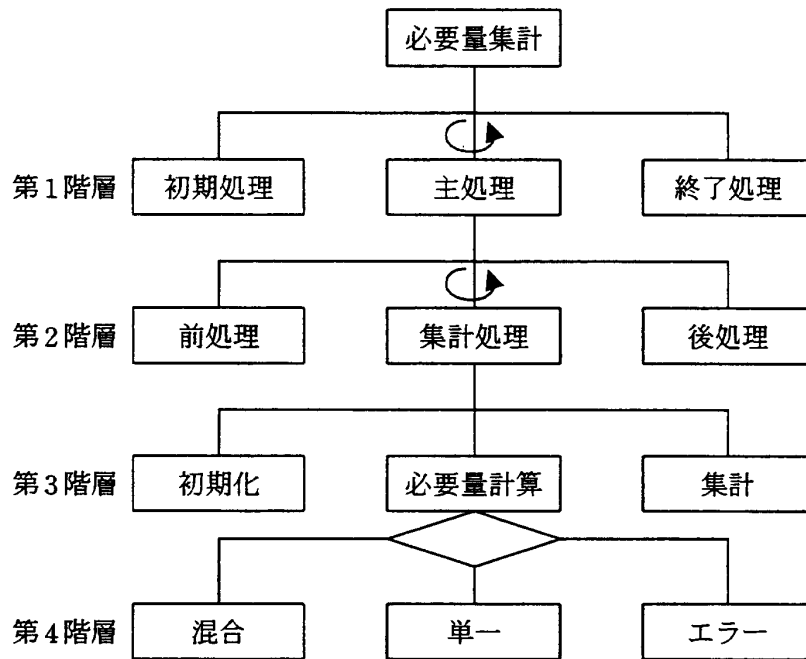


図5 モジュール構造図

設問1 図6は、主処理と必要量計算の流れ図である。図中の  に入れる正しい答えを、解答群の中から選べ。

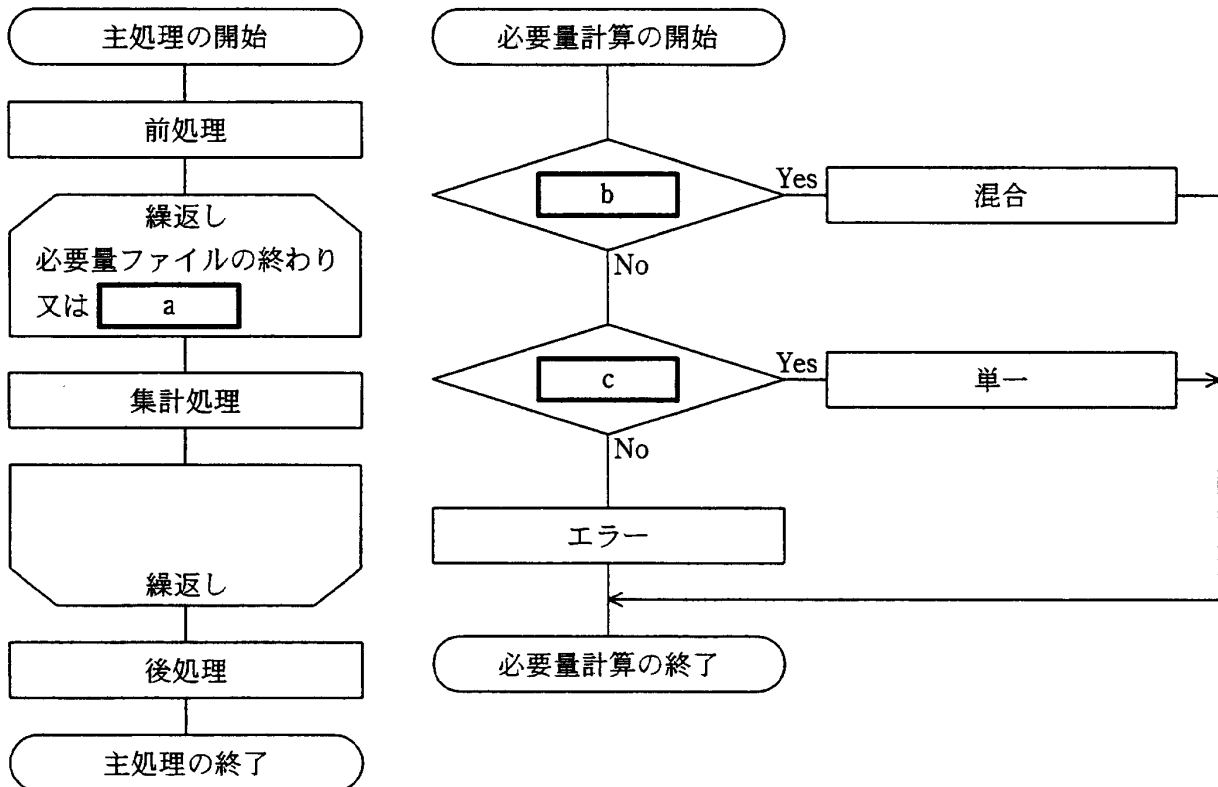


図6 主処理と必要量計算の流れ図

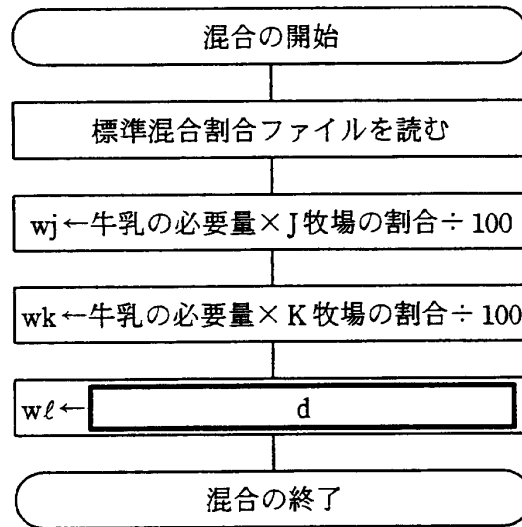
a に関する解答群

- |            |             |
|------------|-------------|
| ア 製造日が変わる  | イ 製品コードが変わる |
| ウ 牧場区分が変わる | エ ロット番号が変わる |

b, c に関する解答群

- ア (牧場区分 ≠ “J”) or (牧場区分 ≠ “K”) or (牧場区分 ≠ “L”)
- イ (牧場区分 ≠ “M”) or ( (牧場区分 ≠ “J”) and (牧場区分 ≠ “K”) and (牧場区分 ≠ “L”) )
- ウ (牧場区分 ≠ “M”) or (牧場区分 ≠ “J”) or (牧場区分 ≠ “K”) or (牧場区分 ≠ “L”)
- エ (牧場区分 = “J”) or (牧場区分 = “K”) or (牧場区分 = “L”)
- オ (牧場区分 = “M”) or ( (牧場区分 ≠ “J”) and (牧場区分 ≠ “K”) and (牧場区分 ≠ “L”) )
- カ 牧場区分 ≠ “M”
- キ 牧場区分 = “M”

設問2 図7は、混合の流れ図である。図中の  に入れる正しい答えを、解答群の中から選べ。



注 “牛乳の必要量”は、必要量ファイルから入力した値である。  
 “J牧場の割合”及び“K牧場の割合”は、標準混合割合ファイルの値である。  
 $w_j$ ,  $w_k$  及び  $w_l$  には、それぞれ、J牧場、K牧場及びL牧場の牛乳の必要量を格納する。  
 除算の計算結果は、小数点以下を切り捨てる。

図7 混合の流れ図

解答群

- ア 牛乳の必要量 -  $w_j$
- イ 牛乳の必要量 -  $w_j - w_k$
- ウ 牛乳の必要量 -  $w_k$
- エ 牛乳の必要量  $\times (100 - \text{J牧場の割合} - \text{K牧場の割合}) \div 100$
- オ 牛乳の必要量  $\times \text{L牧場の割合} \div 100$

設問3 図5中の初期化を除く第3階層及び第4階層の最下位のモジュールのうち、“牧場区分”によって処理を分ける必要があるものを、解答群の中から選べ。  
 ここで、初期化モジュールは、作業用記憶の初期化だけを行うものとする。

解答群

- ア エラー
- イ 混合
- ウ 集計
- エ 単一

次の問6から問9までの4問については、この中から1問を選択し、答案用紙の選択欄の(選)をマークして解答してください。

なお、2問以上選択した場合には、はじめの1問について採点します。

問6 次のCプログラムの説明及びプログラムを読んで、設問に答えよ。

[プログラムの説明]

プログラムが生成した各けたの数字が異なる4けたの目標数を、なるべく少ない回数の推測で当てるゲーム `GuessNumber` である。回答者は、各けたが異なる4けたの推測数を入力し、当たらなかった場合、次の情報を得ることができる。

- ① 目標数に含まれていて、けたも一致している数字の個数（以下、Hit 数という）
- ② 目標数に含まれているが、けたが一致していない数字の個数（以下、Blow 数という）

例えば、目標数が“1632”で推測数が“7613”の場合、“6”はけたも一致しているのでHit数が1、“1”と“3”はけたが一致していないのでBlow数が2となる。

(1) 目標数はプログラムが自動的に生成する。生成した数字列は `char` 型の配列 `target` に格納される。推測数を文字列として標準入力から読み込み、`char` 型の配列 `num` に格納したうえで照合する。目標数と完全に一致するまで繰り返す。

(2) 次の関数が用意されている。

```
void createRandomNumber(char[] target)
```

機能：4けたの目標数を文字列として左のけたから順に配列 `target` に格納する。各けたの数字はすべて異なっている。

```
int isValidNumber(char[] num)
```

機能：文字列 `num` 中の各文字がすべて異なる数字のとき `TRUE` を、それ以外  
のとき `FALSE` を返す。

[プログラム]

```
#include <stdio.h>
#define DIGITS 4 /* けた数 */
#define TRUE 1
#define FALSE 0

void createRandomNumber(char[]);
int isValidNumber(char[]);
void GuessNumber();
int isMatch(char[], char[]);

void GuessNumber(){
    char target[DIGITS + 1]; /* 目標数の保存領域 */
    char num[DIGITS + 1]; /* 推測数の保存領域 */
    int count = 0; /* 推測回数 */

    createRandomNumber(target); /* 目標数の生成 */
    do{
        printf("[%d回目] 各けたが異なる%dけたの数を入力してください:",
                ++count, DIGITS);

        scanf("%4s", num);
        while( [a] ){
            printf("入力が正しくありません。再度入力してください:");
            scanf("%4s", num);
        }
    }while( [b] );
}

int isMatch(char target[], char num[]){
    int i, j, numHit = 0, numBlow = 0;

    for(i = 0; i < DIGITS; i++){
        for(j = 0; j < DIGITS; j++){
            if( [c] ){
                if( [d] ){
                    numHit++;
                } else {
                    numBlow++;
                }
            }
        }
    }
    if( [e] ){
        printf("正解です。 \n");
        return TRUE;
    }
    printf("%sは%dHit,%dBlowです。 \n\n", num, numHit, numBlow);
    return FALSE;
}
```

設問 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 `isMatch(target, num) == FALSE`
- イ `isMatch(target, num) == TRUE`
- ウ `isValidNumber(num) == FALSE`
- エ `isValidNumber(num) == TRUE`

c に関する解答群

- ア `num[i] == num[j]`
- イ `target[i] == num[i]`
- ウ `target[i] == num[j]`
- エ `target[i] == target[j]`
- オ `target[j] == num[j]`

d に関する解答群

- ア `i != j`
- イ `i < j`
- ウ `i <= j`
- エ `i == j`
- オ `i > j`
- カ `i >= j`

e に関する解答群

- ア `numHit != DIGITS`
- イ `numHit + numBlow == DIGITS`
- ウ `numHit + numBlow >= DIGITS`
- エ `numHit == DIGITS`
- オ `numHit == numBlow`
- カ `numHit > numBlow`

問7 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]

更新指示に基づき、電話帳ファイルのレコードを追加、削除するプログラムである。

(1) 電話帳ファイルのレコード様式は、次のとおりである。

氏名キー		電話番号 11 けた
氏名 20 けた	重複番号 1 けた	

- ① 氏名キーをキーとする索引ファイルである。
- ② 重複番号の初期値は 1 とし、同じ氏名の人登録されるたびに 1 ずつ増える。同じ氏名の人 9 人まで登録できるものとする。

例：

氏名	重複番号	電話番号
ｽｽﾞｷ ｷﾀﾞ	1	06012345678
ｽｽﾞｷ ｷﾀﾞ	2	08087654321

(2) 更新指示は、32 けたであり、次のレコード様式で入力される。

指示 1 けた	氏名 20 けた	電話番号 11 けた
------------	-------------	---------------

- ① 指示には、追加する場合“ I ”、削除する場合“ D ”、プログラムを終了する場合“ Q ”が入力される。“ I ”又は“ D ”のとき、氏名及び電話番号が省略されることはなく、各項目は左詰めで、残ったけたには空白が入力される。“ Q ”のとき、後ろ 31 けたには空白が詰められる。
- ② 指示が“ I ”のとき、電話帳ファイルに同じ氏名の人登録されていない場合は、重複番号を 1 として電話帳ファイルに登録する。登録されている場合は、最後の重複番号に 1 を加えた数を、追加する人の重複番号とする。
- ③ 指示が“ D ”のとき、電話帳ファイルから氏名と電話番号が一致するレコードを検索し、削除する。同じ氏名の人ほかに登録されている場合は、重複番号が欠番にならないよう、後ろの人の重複番号を 1 ずつ繰り上げる。
- ④ 指示が“ Q ”のとき、プログラムを終了する。
- ⑤ 入力されるデータに誤りはなく、追加・削除の処理でエラーが発生した場合は、エラーメッセージを表示し、プログラムを終了させる。



[プログラム]

DATA DIVISION.

FILE SECTION.

FD DENWA-F.

01 DENWA-R.

03 D-KEY.

05 D-SHIMEI PIC X(20).

05 D-BANGO PIC 9.

03 D-TEL PIC X(11).

WORKING-STORAGE SECTION.

01 END-SW1 PIC X(3).

01 END-SW2 PIC X(3).

01 W-BANGO PIC 9(2).

01 W-DATA.

03 W-SHIJI PIC X.

03 W-SHIMEI PIC X(20).

03 W-TEL PIC X(11).

PROCEDURE DIVISION.

SHORI.

OPEN I-O DENWA-F.

MOVE SPACE TO END-SW1.

PERFORM UNTIL END-SW1 = "END"

DISPLAY "INPUT SHIJI:I/D SHIMEI(20) TEL(11) OR Q"

ACCEPT W-DATA

MOVE SPACE TO END-SW2

EVALUATE W-SHIJI WHEN "I" PERFORM TSUIKA

WHEN "D" PERFORM SAKUJO

WHEN "Q" MOVE "END" TO END-SW1

END-EVALUATE

END-PERFORM.

CLOSE DENWA-F.

STOP RUN.

TSUIKA.

PERFORM VARYING W-BANGO FROM 1 BY 1

UNTIL W-BANGO > 9 OR END-SW2 = "END"

MOVE W-SHIMEI TO D-SHIMEI

MOVE W-BANGO TO D-BANGO

READ DENWA-F

a

MOVE W-SHIMEI TO D-SHIMEI

MOVE W-BANGO TO D-BANGO

MOVE W-TEL TO D-TEL

WRITE DENWA-R

INVALID DISPLAY "WRITE ERROR", W-DATA

STOP RUN

END-WRITE

MOVE "END" TO END-SW2

NOT a IF W-BANGO = 9 THEN

DISPLAY "OVER 9", W-DATA

STOP RUN

END-IF

END-READ

END-PERFORM.

COBOL

SAKUJO.

```
PERFORM VARYING W-BANGO FROM 1 BY 1
                                UNTIL W-BANGO > 9 OR END-SW2 = "END"
MOVE W-SHIMEI TO D-SHIMEI
MOVE W-BANGO TO D-BANGO
READ DENWA-F
    [ b ] DISPLAY "READ ERROR", W-DATA
        STOP RUN
    [ c ] IF W-TEL = D-TEL THEN
        DELETE DENWA-F
            INVALID DISPLAY "DELETE ERROR", W-DATA
            STOP RUN
        END-DELETE
        MOVE "END" TO END-SW2
    END-IF
END-READ
END-PERFORM.
PERFORM [ d ]
MOVE W-SHIMEI TO D-SHIMEI
MOVE W-BANGO TO D-BANGO
READ DENWA-F
    INVALID MOVE 10 TO W-BANGO
    NOT INVALID [ e ]
END-READ
COMPUTE W-BANGO = W-BANGO + 1
END-PERFORM.
```

COBOL

設問1 プログラム中の [ a ] ~ [ d ] に入れる正しい答えを、解答群の中から選べ。解答は、重複して選んでもよい。

解答群

- ア AT END
- イ INVALID
- ウ NOT AT END
- エ NOT INVALID
- オ UNTIL W-BANGO = 9
- カ UNTIL W-BANGO > 9
- キ VARYING W-BANGO FROM 1 BY 1 UNTIL W-BANGO = 9
- ク VARYING W-BANGO FROM 1 BY 1 UNTIL W-BANGO > 9

設問2 プログラム中の e に入る正しい文の組合せを、解答群の中から選べ。  
なお、各文の実行は矢印の順に行うものとする。

- ① COMPUTE D-BANGO = D-BANGO - 1
- ② COMPUTE D-BANGO = D-BANGO + 1
- ③ DELETE DENWA-F  
    INVALID DISPLAY "DELETE ERROR2", W-SHIMEI, W-BANGO  
    STOP RUN  
END-DELETE
- ④ REWRITE DENWA-R  
    INVALID DISPLAY "REWRITE ERROR", W-SHIMEI, W-BANGO  
    STOP RUN  
END-REWRITE
- ⑤ WRITE DENWA-R  
    INVALID DISPLAY "WRITE ERROR2", W-SHIMEI, W-BANGO  
    STOP RUN  
END-WRITE

解答群

ア ①→④

イ ①→⑤

ウ ①→⑤→②→③

エ ③→①→⑤

オ ③→②→⑤

問8 次のJavaプログラムの説明及びプログラムを読んで、設問に答えよ。

(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]

クラス `UniqueMap` は、キーと値を1対1に対応付けて保持する。

例えば、乗物の乗客(キー)と座席(値)との間に着席という関連があるとき、一つの座席には多くても1人の乗客だけが座り、1人の乗客は一つの席にしか座らない。クラス `UniqueMap` はこのような2種類のオブジェクトを、一方をキー、他方を値として1対1に対応させるクラスであり、次のメソッドをもつ。

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

キー (`key`) と値 (`value`) を1対1に対応付けて登録する。`key` 又は `value` が `null` ならば、`NullPointerException` を投げる。また、`value` が既にほかのキーと対応付けられていれば、`IllegalArgumentException` を投げる。

`key` が既にほか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)
```

キー (`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)
```

キー (`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 void putForce(K key, V value)
```

キー (`key`) と値 (`value`) を1対1に対応付けて登録する。`value` が既にほかのキーと対応付けられていれば、その対応を削除したうえで登録する。戻り値はない。その他の仕様は、メソッド `put` と同一である。

[プログラム]

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

public class UniqueMap<K, V> { // Kはキーのクラス, Vは値のクラス。
    // キーと値を対応させて記憶する。
    private Map<K, V> map = new HashMap<K, V>();
    // 値からキーを取り出せるようにする。
    private Map<V, K> reverse = new HashMap<V, K>();

    public V put(K key, V value) {
        if ( )
            throw new NullPointerException();
        if (reverse.containsKey(value))
            throw new IllegalArgumentException();
        reverse.remove(map.get(key));
        reverse.put( );
        return map.put(key, value);
    }

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

    public V remove(K key) {
        V value = map.remove(key);
        reverse.remove(value);
        return value;
    }

    public void putForce(K key, V value) {
        // valueがほかのキーと対応付けられていれば、その対応付けを削除する。
        // valueに対応するキーの存在の有無をチェックする必要はない。
        map.remove( );
        (key, value);
    }
}
```

設問 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 `key == null || value == null`
- イ `key == null && value == null`
- ウ `key != null || value != null`
- エ `key != null && value != null`

b に関する解答群

- ア `key, key`
- イ `key, value`
- ウ `value, key`
- エ `value, value`

c に関する解答群

- ア `key`
- イ `reverse.get(value)`
- ウ `reverse.put(value, key)`
- エ `reverse.remove(value)`

d に関する解答群

- ア `map.put`
- イ `put`
- ウ `putForce`
- エ `reverse.put`

問9 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, 2に答えよ。

〔プログラムの説明〕

2語の中にあるビット列を結合してレジスタに設定する副プログラム COMBINE と、レジスタ中のビット列を分割して2語に格納する副プログラム DIVIDE である。

(1) 図1に COMBINE と DIVIDE のデータ形式と結果を示す。ただし、 $m + n \leq 16$  である。

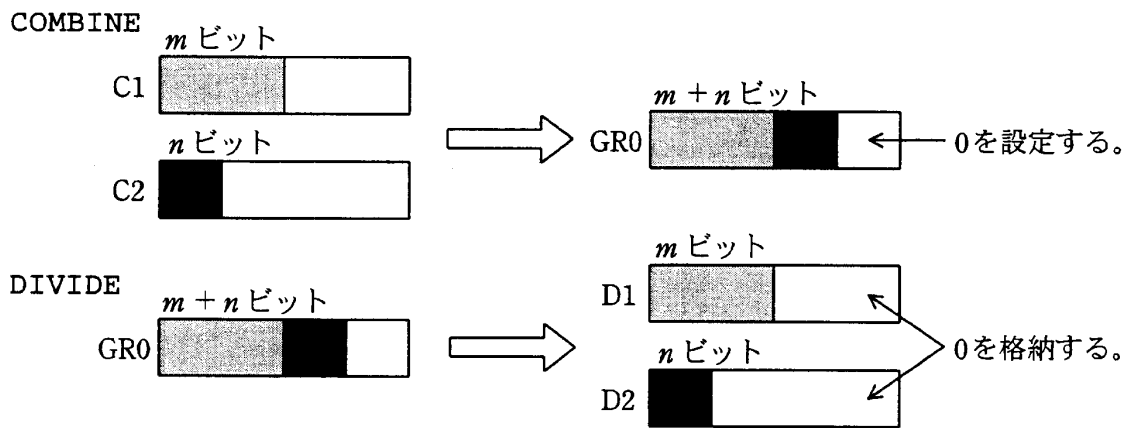


図1 COMBINE と DIVIDE のデータ形式と結果

(2) COMBINE は、GR1~GR4 にそれぞれ次の内容が設定されて主プログラムから呼ばれる。

- GR1 : C1 のアドレス
- GR2 : C1 のビット列の長さ ( $m$ )
- GR3 : C2 のアドレス
- GR4 : C2 のビット列の長さ ( $n$ )

(3) COMBINE は、結合した  $m + n$  ビットを GR0 に左詰めで設定し、残りのビットには0を設定する。

(4) DIVIDE は、GR0~GR4 にそれぞれ次の内容が設定されて主プログラムから呼ばれる。

- GR0 : 分割するビット列
- GR1 : D1 のアドレス
- GR2 : D1 に格納するビット列の長さ ( $m$ )
- GR3 : D2 のアドレス
- GR4 : D2 に格納するビット列の長さ ( $n$ )

- (5) DIVIDE は、D1 には  $m$  ビットを、D2 には  $n$  ビットをそれぞれ左詰めで格納し、残りのビットには 0 を格納する。
- (6) COMBINE 及び DIVIDE 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]

```

COMBINE  START
        RPUSH
        LD   GR0,0,GR1
        LD   GR5,=16
        SUBL GR5,GR2           ; 16 - m
        

|   |
|---|
| a |
|---|

           ; }
        SLL  GR0,0,GR5         ; } 左側 m ビット以外のビットを 0 にする。
;
        LD   GR6,0,GR3
        LD   GR7,=16
        SUBL GR7,GR4           ; 16 - n
        SRL  GR6,0,GR7
        SUBL GR5,GR4           ; 16 - m - n
        

|   |
|---|
| b |
|---|


        OR   GR0,GR6
        RPOP
        RET
        END

```

[プログラム 2]

```

DIVIDE  START
        RPUSH
        LD   GR5,GR0
        LD   GR6,=16
        SUBL GR6,GR2           ; 16 - m
        SRL  GR0,0,GR6
        SLL  GR0,0,GR6
        ST   GR0,0,GR1 ← α
;
        SUBL GR6,GR4           ; 16 - m - n
        

|   |
|---|
| c |
|---|


        LD   GR6,=16
        SUBL GR6,GR4           ; 16 - n
        SLL  GR5,0,GR6
        ST   GR5,0,GR3 ← β
        RPOP
        RET
        END

```



設問1 プログラム1及びプログラム2中の  に入れる正しい答えを、解答群の中から選べ。

解答群

- |   |     |           |   |     |           |
|---|-----|-----------|---|-----|-----------|
| ア | SLL | GR5,0,GR6 | イ | SLL | GR6,0,GR5 |
| ウ | SRL | GR0,0,GR5 | エ | SRL | GR0,0,GR6 |
| オ | SRL | GR5,0,GR6 | カ | SRL | GR6,0,GR5 |

設問2 次の記述中の  に入れる正しい答えを、解答群の中から選べ。

DIVIDE において、データを格納しない部分は元の内容を保持したままとするためには、プログラム2中の  $\alpha$ 、 $\beta$  の部分を次のように変更する必要がある。

図2に変更後のデータ形式と結果を示す。

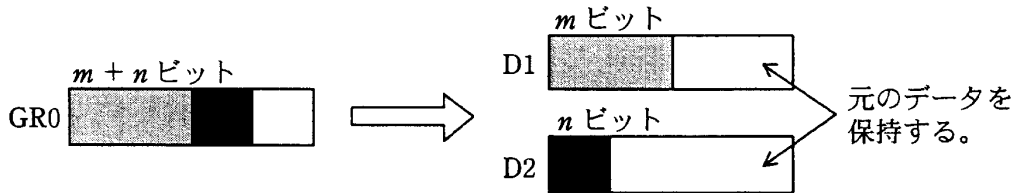


図2 変更後のデータ形式と結果

処置	変更内容
$\alpha$ を置換	LD GR7,=#FFFF SRL GR7,0,GR2 <input type="text"/> d ; 元のデータを取り出す。 OR GR0,GR7 ST GR0,0,GR1
$\beta$ を置換	LD GR7,=#FFFF SRL GR7,0,GR4 AND GR7,0,GR3 <input type="text"/> e ; データを結合する。 ST GR5,0,GR3

解答群

- |   |     |           |   |     |           |   |     |           |
|---|-----|-----------|---|-----|-----------|---|-----|-----------|
| ア | AND | GR6,0,GR1 | イ | AND | GR6,0,GR2 | ウ | AND | GR7,0,GR1 |
| エ | AND | GR7,0,GR2 | オ | OR  | GR5,GR6   | カ | OR  | GR5,GR7   |
| キ | OR  | GR7,GR5   | ク | OR  | GR7,GR6   |   |     |           |

次の問10 から問13 までの 4 問については、この中から 1 問を選択し、答案用紙の選択欄の (選) をマークして解答してください。

なお、2 問以上選択した場合には、はじめの 1 問について採点します。

問 10 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。

[プログラムの説明]

関数 `pattern_match_string` は、対象文字列を先頭から 1 文字ずつ順に調べ、パターン文字列が表現している条件を満足しているかどうかを判定するプログラムである。

(1) 条件文字列は、対象文字列の各文字（対象文字）に対する条件を表現した 1 文字以上の文字列（パターン文字列）を連結したものである。対象文字とパターン文字列の対応関係は、図のとおりである。

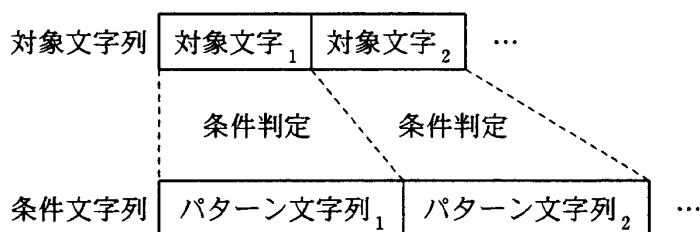


図 対象文字とパターン文字列の対応関係

(2) 関数 `pattern_match_string` の引数は、次のとおりである。

`target` : 対象文字列

`cond` : 条件文字列

(3) 対象文字は、英数字だけである。

(4) パターン文字列に含まれる文字は、次のとおりである。

① 英数字

② “\$”, “[”, “]”, “{”, “}”, “@”

(5) 対象文字列及び条件文字列に誤りはない。

(6) 表の例では、対象文字列はいずれの条件文字列で表現される条件も満足している。

表 対象文字列と条件文字列の例

対象文字列	FE2006
条件文字列	FE2@06
	F\$U2006

[プログラム]

```
#include <string.h>

int pattern_match_string(char *, char *);

static char numeral[] = "0123456789";          /* 数字 */
static char lower[] = "abcdefghijklmnopqrstuvwxyz"; /* 英小文字 */
static char upper[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; /* 英大文字 */

int pattern_match_string(char *target, char *cond){
    int flg = 0, i;

    while(*target != '\0' && *cond != '\0' && flg == 0){
        if(*cond == '$'){                          /* 先頭が'$'の場合 */
            cond++;
            if(*cond == 'N'){
                for(i = 0; numeral[i] != *target
                    && numeral[i] != '\0'; i++){
                    if(numeral[i] != *target) flg = 1;
                }
            }else if(*cond == 'L'){
                for(i = 0; lower[i] != *target
                    && lower[i] != '\0'; i++){
                    if(lower[i] != *target) flg = 1;
                }
            }else if(*cond == 'U'){
                for(i = 0; upper[i] != *target
                    && upper[i] != '\0'; i++){
                    if(upper[i] != *target) flg = 1;
                }
            }else if(*cond != *target){
                flg = 1;
            }
        }else if(*cond == '['){                    /* 先頭が '[' の場合 */
            flg = 1;
            for(cond++; *cond != ']'; cond++)
                if(*target == *cond) flg = 0;
        }else if(*cond == '{'){                  /* 先頭が '{' の場合 */
            for(cond++; *cond != '}'; cond++)
                if(*target == *cond) flg = 1;
        }else if(*cond != '@'){                 /* 先頭が '@' でない場合 */
            if(*target != *cond) flg = 1;
        }
        if(flg == 0){
            target++;
            cond++;
        }
    }
    if(flg != 0){
        return strlen(target);
    }else if(*target == *cond){
        return 0;
    }else{
        return -1;
    }
}
}
```

設問1 パターン文字列に関する次の説明中の  に入れる正しい答えを、  
解答群の中から選べ。

- (1) 対象文字が任意の英数字であることを表現するパターン文字列は、  である。
- (2) 対象文字が数字であることを表現するパターン文字列は、  である。
- (3) 対象文字が“x”，“y”，“z”のいずれかの文字であることを表現するパターン文字列は、  である。
- (4) 対象文字が“x”，“y”，“z”のいずれの文字でもないことを表現するパターン文字列は、  である。

a, bに関する解答群

- |         |         |         |          |
|---------|---------|---------|----------|
| ア “e”   | イ “\$A” | ウ “\$L” | エ “\$LU” |
| オ “\$N” | カ “\$U” | キ “\$e” |          |

c, dに関する解答群

- |               |               |             |
|---------------|---------------|-------------|
| ア “[XYZ]”     | イ “{XYZ}”     | ウ “\$XYZ\$” |
| エ “\$[XYZ\$]” | オ “\${XYZ\$}” |             |

設問2 関数 `pattern_match_string` の返却値に関する次の説明中の  に  
入れる正しい答えを、解答群の中から選べ。

- (1) 対象文字列のすべての文字が条件文字列で与えた条件を満足している場合、  
返却値として  を返す。
- (2) 対象文字列の途中の文字に条件文字列で与えた条件を満足しないものがあった  
場合、そこで判定を終了し、返却値として  を返す。
- (3) 対象文字列と条件文字列のいずれかが途中で終了してしまった場合（'\0'が  
見つかった場合）、そこで判定を終了し、返却値として `-1` を返す。

解答群

- ア 0
- イ `-1`
- ウ 対象文字列の文字数
- エ 検査した対象文字数
- オ 検査して条件を満足していた対象文字数
- カ 検査していない対象文字数 + 1
- キ 検査していない対象文字数

問 11 次の COBOL プログラムの説明及びプログラムを読んで、設問 1～3 に答えよ。

〔プログラムの説明〕

製品出荷指示ファイルを読み込み、在庫を調べ、部品展開表を使って製造に必要な部品の個数を計算し、部品注文ファイルを作成するプログラムである。製品は幾つかの部品から構成され、製造工程で組み立てられる。図に部品展開表の例を示す。括弧内の数字は、親の製品を 1 個製造するために必要な子の部品の個数（以下、所要個数という）である。この例では、製品 S1 を 1 個製造するために、部品 B1 を 2 個、部品 B2 を 2 個、部品 B3 を 1 個必要とすることを示す。

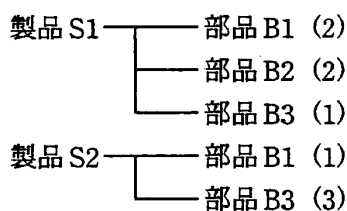


図 部品展開表の例

例えば、製品 S2 の在庫が 75 個、部品 B1 の在庫が 30 個、部品 B3 の在庫が 60 個あるとき、製品 S2 を 100 個出荷するためには、製品 S2 の在庫 75 個では不足であり、 $25 (= 100 - 75)$  個を新たに製造する必要がある。そのために必要な 25 個の部品 B1 は、在庫から割り当てることができる。また、部品 B3 は  $75 (= 25 \times 3)$  個必要であり、在庫の 60 個では不足なので、 $15 (= 75 - 60)$  個を注文する必要がある。

(1) 在庫数のうち出荷又は製造のために割り当てた個数を、割当て済み数という。

また、在庫数から割当て済み数を引いた個数を、有効在庫数という。

(2) 製品出荷指示ファイルは順ファイルで、そのレコード様式は次のとおりである。

出荷番号 5 けた	製品コード 8 けた	出荷数 10 けた
--------------	---------------	--------------

(3) 在庫ファイルは索引ファイルで、そのレコード様式は次のとおりである。

資材コード(製品コード 又は部品コード) 8 けた	在庫数 10 けた	割当て済み数 10 けた
---------------------------------	--------------	-----------------

資材コード（製品コード又は部品コード）をキーとする。製品コードは先頭 1 けたが“S”，部品コードは先頭 1 けたが“B”である。

(4) 部品展開ファイルは索引ファイルで、そのレコード様式は次のとおりである。

部品展開キー		所要個数 5 けた
製品コード 8 けた	部品コード 8 けた	

- ① 製品コードと部品コードの組合せをキーとする。
- ② 所要個数は、製品コードと部品コードの組合せごとに設定される。

先の部品展開表の例の図に対応する部品展開レコードの例を、表に示す。

表 図に対応する部品展開レコードの例

S1	B1	2
S1	B2	2
S1	B3	1
S2	B1	1
S2	B3	3

(5) 部品注文ファイルは順ファイルで、そのレコード様式は次のとおりである。

出荷番号 5 けた	部品コード 8 けた	発注数 10 けた
--------------	---------------	--------------

製品出荷指示ファイルの1レコードごとに必要な部品注文レコードが出力される。

(6) すべてのデータに誤りはないものとする。

[プログラム]

(行番号)

```

1 DATA DIVISION.
2 FILE SECTION.
3 FD SHUKKA-F.
4 01 SHUKKA-R.
5     05 S-SHUKKA-NO PIC 9(5).
6     05 S-SEIHIN-CD PIC X(8).
7     05 S-SHUKKA-SU PIC 9(10).
8 FD ZAIKO-F.
9 01 ZAIKO-R.
10    05 Z-SHIZAI-CD PIC X(8).
11    05 Z-ZAIKO-SU PIC 9(10).
12    05 Z-WARIATE-SU PIC 9(10).
```

```

13 FD  BUHIN-F.
14 01  BUHIN-R.
15     05  B-BUHIN-KEY.
16         10  B-SEIHIN-CD  PIC X(8).
17         10  B-BUHIN-CD  PIC X(8).
18     05  B-SHOYO-KOSU  PIC 9(5).
19 FD  CHUMON-F.
20 01  CHUMON-R.
21     05  C-SHUKKA-NO  PIC 9(5).
22     05  C-BUHIN-CD  PIC X(8).
23     05  C-CHUMON-SU  PIC 9(10).
24 WORKING-STORAGE SECTION.
25 01  W-YUKO-ZSU  PIC S9(10).
26 01  W-HITSUYO-SU  PIC 9(10).
27 01  W-SEIHIN-SU  PIC 9(10).
28 01  W-SHUKKA-EOF  PIC 9.
29 PROCEDURE DIVISION.
30 MAIN-CTL.
31     OPEN INPUT SHUKKA-F BUHIN-F,
32           I-O ZAIKO-F, OUTPUT CHUMON-F.
33     MOVE 0 TO W-SHUKKA-EOF.
34     PERFORM UNTIL W-SHUKKA-EOF = 1
35         READ SHUKKA-F
36             AT END      MOVE 1 TO W-SHUKKA-EOF
37             NOT AT END  PERFORM MRP-PROC
38     END-READ
39     END-PERFORM.
40     CLOSE SHUKKA-F BUHIN-F ZAIKO-F CHUMON-F.
41     STOP RUN.
42 *
43 MRP-PROC.
44     MOVE S-SEIHIN-CD TO Z-SHIZAI-CD.
45     MOVE S-SHUKKA-SU TO W-HITSUYO-SU.
46     a.
47     IF W-HITSUYO-SU > 0 THEN
48         MOVE W-HITSUYO-SU TO W-SEIHIN-SU
49         MOVE LOW-VALUE TO B-BUHIN-KEY
50         MOVE S-SEIHIN-CD TO B-SEIHIN-CD
51     b
52     PERFORM UNTIL B-SEIHIN-CD NOT = S-SEIHIN-CD
53         MOVE B-BUHIN-CD TO Z-SHIZAI-CD
54         COMPUTE W-HITSUYO-SU =
55             W-SEIHIN-SU * B-SHOYO-KOSU
56         PERFORM CHECK-ZAIKO
57         PERFORM READ-BUHIN-NEXT
58     END-PERFORM
59     END-IF.
60 *

```



```

61 CHECK-ZAIKO.
62     READ ZAIKO-F
63         INVALID KEY DISPLAY "ZAIKO-INV ERROR"
64             STOP RUN
65     END-READ.
66     COMPUTE W-YUKO-ZSU = Z-ZAIKO-SU - Z-WARIATE-SU.
67     IF W-YUKO-ZSU <= 0 THEN
68         PERFORM CHUMON-PROC
69     ELSE
70         IF W-HITSUYO-SU > W-YUKO-ZSU THEN
71             ADD      W-YUKO-ZSU    TO    Z-WARIATE-SU
72             PERFORM  REWRITE-ZAIKO
73             SUBTRACT W-YUKO-ZSU    FROM W-HITSUYO-SU
74             PERFORM  CHUMON-PROC
75         ELSE
76             ADD      W-HITSUYO-SU TO    Z-WARIATE-SU
77             MOVE     0                TO    W-HITSUYO-SU
78             PERFORM  REWRITE-ZAIKO
79         END-IF
80     END-IF.
81 *
82 START-BUHIN.
83     START BUHIN-F C
84         INVALID KEY MOVE HIGH-VALUE TO BUHIN-R
85         NOT INVALID KEY PERFORM READ-BUHIN-NEXT
86     END-START.
87 *
88 READ-BUHIN-NEXT.
89     READ BUHIN-F NEXT
90         AT END MOVE HIGH-VALUE TO BUHIN-R
91     END-READ.
92 *
93 CHUMON-PROC.
94     IF Z-SHIZAI-CD(1:1) = "B" THEN
95         MOVE S-SHUKKA-NO TO C-SHUKKA-NO
96         MOVE Z-SHIZAI-CD TO C-BUHIN-CD
97         MOVE W-HITSUYO-SU TO C-CHUMON-SU
98         WRITE CHUMON-R
99         MOVE 0 TO W-HITSUYO-SU
100    END-IF.
101 *
102 REWRITE-ZAIKO.
103     REWRITE ZAIKO-R
104         INVALID KEY DISPLAY "ZAIKO-INV ERROR"
105             STOP RUN
106     END-REWRITE.

```

COBOL

設問 1. プログラム中の  に入れる正しい答えを、解答群の中から選べ。

a, b に関する解答群

- |   |                     |   |                         |
|---|---------------------|---|-------------------------|
| ア | PERFORM CHECK-ZAIKO | イ | PERFORM CHUMON-PROC     |
| ウ | PERFORM MRP-PROC    | エ | PERFORM READ-BUHIN-NEXT |
| オ | PERFORM START-BUHIN |   |                         |

c に関する解答群

- |   |                   |   |                       |
|---|-------------------|---|-----------------------|
| ア | KEY > B-BUHIN-KEY | イ | KEY < B-BUHIN-KEY     |
| ウ | KEY = B-BUHIN-KEY | エ | KEY NOT = B-BUHIN-KEY |

設問 2 このプログラムの環境部のファイル管理記述段落で、BUHIN-F の呼出し法の指定として正しくないものを、解答群の中から選べ。

解答群

- |   |                           |   |                       |
|---|---------------------------|---|-----------------------|
| ア | ACCESS MODE IS DYNAMIC    | イ | ACCESS MODE IS RANDOM |
| ウ | ACCESS MODE IS SEQUENTIAL |   |                       |

設問 3 部品注文ファイルを作成し発注してから、部品が入庫され製品が組み立てられるまでには、出荷遅延が発生することが考えられる。また、予定外の出荷指示に対して、すぐに応じられないという問題も考えられる。これをある程度防ぐため、有効在庫数がある一定の個数（余裕在庫数）以下となった場合に部品注文ファイルを作成するように、在庫ファイルのレコード様式及びプログラムを変更する。

d  に入れる正しい答えを、解答群の中から選べ。

処置	変更内容
行番号 12 と 13 の間に挿入	05 Z-YOYUU-SU PIC 9(10).
行番号 66 を置換	COMPUTE W-YUKO-ZSU = <input type="text"/> d <input type="text"/> .

解答群

- |   |  |
|---|--|
| ア | Z-ZAIKO-SU + Z-WARIATE-SU + Z-YOYUU-SU |
| イ | Z-ZAIKO-SU + Z-WARIATE-SU - Z-YOYUU-SU |
| ウ | Z-ZAIKO-SU - Z-WARIATE-SU + Z-YOYUU-SU |
| エ | Z-ZAIKO-SU - Z-WARIATE-SU - Z-YOYUU-SU |

問 12 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]

図書の貸出しと返却の処理をするプログラムである。図書には書籍と雑誌があり、利用者が借りることのできる図書の冊数には上限がある。

- (1) 抽象クラス `Book` は、図書に共通の属性と処理を定義する。属性 `name` は図書の名前であり、属性 `id` は図書に一意に付けられた識別子である。
- (2) クラス `RegularBook` は書籍を表す。
- (3) クラス `Magazine` は雑誌を表す。属性 `issueNo` は、雑誌の創刊号からの号数で、古い号から昇順の値をもつ。各雑誌について、`issueNo` が最大のものが最新号である。
- (4) クラス `User` は図書の利用者を表す。属性 `name` は利用者の名前であり、属性 `id` は利用者一意に付けられた識別子である。
- (5) クラス `Library` は、貸出しと返却の処理をする。属性 `availables` は貸出し可能な図書の集合を表し、属性 `checkedOut` は貸出し中の図書の集合を表す。属性 `limit` は利用者が借りることのできる図書の冊数の上限を与える。メソッド `checkoutBook` は、貸出し処理を実行する。指定された図書が既に貸し出されている場合と利用者が既に貸出し冊数の上限まで図書を借りている場合、例外を投げる。メソッド `returnBook` は返却処理を実行する。

メソッド `main` はテスト用のメインプログラムである。実行結果を図に示す。

```
Taro is checking out Magazine: JITEC News, No.3912 (M001)
Taro is checking out Magazine: JITEC News, No.4001 (M002)
Taro is checking out Book: Java Programming (P001)
Taro is checking out Book: Ruby Programming (P003)
=> failed: exceeding checkout limit
Taro returned Magazine: JITEC News, No.3912 (M001)
Hana is checking out Magazine: JITEC News, No.3912 (M001)
Hana is checking out Book: Java Programming (P001)
=> failed: unavailable
```

図 実行結果

[プログラム]

(行番号)

```
1 import java.util.*;
2
3 abstract class Book {
4     String name;
5     String id;
6     Book(String name, String id) {
7         this.name = name;
8         this.id = id;
9     }
10    public boolean equals(Object object) {
11        return (object instanceof Book)
12            && id.equals(((Book) object).id);
13    }
14    public int hashCode() {
15        return id.hashCode();
16    }
17 }
18 class RegularBook extends Book {
19     RegularBook(String name, String id) {
20         a;
21     }
22     public String toString() {
23         return "Book: " + name + " (" + id + ")";
24     }
25 }
26
27 class Magazine extends Book {
28     int issueNo;
29     Magazine(String name, int issueNo, String id) {
30         a;
31         this.issueNo = issueNo;
32     }
33     public String toString() {
34         return "Magazine: " + name + ", No." + issueNo
35             + " (" + id + ")";
36     }
37 }
38 class User {
39     String name;
40     String id;
41     User(String name, String id) {
42         this.name = name;
43         this.id = id;
44     }
45     public String toString() {
46         return "User: " + name + " (" + id + ")";
47     }
48 }
```

```

49 class Library {
50     Set<Book> availables = new HashSet<Book>();
51     Map<Book, User> checkedOut = new HashMap<Book, User>();
52     int limit;
53     Library(Book[] books, int limit) {
54         for (Book book : books) register(book);
55         this.limit = limit;
56     }
57     void register(Book book) {
58         availables.add(book);
59     }
60     void checkoutBook(User user, Book book) throws Exception {
61         if (! availables.contains(book))
62             b ("unavailable");
63         int count = 0;
64         for (Map.Entry<Book, User> entry :
65             checkedOut.entrySet())
66             if (entry.getValue().equals(user)) count++;
67         if (count >= limit)
68             b ("exceeding checkout limit");
69         availables.remove(book);
70         checkedOut.put(book, user);
71     }
72     void returnBook(Book book) {
73         for (Map.Entry<Book, User> entry :
74             checkedOut.entrySet())
75             if (entry.getKey().equals(book)) {
76                 c (book);
77                 availables.add(book);
78                 return;
79             }
80     }
81     public static void main(String[] args) {
82         Book
83         java = new RegularBook("Java Programming", "P001"),
84         perl = new RegularBook("Perl Programming", "P002"),
85         ruby = new RegularBook("Ruby Programming", "P003"),
86         jit39_12 = new Magazine("JITEC News", 3912, "M001"),
87         jit40_01 = new Magazine("JITEC News", 4001, "M002");
88         Book[] books = {java, perl, ruby, jit39_12, jit40_01};
89         Library library = new Library(books, 3);
90         User taro = new User("Taro", "ID-01");
91         User hana = new User("Hana", "ID-02");
92         Book[] books1 = {jit39_12, jit40_01, java, ruby};
93         for (Book book : books1)
94             try {
95                 System.out.print("Taro is checking out " + book);
96                 library.checkoutBook(taro, book);
97                 System.out.println();
98             }
99     }

```

```

97         catch (Exception e) {
98             System.out.println("\n" + "    => failed: "
                                   + e.getMessage());
99         }
100     library.returnBook(jit39_12);
101     System.out.println("Taro returned " + jit39_12);
102     Book[] books2 = {jit39_12, java};
103     for (Book book : books2)
104         try {
105             System.out.print("Hana is checking out " + book);
106             library.checkoutBook(hana, book);
107             System.out.println();
108         }
109         catch (Exception e) {
110             System.out.println("\n" + "    => failed: "
                                   + e.getMessage());
111         }
112     }
113 }

```

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 new Book()
- イ new Book(name, id)
- ウ super()
- エ super(name, id)
- オ this.name = name; this.id = id

bに関する解答群

- |                      |                        |
|----------------------|------------------------|
| ア return Exception   | イ return new Exception |
| ウ System.err.println | エ System.out.println   |
| オ throw Exception    | カ throw new Exception  |

cに関する解答群

- |                  |                     |
|------------------|---------------------|
| ア checkedOut.add | イ checkedOut.remove |
| ウ entry.add      | エ entry.remove      |

設問2 雑誌について、最新号の貸出しを禁止する処理を追加したい。このため、次のようにプログラムを変更する。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。ただし、各雑誌の各号は1冊ずつしかなく、雑誌の登録は、必ずしも古い号から行われるとは限らない。また、  b  には設問1の正しい答えが入っているものとする。

処置	追加内容
行番号 50 の直後に追加	<code>Set&lt;Magazine&gt; latestMagazines = new HashSet&lt;Magazine&gt;();</code>
行番号 58 の直後に追加	<code>if (book instanceof Magazine) updateLatestMagazineList((Magazine) book);</code>
行番号 62 の直後に追加	<code>if (book instanceof Magazine) { Magazine magazine = (Magazine) book; if (latestMagazines.contains(magazine)) <input type="text"/> b <input type="text"/> ("latest issue"); }</code>
行番号 78 の直後に追加	<code>void updateLatestMagazineList(Magazine magazine) { for (Magazine magazine2 : latestMagazines) if (magazine2.name.equals(magazine.name)) { if (magazine.issueNo <input type="text"/> d <input type="text"/> magazine2.issueNo) { return; } else { latestMagazines.remove(magazine2); break; } } <input type="text"/> e <input type="text"/> ; }</code>

dに関する解答群

ア <                      イ >                      ウ !=                      エ ==

eに関する解答群

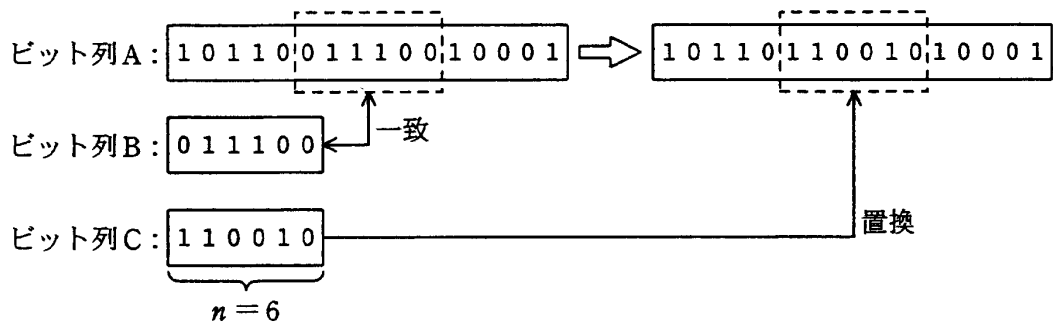
- ア latestMagazines.add(magazine)
- イ register((Magazine) book)
- ウ return
- エ returnBook(magazine)

問 13 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, 2 に答えよ。

[プログラムの説明]

- (1) 副プログラム BREP は、16 ビットからなるビット列 A の中に、 $n$  ビットのビット列 B と一致する部分ビット列があれば、それを  $n$  ビットのビット列 C に置き換えるプログラムである。

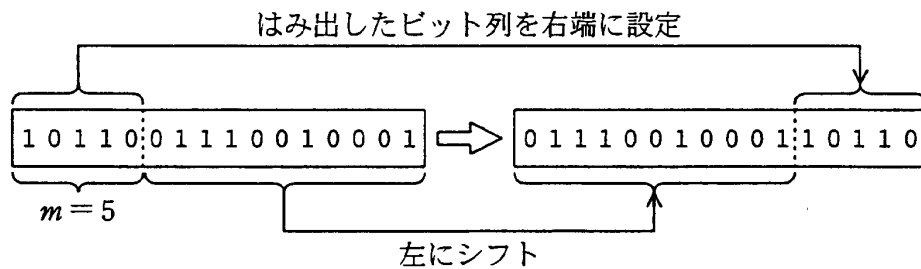
例：



- ① GR1～GR4 にはそれぞれ次の内容を設定して、主プログラムから渡される。  
 GR1：ビット列 A を格納している語のアドレス  
 GR2：ビット列 B (左詰めで設定され、残りのビットには 0 が設定されている。)  
 GR3：ビット列 C (左詰めで設定され、残りのビットには 0 が設定されている。)  
 GR4： $n$  ( $1 \leq n \leq 16$ )
  - ② ビット列 A の中にビット列 B に一致する部分ビット列が複数個あれば、それらのすべてをビット列 C に置き換える。
  - ③ ビット列 A の左端から検索する。ビット列 B に一致する部分ビット列をビット列 C に置き換えた場合、その部分ビット列の右隣のビットから検索を続ける。
  - ④ 副プログラムから戻るとき、汎用レジスタ GR1～GR7 の内容は元に戻す。
- (2) 副プログラム ROTSL は、16 ビットからなるビット列を  $m$  ビットだけ左に循環シフトするプログラムである。



例：



① GR0 と GR1 にはそれぞれ次の内容を設定して、主プログラムから渡される。

GR0 :  $m$  ( $0 \leq m \leq 16$ )

GR1 : ビット列を格納している語のアドレス

② 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]

```

BREP  START
      RPUSH
      LD   GR5,=16           ; 未検査ビット数の初期化
      LD   GR6,=#8000       ; } マスクの作成
      SRA  GR6,-1,GR4       ; }
LOOP  LD   GR7,0,GR1       ; GR7←ビット列 A
      a           ; 左端 n ビット以外のビットを 0 にする。
      CPL  GR7,GR2         ; 左端 n ビットとビット列 B を比較
      JZE  MATCH          ; 一致
      LD   GR0,=1          ; 左循環シフトするビット数
      JUMP CONT
MATCH LD   GR7,0,GR1       ; GR7←ビット列 A
      SLL  GR7,0,GR4       ; } 左端 n ビットを 0 にする。
      SRL  GR7,0,GR4       ; }
      b           ; ビット列 C を左端 n ビットに設定
      ST   GR7,0,GR1       ; 元の領域に戻す。
      LD   GR0,GR4         ; 左循環シフトするビット数
CONT  CALL ROTSL          ; ビット列 A を左循環シフト
      SUBA GR5,GR0         ; 未検査ビット数の更新
      CPA  GR5,GR4         ; 未検査ビット数と n を比較
      c           ; n 未満であれば終了処理へ
      JUMP LOOP
FIN   LD   GR0,GR5         ; 未検査ビット数
      CALL ROTSL          ; ビット位置を元に戻す。
      RPOP
      RET
;

```

アセンブラ

```

ROTSL  RPUSH
LD      GR3,GR0          ; GR3← m
LD      GR4,=16
SUBA    GR4,GR3          ; GR4←(16-m)
LD      GR5,0,GR1
LD      GR6,GR5
SLL     GR5,0,GR3


d

 ; はみ出すビット列
OR      GR5,GR6          ; はみ出すビット列を右端に設定
ST      GR5,0,GR1        ; 元の領域に戻す。
RPOP
RET
END

```

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 AND GR7,GR3	イ AND GR7,GR6
ウ OR GR7,GR3	エ OR GR7,GR6

c に関する解答群

ア JMI FIN	イ JNZ FIN
ウ JPL FIN	エ JZE FIN

d に関する解答群

ア SLL GR6,0,GR3	イ SLL GR6,0,GR4
ウ SRL GR6,0,GR3	エ SRL GR6,0,GR4

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。

主プログラムから渡されたビット列 A, B, C 及び  $n$  が次のとおりであったとき、副プログラム BREP は副プログラム ROTSL を  回呼び出す。

ビット列 A: 0001110100111001

ビット列 B: 0111

ビット列 C: 1001

$n$ : 4

解答群

ア 2

イ 6

ウ 8

エ 12

オ 16

## ■ 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。

入れ子クラス

**public static interface Map.Entry<K, V>**

マップに登録されている型 K のキーと型 V の値との対 (エントリ) を表す。

メソッド

**public boolean containsKey(Object key)**

指定されたキーに値が対応付けられていれば、true を返す。

引数: key — キー

戻り値: 指定されたキーに値が対応付けられていれば true  
それ以外は false

**public Set<Map.Entry<K, V>> entrySet()**

マップに登録されているキーと値の対の集合を返す。

引数: なし

戻り値: マップに登録されているキーと値の対の集合

**public V get(Object key)**

指定されたキーに対応付けられた値を返す。

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

**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

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。キー及び値は, null でもよい。

コンストラクタ

**public** HashMap()

空の HashMap を作る。

メソッド

**public boolean** containsKey(Object key)

インタフェース Map のメソッド containsKey と同じ

**public Set**<Map.Entry<K, V>> entrySet()

インタフェース Map のメソッド entrySet と同じ

**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

java.util

**public static interface** Map.Entry<K, V>

マップに登録されているキーと値の対 (エントリ) を表す。

メソッド

**public K** getKey()

キーを返す。

引数: なし

戻り値: 型 K のキー

**public V** getValue()

値を返す。

引数: なし

戻り値: 型 V の値

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 — 集合に追加される要素

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 contains(Object o)**

指定された要素が集合に含まれていれば、true を返す。

引数： o — 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 remove(Object o)**

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： o — 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

java.util

**public class HashSet<E>**

インタフェース Set のハッシュを用いた実装である。

コンストラクタ

**public HashSet()**

空の HashSet を作成する。

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 contains(Object o)**

インタフェース Set のメソッド contains と同じ

**public boolean remove(Object o)**

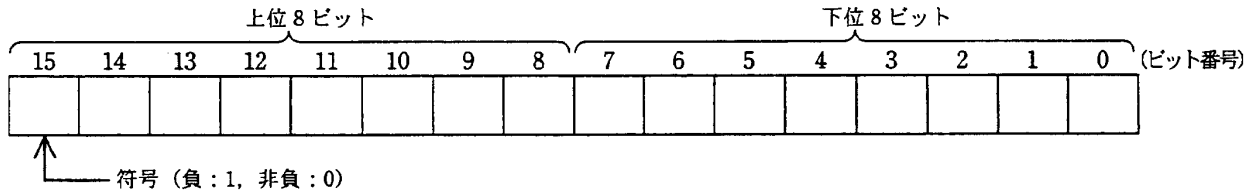
インタフェース Set のメソッド remove と同じ

## ■アセンブラ言語の仕様

### 1. システム COMET II の仕様

#### 1.1 ハードウェアの仕様

(1) 1語は16ビットで、そのビット構成は、次のとおりである。



(2) 主記憶の容量は65536語で、そのアドレスは0～65535番地である。

(3) 数値は、16ビットの2進数で表現する。負数は、2の補数で表現する。

(4) 制御方式は逐次制御で、命令語は1語長又は2語長である。

(5) レジスタとして、GR (16ビット)、SP (16ビット)、PR (16ビット)、FR (3ビット) の4種類がある。

GR (汎用レジスタ, General Register) は、GR0～GR7の8個があり、算術、論理、比較、シフトなどの演算に用いる。このうち、GR1～GR7のレジスタは、指標レジスタ (index register) としてアドレスの修飾にも用いる。

SP (スタックポインタ, Stack Pointer) は、スタックの最上段のアドレスを保持している。

PR (プログラムレジスタ, Program Register) は、次に実行すべき命令語の先頭アドレスを保持している。

FR (フラグレジスタ, Flag Register) は、OF (Overflow Flag)、SF (Sign Flag)、ZF (Zero Flag) と呼ぶ3個のビットからなり、演算命令などの実行によって次の値が設定される。これらの値は、条件付き分岐命令で参照される。

OF	算術演算命令の場合は、演算結果が-32768～32767に収まらなくなったとき1になり、それ以外の場合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、それ以外の場合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、それ以外の場合0になる。
ZF	演算結果が零 (全部のビットが0) のとき1、それ以外の場合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	$r1, r2$	$r1 \leftarrow (r2)$	○*1
		$r, \text{adr} [, x]$	$r \leftarrow (\text{実効アドレス})$	
ストア STore	ST	$r, \text{adr} [, x]$	実効アドレス $\leftarrow (r)$	—
ロードアドレス Load Address	LAD	$r, \text{adr} [, x]$	$r \leftarrow \text{実効アドレス}$	

(2) 算術, 論理演算命令

算術加算 ADD Arithmetic	ADDA	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) + (r2)$ $r \leftarrow (r) + (\text{実効アドレス})$	○
論理加算 ADD Logical	ADDL	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) +_L (r2)$ $r \leftarrow (r) +_L (\text{実効アドレス})$	
算術減算 SUBtract Arithmetic	SUBA	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) - (r2)$ $r \leftarrow (r) - (\text{実効アドレス})$	
論理減算 SUBtract Logical	SUBL	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) -_L (r2)$ $r \leftarrow (r) -_L (\text{実効アドレス})$	
論理積 AND	AND	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) \text{ AND } (r2)$ $r \leftarrow (r) \text{ AND } (\text{実効アドレス})$	
論理和 OR	OR	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) \text{ OR } (r2)$ $r \leftarrow (r) \text{ OR } (\text{実効アドレス})$	
排他的論理和 eXclusive OR	XOR	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) \text{ XOR } (r2)$ $r \leftarrow (r) \text{ XOR } (\text{実効アドレス})$	

(3) 比較演算命令

算術比較 ComPare Arithmetic	CPA	$r1, r2$ $r, \text{adr} [, x]$	<p>(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。</p> <table border="1"> <thead> <tr> <th rowspan="2">比較結果</th> <th colspan="2">FR の値</th> </tr> <tr> <th>SF</th> <th>ZF</th> </tr> </thead> <tbody> <tr> <td>(r1) &gt; (r2)</td> <td>0</td> <td>0</td> </tr> <tr> <td>(r) &gt; (実効アドレス)</td> <td>0</td> <td>0</td> </tr> <tr> <td>(r1) = (r2)</td> <td>0</td> <td>1</td> </tr> <tr> <td>(r) = (実効アドレス)</td> <td>0</td> <td>1</td> </tr> <tr> <td>(r1) &lt; (r2)</td> <td>1</td> <td>0</td> </tr> <tr> <td>(r) &lt; (実効アドレス)</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	比較結果	FR の値		SF	ZF	(r1) > (r2)	0	0	(r) > (実効アドレス)	0	0	(r1) = (r2)	0	1	(r) = (実効アドレス)	0	1	(r1) < (r2)	1	0	(r) < (実効アドレス)	1	0	○*1
比較結果	FR の値																										
	SF	ZF																									
(r1) > (r2)	0	0																									
(r) > (実効アドレス)	0	0																									
(r1) = (r2)	0	1																									
(r) = (実効アドレス)	0	1																									
(r1) < (r2)	1	0																									
(r) < (実効アドレス)	1	0																									
論理比較 ComPare Logical	CPL	$r1, r2$ $r, \text{adr} [, x]$																									

(4) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	$r, \text{adr} [, x]$	<p>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。</p> <p>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。</p>	○*2
算術右シフト Shift Right Arithmetic	SRA	$r, \text{adr} [, x]$		
論理左シフト Shift Left Logical	SLL	$r, \text{adr} [, x]$		
論理右シフト Shift Right Logical	SRL	$r, \text{adr} [, x]$		

(5) 分岐命令

正分岐 Jump on Plus	JPL	$\text{adr} [, x]$	<p>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。</p> <table border="1"> <thead> <tr> <th rowspan="2">命令</th> <th colspan="3">分岐するときの FR の値</th> </tr> <tr> <th>OF</th> <th>SF</th> <th>ZF</th> </tr> </thead> <tbody> <tr> <td>JPL</td> <td></td> <td>0</td> <td>0</td> </tr> <tr> <td>JMI</td> <td></td> <td>1</td> <td></td> </tr> <tr> <td>JNZ</td> <td></td> <td></td> <td>0</td> </tr> <tr> <td>JZE</td> <td></td> <td></td> <td>1</td> </tr> <tr> <td>JOV</td> <td>1</td> <td></td> <td></td> </tr> </tbody> </table>	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			-
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																														
負分岐 Jump on Minus	JMI	$\text{adr} [, x]$																													
非零分岐 Jump on Non Zero	JNZ	$\text{adr} [, x]$																													
零分岐 Jump on Zero	JZE	$\text{adr} [, x]$																													
オーバーフロー分岐 Jump on Overflow	JOV	$\text{adr} [, x]$																													
無条件分岐 unconditional JUMP	JUMP	$\text{adr} [, x]$	無条件に実効アドレスに分岐する。																												



(6) スタック操作命令

プッシュ PUSH	PUSH adr [,x]	SP ← (SP) - <sub>L</sub> 1, (SP) ← 実効アドレス	—
ポップ POP	POP r	r ← ( (SP) ), SP ← (SP) + <sub>L</sub> 1	

(7) コール, リターン命令

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

(8) その他

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

- (注) r, r1, r2      いずれも GR を示す。指定できる GR は GR0 ~ GR7  
 adr                アドレスを示す。指定できる値の範囲は 0 ~ 65535  
 x                   指標レジスタとして用いる GR を示す。指定できる GR は GR1 ~ GR7  
 [     ]            [     ] 内の指定は省略できることを示す。  
 (     )            (     ) 内のレジスタ又はアドレスに格納されている内容を示す。  
 実効アドレス      adr と x の内容との論理加算値又はその値が示す番地  
 ←                 演算結果を、左辺のレジスタ又はアドレスに格納することを示す。  
 +<sub>L</sub>, -<sub>L</sub>            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 CASL II の仕様

### 2.1 言語の仕様

- (1) CASL II は、COMET II 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 (COMET II 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 ( $\geq 0$ ) 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 $-32768 \sim 32767$ 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 ( $0000 \leq h \leq FFFF$ )。
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 ( $\geq 0$ ) 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 ( $\geq 0$ ) 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) 

R PUSH	
--------	--

R PUSH 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) 

R POP	
-------	--

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

## 2.5 機械語命令

機械語命令のオペランドは、次の形式で記述する。

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

## 2.6 その他

- (1) アセンブラによって生成される命令語や領域の相対位置は、アセンブラ言語での記述順序とする。ただし、リテラルから生成される DC 命令は、END 命令の直前にまとめて配置される。
- (2) 生成された命令語、領域は、主記憶上で連続した領域を占める。

## 3. プログラム実行の手引

### 3.1 OS

プログラムの実行に関して、次の取決めがある。

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

[ メモ用紙 ]

[ メモ用紙 ]

[ メモ用紙 ]

- 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
--------	---------------

- 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。
- 問題冊子の余白などは、適宜利用して構いません。
- Java プログラムで使用する API の説明及びアセンブラ言語の仕様は、この冊子の末尾を参照してください。
- 電卓は、使用できません。
- 試験終了後、この問題冊子は持ち帰ることができます。
- 答案用紙は、白紙であっても提出してください。
- 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。

試験問題に記載されている会社名又は製品名は、それぞれ各社の商標又は登録商標です。

なお、試験問題では、® 及び ™ を明記していません。