

平成 17 年度 秋期

基本情報技術者 午後 問題

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も続きます。問題冊子を裏返して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及びマークは、試験開始の合図があってから始めてください。
4. 試験時間は、次の表のとおりです。

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

途中で退出する場合には、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出してください。

退出可能時間	13:40 ~ 15:20
--------	---------------

5. 問題は、次の表に従って解答してください。

問題番号	問 1 ~ 問 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

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。


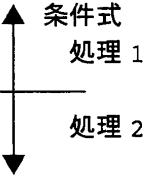
6. 問題に関する質問にはお答えできません。文意どおり解釈してください。
7. 問題冊子の余白などは、適宜利用して構いません。
8. アセンブラ言語の仕様は、この冊子の末尾を参照してください。
9. 電卓は、使用できません。

注意事項は問題冊子の裏表紙に続きます。
こちら側から裏返して、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。
 条件式 処理	単岐選択処理を示す。 条件式が真のときは処理を実行する。
 条件式 処理 1 処理 2	双岐選択処理を示す。 条件式が真のときは処理 1 を実行し，偽のときは処理 2 を実行する。
■ 条件式 処理 ■	前判定繰返し処理を示す。 条件式が真の間，処理を繰り返し実行する。
■ 処理 ■ 条件式	後判定繰返し処理を示す。 処理を実行し，条件式が真の間，処理を繰り返し実行する。
■ 変数：初期値，条件式，増分 処理 ■	繰返し処理を示す。 開始時点で変数に初期値（定数又は式で与えられる）が格納され，条件式が真の間，処理を繰り返す。また，繰り返すごとに，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。

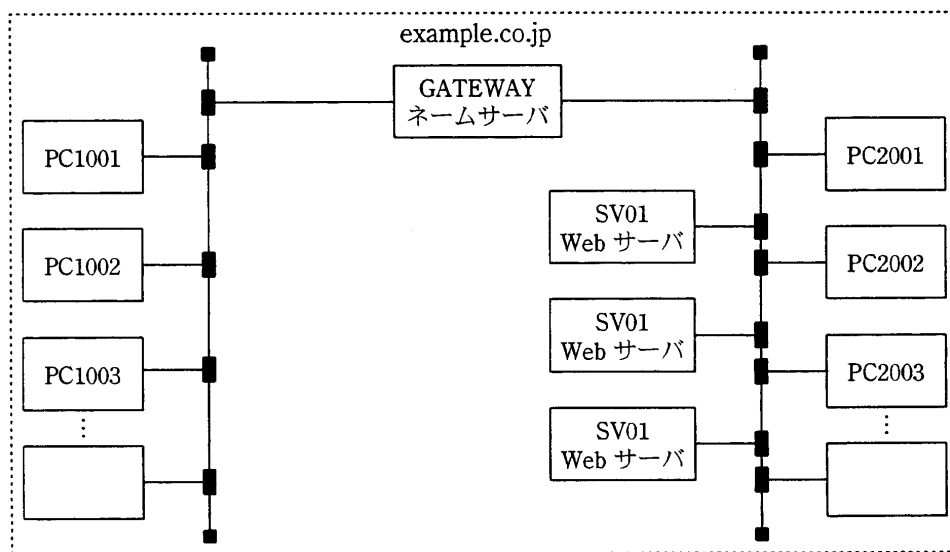
〔論理型の定数〕

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 DNS (Domain Name System) に関する次の記述を読んで、設問 1, 2 に答えよ。

ある企業では、社内のネットワーク上の各コンピュータを、ドメイン名を用いて管理している。この企業のネットワーク構成を次に示す。



各コンピュータのドメイン名と IP アドレスの対応は、表のとおりである。

表 ドメイン名と IP アドレスの対応

ドメイン名	IPアドレス
PC1001.example.co.jp	172.16.0.1
PC1002.example.co.jp	172.16.0.2
PC1003.example.co.jp	172.16.0.3
⋮	⋮
PC2001.example.co.jp	172.31.0.1
PC2002.example.co.jp	172.31.0.2
PC2003.example.co.jp	172.31.0.3
⋮	⋮

ドメイン名	IPアドレス
GATEWAY.example.co.jp	172.16.0.101
	172.31.0.101
SV01.example.co.jp	172.31.0.91
SV01.example.co.jp	172.31.0.92
SV01.example.co.jp	172.31.0.93

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。

ネームサーバは、ドメイン名から IP アドレスを検索するための定義ファイルをもつ。このファイルでは、ネームサーバを次のとおりに定義する。

<定義するドメイン名>. IN NS <ネームサーバのドメイン名>.

また、ドメイン名と IP アドレスの対応は、次のとおりに定義する。

<ドメイン名>. IN A <IP アドレス>

GATEWAY.example.co.jp をネームサーバとするとき、このネームサーバがもつ定義ファイルの記述の一部を次に示す。

```
example.co.jp.   IN NS  a .example.co.jp.  
  
localhost.example.co.jp. IN A 127.0.0.1  
PC1001.example.co.jp. IN A 172.16.0.1  
PC1002.example.co.jp. IN A 172.16.0.2  
PC1003.example.co.jp. IN A 172.16.0.3  
    ⋮  
GATEWAY.example.co.jp. IN A 172.16.0.101  
GATEWAY.example.co.jp. IN A  b  
  
PC2001.example.co.jp. IN A 172.31.0.1  
PC2002.example.co.jp. IN A 172.31.0.2  
PC2003.example.co.jp. IN A 172.31.0.3  
    ⋮  
GATEWAY-1.example.co.jp. IN A 172.16.0.101  
GATEWAY-2.example.co.jp. IN A 172.31.0.101  
  
SV01.example.co.jp. IN A 172.31.0.91  
SV01.example.co.jp. IN A 172.31.0.92  
SV01.example.co.jp. IN A 172.31.0.93
```

解答群

- | | | |
|-------------|---------------|----------------|
| ア 127.0.0.1 | イ 172.31.0.91 | ウ 172.31.0.101 |
| エ GATEWAY | オ localhost | カ SV01 |

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。

DNS では、ネームサーバに問合せを行うクライアントをリゾルバと呼ぶ。

同じドメイン名に対して異なるサブネットワークに属する IP アドレスが定義されている場合、ネームサーバは、リゾルバと同じサブネットワークに属する IP アドレスを優先して返す。この仕組みが適用されるのは .example.co.jp の定義である。

一方、同じドメイン名に対して同じサブネットワークに属する異なる IP アドレスが定義されている場合、ネームサーバは、IP アドレスを定義順に巡回しながら返す。この仕組みをラウンドロビンと呼び、サーバへのアクセスを分散するために用いられる。この仕組みが適用されるのは .example.co.jp の定義である。

解答群

ア GATEWAY-1

イ GATEWAY-2

ウ GATEWAY

エ localhost

オ SV01

問2 オーバレイに関する次の記述を読んで、設問1, 2に答えよ。

近年、組込み (embedded) システムに利用されるプログラムの規模拡大に伴い、オーバレイが必要になってきている。オーバレイとは、プログラムを幾つかのオーバレイセグメント (以下、単にセグメントと呼ぶ) に分割しておき、OS がプログラムの実行に必要なモジュールを含むセグメントだけを主記憶領域に読み込んで実行する方法である。

- (1) 10 個のモジュール A ~ J で構成されるプログラムがあり、各モジュールは図1に示す呼出し構造になっている。例えば、モジュール G は二つのモジュール H 及び I を呼び出す。

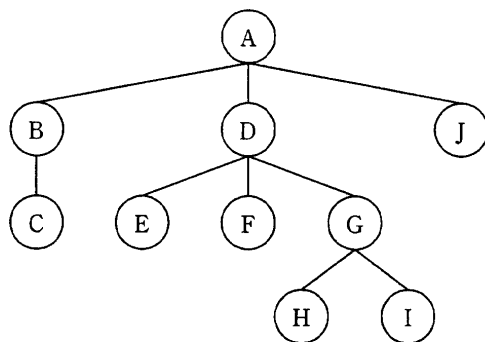


図1 モジュールの呼出し構造

- (2) このプログラムのモジュールの実行順序は、図2のとおりである。

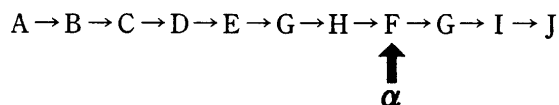


図2 モジュールの実行順序

- (3) A ~ J の各モジュールの実行に必要な主記憶領域の大きさは、表のとおりである。
なお、セグメントの大きさは、セグメントを構成するモジュールの大きさを合計したものになる。

表 モジュールの大きさ

モジュール	A	B	C	D	E	F	G	H	I	J
大きさ (M バイト)	10	8	6	5	6	4	6	2	3	15

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。

プログラムのオーバーレイ構造と各セグメントを構成するモジュールを図 3 に示す。図 3 は、モジュール D がモジュール A から呼び出されたとき、主記憶領域にセグメント P0 及び P2 が読み込まれていることを表している。また、セグメント P1, P2 及び P3 は、主記憶領域の同じ番地を先頭としてそれ以降に読み込まれることを表している。

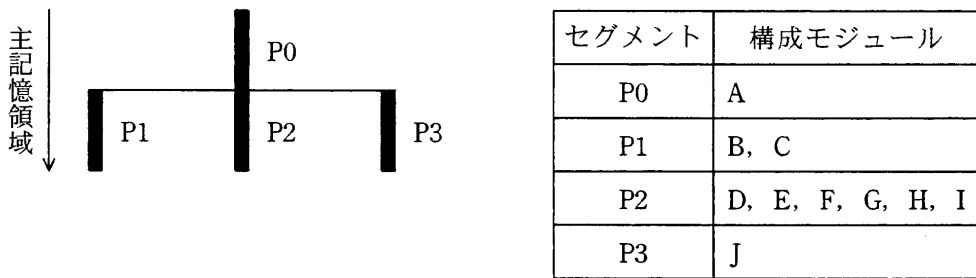


図 3 オーバーレイ構造とセグメント

このプログラムを実行したとき、プログラムの実行が終了するまでに、セグメントは P0, P1, P2, P3 の順に、計 4 回だけ主記憶領域に読み込まれる。図 3 のオーバーレイ構造で実行するのに必要な主記憶領域は、 M バイトである。

解答群

- | | | | |
|------|------|------|------|
| ア 24 | イ 25 | ウ 35 | エ 36 |
| オ 37 | カ 38 | キ 39 | ク 40 |

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。

プログラムの実行時に使用する主記憶領域を減らすことが必要になったので、オーバーレイ構造及びセグメントの見直しを行って、図4と図5に示す二つの案を作成した。

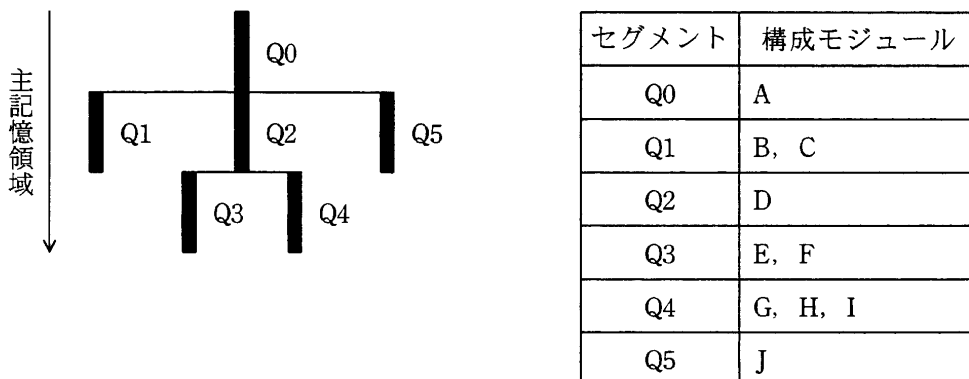


図4 オーバレイ構造とセグメント案1

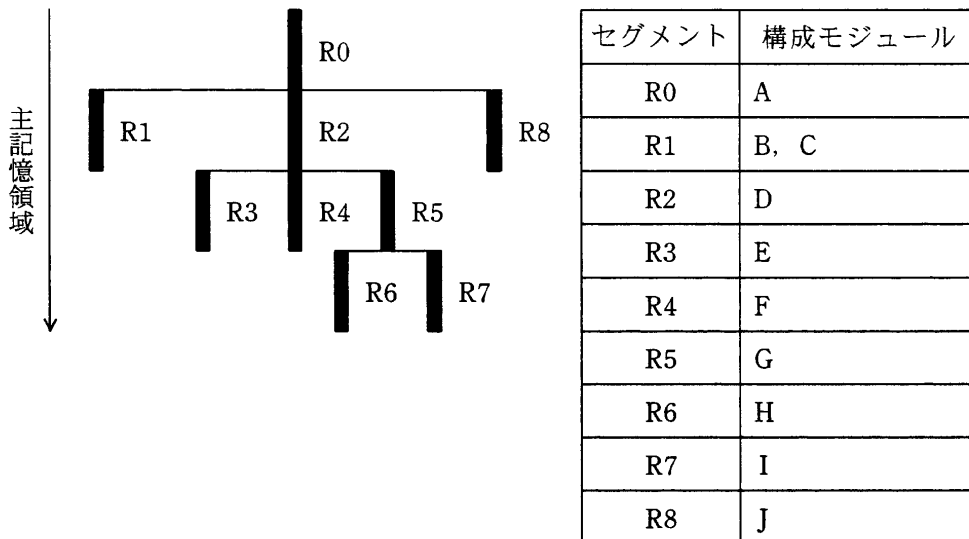


図5 オーバレイ構造とセグメント案2

図3のオーバーレイ構造に比べて、プログラムの実行時に使用する主記憶領域を、案1では10Mバイト、案2では Mバイト減らすことができる。

プログラムを実行したとき、図2中の α の位置にあるモジュールFの実行が終了

するまでに主記憶領域にセグメントが読み込まれる回数は、案1では合計6回、案2では合計 回である。

プログラムの実行が終了するまでに主記憶領域にセグメントが読み込まれる回数は、図3のオーバレイ構造に比べて、案1では合計 回、案2では合計6回増える。

aに関する解答群

ア 10	イ 11	ウ 12
エ 13	オ 14	カ 15

b, cに関する解答群

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

問 3 暗号通信に関する次の記述を読んで、設問に答えよ。

安全でない通信経路を利用する 2 者が、秘密の共通かぎを安全に共有する方式として Diffie-Hellman 法（以下、DH 法という）が知られている。DH 法で利用者 A と利用者 B が共通かぎ K を共有するまでの手順は、次のとおりである。

- (1) 素数 p と、 p よりも小さいある自然数 α が公開されていて、利用者 A と利用者 B がともに知ることができる。
- (2) 利用者 A は、 p よりも小さい任意の自然数 X_A を選び、秘密かぎとして保持するとともに、次の式で得られる公開かぎ Y_A を利用者 B に送る。

$$Y_A = \alpha^{X_A} \bmod p$$

ここで、 $x \bmod y$ は 整数 x を整数 y で割った余り（剰余）である。

- (3) 利用者 B は、 p よりも小さい任意の自然数 X_B を選び、秘密かぎとして保持するとともに、次の式で得られる公開かぎ Y_B を利用者 A に送る。

$$Y_B = \alpha^{X_B} \bmod p$$

- (4) 利用者 A は、利用者 B の公開かぎ Y_B を使って、次の式によって共通かぎ K を得る。

$$K = Y_B^{X_A} \bmod p$$

- (5) 利用者 B は、利用者 A の公開かぎ Y_A を使って、次の式によって利用者 A と同じ共通かぎ K を得る。

$$K = Y_A^{X_B} \bmod p$$

DH 法によるかぎ共有を利用して、図に示すように、安全でない通信経路を利用する 2 者間で機密情報を送信する仕組みを作る。まず、利用者 A と利用者 B は DH 法を使って、共通かぎ K を共有する。次に、利用者 A は平文を共通かぎ K を使って暗号化して送信する。利用者 B は受信した暗号文を共通かぎ K を使って復号して、元の平文を得ることができる。

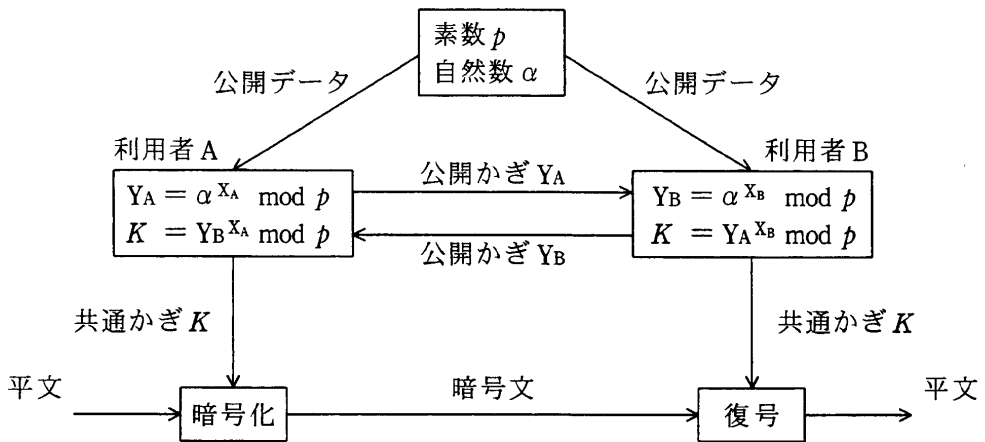


図 DH 法を使った機密情報の通信

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。

DH 法は、目的が a に限定されたアルゴリズムであるが、この方式の安全性はほかの公開かぎ暗号と同じく計算の一方方向性にに基づいている。すなわち、DH 法において、素数 p の値が十分に大きい場合、秘密かぎから公開かぎを求めるのは容易であるが、公開かぎから秘密かぎを求めるのは非常に困難である。

反対に、素数 p の値が小さい場合には、かぎの値が小さくなるので公開かぎから秘密かぎを短時間で求めることも可能であり、安全性に問題がある。例えば、利用者 A と利用者 B が使う通信経路上に通信を傍受している第三者 C がいて、公開されている素数 p が 7、 α が 5 であることに加え、利用者 A が送った公開かぎ Y_A が 6、利用者 B が送った公開かぎ Y_B が 3 であることを知ったとする。このとき、共通かぎ K の値は、 b であることが容易に分かる。

また、DH 法は、通信経路上の第三者 C が、利用者 A から送られる情報を、にせの情報にすりかえて利用者 B に送信する中間者攻撃に対して、弱いことが知られている。中間者攻撃を防ぐためには公開かぎに信頼できる第三者によるデジタル署名をつけるなどの対策が必要である。さらに、第三者 C が暗号文を傍受してそれを手掛かりとして共通かぎ K を見つけるリスクもあるので、継続的な通信の安全性を高めるための対策として、共通かぎ K の値が十分に大きいものを使うだけでなく、 c も有効である。

a, c に関する解答群

- ア 共通かぎ K の更新間隔の短縮
- イ 公開かぎ Y_A, Y_B の交換回数の削減
- ウ 公開かぎの交換
- エ 秘密かぎによる情報の復号
- オ 秘密の共通かぎの共有
- カ より大きな値の素数 p の使用

b に関する解答群

- | | | | |
|-----|-----|-----|-----|
| ア 0 | イ 1 | ウ 2 | エ 3 |
| オ 4 | カ 5 | キ 6 | |

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1～3に答えよ。

[プログラムの説明]

中置表記法による正しい数式を、スタックを用いて後置表記法に変換する副プログラム `toPostfix` である。

- (1) 例えば、中置表記法による数式 $4 \times (9 + 3)$ は、`toPostfix` の処理の結果、後置表記法による数式 $4 9 3 + \times$ に変換される。
- (2) 中置表記法によるここでの数式は、1けたの数字“0”～“9”、演算子“+”、“-”、“ \times ”、“ \div ”及び括弧“(”、“)”の各要素からなる。各要素は表1に示す変換の優先度をもつ。ここで、数値が大きいほど優先度が高い。プログラムでは、各要素を文字として扱い、文字型の配列 `Exptext` に1要素ずつ格納している。また、表1に示す要素以外に、スタックの底を示すスタック制御文字 `EOS` を使用し、その優先度を-1とする。

表1 変換の優先度

数式の要素	優先度
(4
0, 1, ..., 9	3
\times , \div	2
+, -	1
)	0

- (3) 変換結果の後置表記法の数式は、文字型の配列 `Postfix` に1要素ずつ格納される。
- (4) `toPostfix` は、スタックを初期化して、`Exptext` の先頭の要素から順番に次の①、②の処理を繰り返し、`Exptext` の要素をすべて処理したら、スタックに残っているすべての数式の要素を順に取り出して `Postfix` に格納する。
- ① `Exptext[i]` の優先度がスタックの先頭要素の優先度以下で、かつスタックの先頭要素が“(”でない間、スタックから要素を取り出して `Postfix` に格納する。

② Exptext[i] が “)” でなければ、Exptext[i] をスタックに積み、“)” であれば、スタックから要素を一つだけ取り出す。

(5) toPostfix の引数の仕様を、表 2 に示す。すべての配列の添字は、0 から始まる。

表 2 副プログラム toPostfix の引数の仕様

変数名	型	入力／出力	意味
Exptext[]	文字型	入力	中置表記法による数式が格納されている 1 次元配列
Textlen	整数型	入力	中置表記法による数式の要素数
Postfix[]	文字型	出力	後置表記法に変換後の数式が格納される 1 次元配列
Postfixlen	整数型	出力	後置表記法に変換後の数式の要素数

(6) toPostfix は、スタックを初期化する手続 `initStack`、数式の要素をスタックに積む手続 `push`、スタックから取り出す関数 `pop`、スタックの先頭要素の値を返却する関数 `top`、及び引数で指定された要素の優先度を返す関数 `getPriority` を使用する。`initStack`、`push`、`pop`、`top` 及び `getPriority` の仕様を表 3 に示す。

表 3 toPostfix で使用する手続／関数の仕様

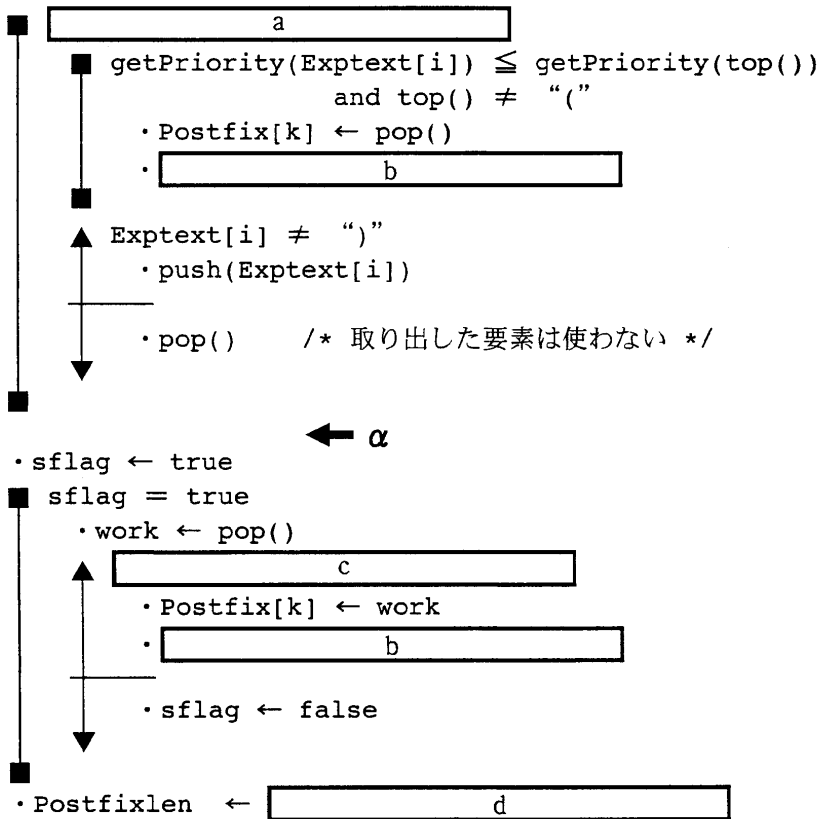
手続／関数	仕様
<code>initStack()</code>	スタックを初期化し、スタックに EOS を格納する。
<code>push(文字型 : element)</code>	スタックに <code>element</code> を格納する。
文字型 : <code>pop()</code>	スタックから要素を取り出し、その要素を返却する。
文字型 : <code>top()</code>	スタックの先頭要素の値を返却する。
整数型 : <code>getPriority</code> (文字型 : <code>element</code>)	引数で指定された要素 <code>element</code> の優先度を返却する。

[プログラム]

○ toPostfix(文字型: Exptext[], 整数型: Textlen,
 文字型: Postfix[], 整数型: Postfixlen)

- 整数型: k, i
- 論理型: sflag
- 文字型: work

・initStack()
・k ← 0



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 $i:0, i < \text{Textlen}, 1$ | イ $i:0, i \leq \text{Textlen}, 1$ |
| ウ $i:1, i < \text{Textlen}, 1$ | エ $i:1, i \leq \text{Textlen}, 1$ |

bに関する解答群

- | | |
|------------------------|------------------------|
| ア $i \leftarrow i + 1$ | イ $i \leftarrow i - 1$ |
| ウ $k \leftarrow k + 1$ | エ $k \leftarrow k - 1$ |

cに関する解答群

- | | |
|--------------------------------|---------------------------------|
| ア $\text{sflag} = \text{true}$ | イ $\text{sflag} = \text{false}$ |
| ウ $\text{work} = \text{EOS}$ | エ $\text{work} \neq \text{EOS}$ |

dに関する解答群

- | | | |
|-------|-----------|-----------|
| ア i | イ $i + 1$ | ウ $i - 1$ |
| エ k | オ $k + 1$ | カ $k - 1$ |

設問2 Exptext の内容が次の数式のと看、プログラム中の α で示す箇所での Postfix の内容として正しい答えを、解答群の中から選べ。

Exptext の内容

(2	+	4)	×	3	+	5	×	6
---	---	---	---	---	---	---	---	---	---	---

解答群

- | | |
|-----------------------------|-------------------------------|
| ア $24 + 3$ | イ $24 + 3 \times 5$ |
| ウ $24 + 3 \times 56 \times$ | エ $24 + 3 \times 56 \times +$ |

設問3 引数で指定された要素の優先度を返す関数 `getPriority` のプログラムを次に示す。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。

[プログラム]

```

○ 整数型: getPriority ( 文字型: element )
/* 関数 getPriority は、数字、演算子、括弧又はスタック制御文字を
   element で受け取り、その優先度を整数型で返す */

/* 文字型の配列 elem を初期化する */
/* 添字の範囲は 0 ~ 16 である */
○ 文字型: elem[0:16] ← { "0", "1", "2", "3", "4", "5", "6", "7",
                          "8", "9", "+", "-", "x", "÷", "(", ")" , EOS }
/* 整数型の配列 priority を elem の各要素の優先度で初期化する */
/* 添字の範囲は 0 ~ 16 である */
○ 整数型: priority[0:16] ← { 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 1, 1, 2, 2, 4, 0, -1 }

○ 整数型: i

. i ← 0
■ 
|
■   . i ← i + 1
. return priority[i]   /* element の優先度を返す */

```

解答群

ア <code>element = elem[i]</code>	イ <code>element = elem[i-1]</code>
ウ <code>element = EOS</code>	エ <code>element = priority[i]</code>
オ <code>element = priority[i-1]</code>	カ <code>element ≠ elem[i]</code>
キ <code>element ≠ elem[i-1]</code>	

問5 プログラム設計に関する次の記述を読んで、設問1～4に答えよ。

社員が所属する部署、従事する職務や階級などを管理する社員表を、昇級、昇格、職務変更などの人事変更情報を基に更新するシステムを設計する。

[A社の人事規定と人事変更情報]

- (1) 社員には、入社時に7けたの数字からなる社員コードを割り当てる。社員コードは、社員を一意に特定できる重複しない番号である。
- (2) すべての社員は、一つの部署にだけ所属する。部署名は、部署コードをキーとして部署表に格納されている。部署表のレコード様式を次に示す。下線は、主キーを表す。

<u>部署コード</u>	部署名
--------------	-----

- (3) 営業、事務、技術などの職務名は、職務コードをキーとして職務表に格納されている。職務表のレコード様式を次に示す。また、すべての社員は一つだけの職務につく。

<u>職務コード</u>	職務名
--------------	-----

- (4) すべての社員は、管理職か一般職かのいずれかの職群に属する。管理職には1～20級の階級があり、一般職には1～30級の階級がある。階級の値が減ることを階級が上がるという。
- (5) 社員の入社年月日、所属する部署、従事する職務などの情報は、社員コードをキーとして社員表に格納されている。社員表のレコード様式を次に示す。

<u>社員コード</u>	社員名	入社年月日	部署コード	職務コード	職群	階級
--------------	-----	-------	-------	-------	----	----

- (6) 別の職務を希望する社員は、毎年3月1日から3月15日までに、希望する職務を会社に伝える。この情報は、職務変更ファイルに許可フラグの初期値を0として格納される。職務変更ファイルのレコード様式を次に示す。

社員コード	希望職務コード	許可フラグ
-------	---------	-------

- (7) 会社は、社員が希望する別の職務への従事を許可する場合、毎年3月31日までに、職務変更ファイルの許可フラグに1を設定する。その年の4月1日には、会社が許可した社員の職務を希望した職務に変更する。これを職務変更という。
- (8) 会社は、一般職の社員のうち管理能力に秀でた社員を、管理職に任命する。これを昇格という。昇格候補の社員の情報は、毎年3月31日までに昇格候補ファイルに格納される。昇格はその年の4月1日に行い、当日職務変更していない社員だけが昇格できる。昇格のとき、職群は管理職に、階級は20級になる。昇格候補ファイルのレコード様式を次に示す。

社員コード	昇格理由
-------	------

- (9) 会社は、毎年4月1日に職務変更も昇格もしていない社員の階級を2上げる。これを昇級という。ただし、昇級前の階級が2級又は1級の場合は1級となる。また、入社後1年に満たない社員は、昇級の対象から除外する。
- (10) 会社は、毎年4月1日に、社員に対して所属している部署から別の部署へ異動する人事異動を発令する。人事異動の発表資料の例を図1に示す。異動の対象社員の情報は、前日までに異動ファイルに格納される。異動ファイルのレコード様式を次に示す。

社員コード	異動先部署コード
-------	----------

人事異動

社員 コード	社員名	新				旧			
		部署名	職群	階級	職務名	部署名	職群	階級	職務名
1200851	情報 太郎	営業一部	一般職	18 級	技術	営業二部	一般職	18 級	営業
2000509	技術 花子	営業一部	一般職	12 級	事務	営業二部	一般職	14 級	事務
9971006	資格 弘海	営業一部	管理職	20 級	技術	営業二部	一般職	1 級	技術
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

図 1 人事異動の発表資料の例

- (11) 図 2 に社員表の更新に必要なファイル作成の流れ図を、図 3 に社員表の更新と人事異動の発表資料作成の流れ図を示す。図中の昇格者ファイルは、職群を更新する社員の社員コードだけのレコードからなるファイルである。階級更新ファイルは、階級を更新する社員の社員コードだけのレコードからなるファイルである。

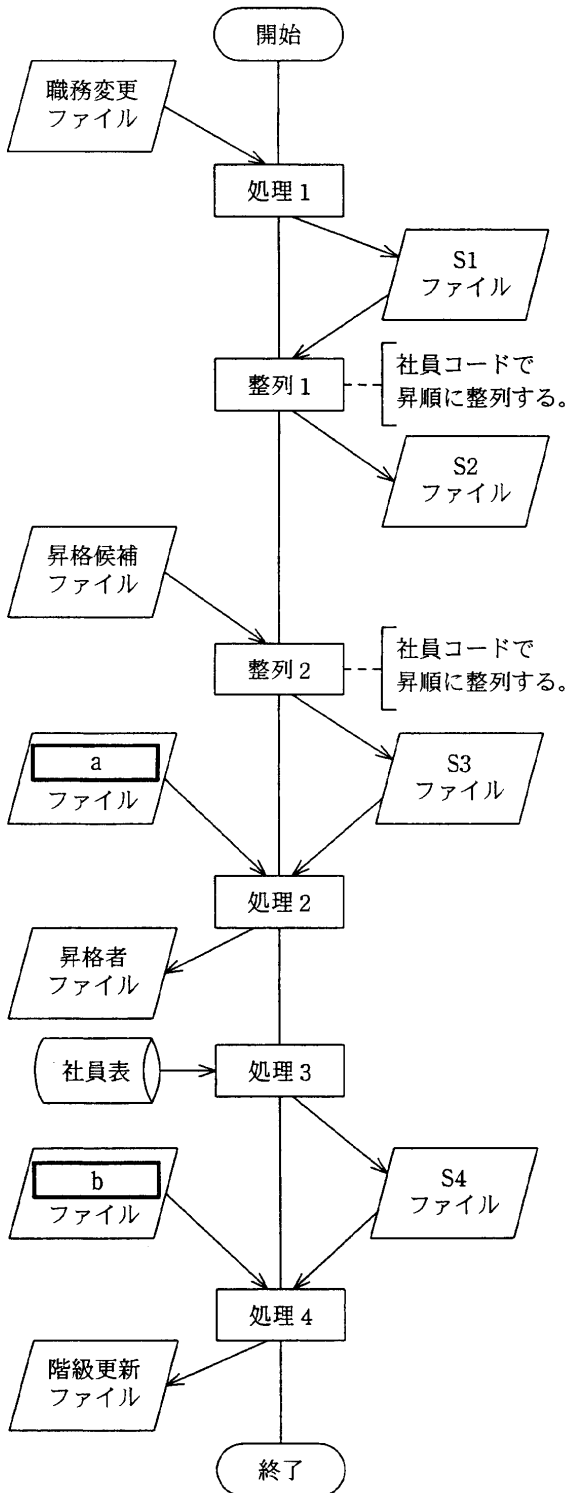


図2 昇格者ファイルと階級更新
ファイルの作成の流れ

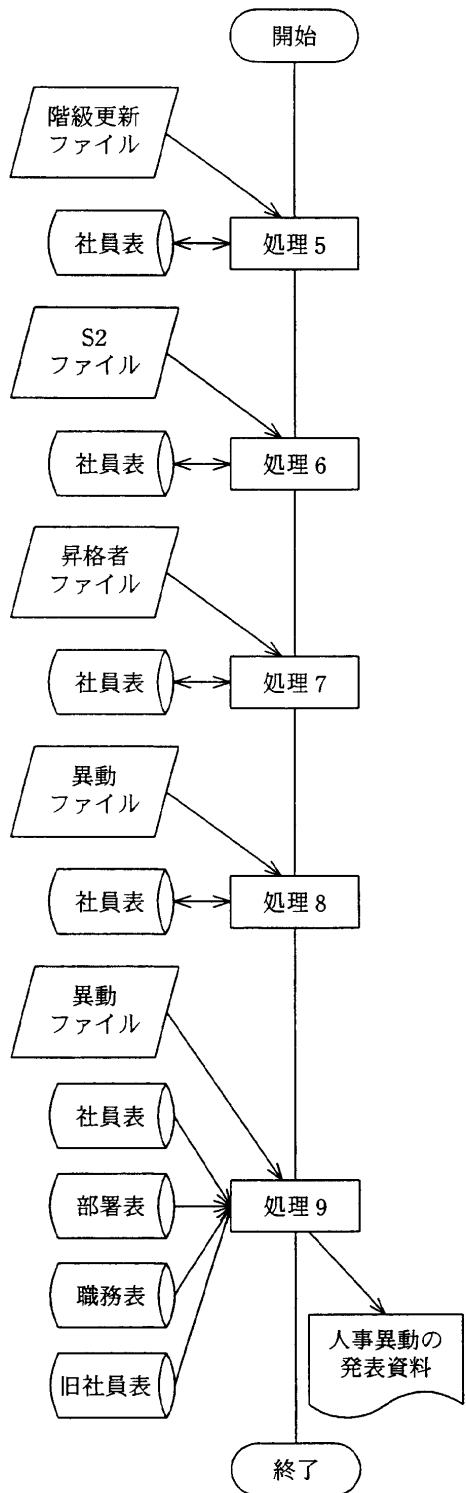


図3 社員表の更新と人事異動の
発表資料作成の流れ

表は、図 2 と図 3 で使用する処理を分類し、機能概要を一覧にしたものである。

表 処理と機能概要の一覧

処理	分類	機能概要
1	抽出	職務変更ファイルからレコードを読み込み、許可フラグが 1 のレコードを S1 ファイルとして出力する。
2	突合せ	<input type="text" value="a"/> ファイルと S3 ファイルを読み込み、読み込んだレコードの社員コードを比較し、 <input type="text" value="a"/> ファイルにはなく S3 ファイルにはある社員コードのレコードを、昇格者ファイルとして出力する。
3	抽出	社員表からレコードを読み込み、階級の値が 2～30 であって入社年月日が前年の 4 月 1 日以前のレコードを選び、社員コードの昇順に並べ、S4 ファイルとして出力する。
4	突合せ	<input type="text" value="b"/> ファイルと S4 ファイルを読み込み、読み込んだレコードの社員コードを比較し、 <input type="text" value="b"/> ファイルにはなく S4 ファイルにはある社員コードのレコードを、階級更新ファイルとして出力する。
5	更新	階級更新ファイルから読み込んだレコードの社員コードを用いて、社員表の階級を更新する。社員表を社員コードで検索して取得した階級が 2 級の場合、該当レコードの階級の値を 1 にする。それ以外の場合は、階級の値を 2 だけ引いた値に更新する。
6	更新	S2 ファイルから読み込んだレコードの社員コードと希望職務コードを用いて、社員表の該当レコードの職務コードを更新する。
7	更新	昇格者ファイルから読み込んだレコードの社員コードを用いて、社員表の該当レコードの職群を“管理職”に、階級の値を 20 に更新する。
8	更新	異動ファイルから読み込んだレコードの社員コードと異動先部署コードを用いて、社員表の該当レコードの部署コードを更新する。
9	作表	異動ファイルから読み込んだレコードの社員コードで社員表を検索し、旧社員表、部署表と職務表を用いて人事異動の発表資料を作表する。

設問 1 処理 2 では職群を更新する社員コードだけのレコードからなる昇格者ファイルを、処理 4 では階級を更新する社員コードだけのレコードからなる階級更新ファイルを、それぞれ作成する。図 2 及び表中の 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 S1 イ S2 ウ S3 エ 昇格者 オ 職務変更

設問2 図3の中で、処理5～8の順番を入れ替え、入替え前と同じ処理結果が得られるように並べたものはどれか。正しい答えを、解答群の中から選べ。

解答群

- ア 処理6→処理7→処理8→処理5 イ 処理7→処理8→処理5→処理6
ウ 処理8→処理5→処理6→処理7 エ 処理8→処理7→処理6→処理5

設問3 人事規定の見直しがあり、〔A社の人事規定と人事変更情報〕の(9)が次のとおり変更される(変更部分を~~~~で示す)。ファイルの項目を変更しない場合、図2と図3の中で、修正が必要な処理はどれか。正しい答えを、解答群の中から二つ選べ。

会社は、毎年4月1日に職務変更も昇格もしていない社員のうち入社してから1年以上の社員は、階級を2上げる。これを昇級という。ただし、昇級前の階級が2級又は1級の場合は1級となる。また、入社後1年に満たない社員は、階級を1だけ上げる。

解答群

- ア 処理1 イ 処理2 ウ 処理3 エ 処理4
オ 処理5 カ 処理6 キ 処理7 ク 処理8

設問4 図3の処理9で用いる社員表には、人事異動の結果が反映されている。人事異動の発表資料を作表するために必要な旧社員表として正しい答えを、解答群の中から選べ。

解答群

- ア 処理5の更新反映前の社員表
イ 処理5の更新反映後、かつ処理6の更新反映前の社員表
ウ 処理6の更新反映後、かつ処理7の更新反映前の社員表
エ 処理7の更新反映後、かつ処理8の更新反映前の社員表

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]

関数 `update_master` は、処理年月を与えて、月刊誌の定期購読者マスタファイルの更新を行うプログラムである。

(1) 関数 `update_master` の引数は、次のとおりである。

<code>omf_name</code>	旧定期購読者マスタファイル名
<code>trf_name</code>	トランザクションファイル名
<code>nmf_name</code>	新定期購読者マスタファイル名
<code>b_year</code>	処理年月の西暦年
<code>b_month</code>	処理年月の月

(2) 新旧の定期購読者マスタファイル及びトランザクションファイルのレコード様式は次のとおりである。

購読者コード 12 けた	空白 1 けた	雑誌コード 12 けた	空白 1 けた	購読終了年月 6 けた
-----------------	------------	----------------	------------	----------------

- ① 購読者コード及び雑誌コードは、空白を含まない12文字の英数字列である。
 - ② 購読終了年月は、YYYYMMの形式で6けたの数字列である。
 - ③ レコードの終端には、改行文字 '\n' が付いている。
 - ④ レコードは、購読者コードを第1キー、雑誌コードを第2キーとして昇順に整列されている。
 - ⑤ 購読者は、複数の雑誌を購読することはあるが、同一の雑誌を重複して購読することはない。
- (3) 旧定期購読者マスタファイル及びトランザクションファイルのレコードに誤りは無いものとする。

- (4) トランザクションファイルには、旧定期購読者マスタファイルを更新するための、新規購読、購読期間延長及び購読打ちりの3種類のレコードがある。
- ① 新規購読及び購読期間延長レコードの購読終了年月は、与えられた処理年月以降になっている。
 - ② 購読打ちりレコードの購読終了年月は、“999999”となっている。
- (5) 次のどちらかの条件を満足するレコードを、抽出レコードとする。
- ① 旧定期購読者マスタファイルのレコードのうち、購読者コードと雑誌コードの組合せがトランザクションファイルには含まれていないもの
 - ② トランザクションファイルに含まれるレコード
- (6) 抽出レコードのうち、購読終了年月が与えられた処理年月以降であり、かつ“999999”ではないものだけを、新定期購読者マスタファイルに出力する。
- (7) プログラム中で使用している関数 `strcmp` の仕様は、次のとおりである。

```
int strcmp(char *string1, char *string2)
```

機能： 引数で指定される二つの文字列 `string1` と `string2` を比較し、その大小関係を返す。

返却値：負の数 (`string1` が `string2` より小さいとき)

0 (`string1` が `string2` に等しいとき)

正の数 (`string1` が `string2` より大きいとき)

[プログラム]

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

void update_master(char *, char *, char *, int, int);

void update_master(char *omf_name, char *trf_name,
                   char *nmf_name, int b_year, int b_month)
{
    FILE *oldmf,          /* 旧定期購読者マスタファイル */
          *trf,          /* トランザクションファイル */
          *newmf;        /* 新定期購読者マスタファイル */
    char om_usrid[13], om_magid[13], t_usrid[13], t_magid[13];
    long b_date, om_ldate, t_ldate, m_sts, t_sts, flg;

    oldmf = fopen(omf_name, "r");
    trf = fopen(trf_name, "r");
    newmf = fopen(nmf_name, "w");
    b_date = b_year * 100 + b_month;
    m_sts = fscanf(oldmf, "%12s %12s %6ld\n",
                  om_usrid, om_magid, &om_ldate);
    t_sts = fscanf(trf, "%12s %12s %6ld\n",
                  t_usrid, t_magid, &t_ldate);

    while (  ) {
        if ( m_sts == EOF )
            flg = 1;
        else if ( t_sts == EOF )
            flg = -1;
        else if ( (flg = strcmp(om_usrid, t_usrid)) == 0 )
            flg = strcmp(om_magid, t_magid);

        if ( flg < 0 ) {
            if ( om_ldate >= b_date )
                fprintf(newmf, "%12.12s %12.12s %6ld\n",  );
            m_sts = fscanf(oldmf, "%12s %12s %6ld\n",
                          om_usrid, om_magid, &om_ldate);
        } else {
            if (  )
                fprintf(newmf, "%12.12s %12.12s %6ld\n",  );
            if ( flg == 0 )
                m_sts = fscanf(oldmf, "%12s %12s %6ld\n",
                              om_usrid, om_magid, &om_ldate);
            t_sts = fscanf(trf, "%12s %12s %6ld\n",
                          t_usrid, t_magid, &t_ldate);
        }
    }
    fclose(oldmf);
    fclose(trf);
    fclose(newmf);
}
```

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 (m_sts == EOF) && (t_sts == EOF)
- イ (m_sts == EOF) || (t_sts == EOF)
- ウ (m_sts != EOF) && (t_sts != EOF)
- エ (m_sts != EOF) || (t_sts != EOF)
- オ m_sts == EOF
- カ m_sts != EOF
- キ t_sts == EOF
- ク t_sts != EOF

b, dに関する解答群

- ア om_usrid, om_magid, om_ldate
- イ om_usrid, om_magid, t_ldate
- ウ om_usrid, t_magid, om_ldate
- エ om_usrid, t_magid, t_ldate
- オ t_usrid, om_magid, om_ldate
- カ t_usrid, om_magid, t_ldate
- キ t_usrid, t_magid, om_ldate
- ク t_usrid, t_magid, t_ldate

cに関する解答群

- | | |
|----------------------|----------------------|
| ア b_date == 999999 | イ b_date != 999999 |
| ウ om_ldate == 999999 | エ om_ldate != 999999 |
| オ t_ldate == 999999 | カ t_ldate != 999999 |

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]

1 学年における全生徒のテスト結果が格納されている入力ファイルを読み込み、5 科目の合計点を求めるプログラムである。このプログラムは、学年内の総合順位が分かるよう、合計点の降順に整列して、出力ファイルに格納する。

(1) 入力ファイル IN-FILE は、次のレコード様式の順ファイルであり、1 学年分の各生徒の得点が格納されている。ここで、各科目は 100 点を満点とする。

クラス番号 2 けた	生徒番号 2 けた	氏名 20 けた	国語 3 けた	数学 3 けた	英語 3 けた	理科 3 けた	社会 3 けた
---------------	--------------	-------------	------------	------------	------------	------------	------------

(2) 出力ファイル OUT-FILE は、次のレコード様式の順ファイルであり、5 科目の合計点の降順に格納する。

クラス番号 2 けた	生徒番号 2 けた	氏名 20 けた	合計点 3 けた
---------------	--------------	-------------	-------------

[プログラム]

(行番号)

```

1 DATA DIVISION.
2 FILE SECTION.
3 FD IN-FILE.
4 01 IN-REC          PIC X(39).
5 FD OUT-FILE.
6 01 OUT-REC        PIC X(27).
7 SD SORT-FILE.
8 01 SORT-REC.
9   02 CLASS-NO     PIC 9(2).
10  02 STUDENT-NO    PIC 9(2).
11  02 STUDENT-NAME PIC X(20).
12  02 TOTAL        PIC 9(3).
13 WORKING-STORAGE SECTION.
14 01 W-IN-REC.
15   02 STUDENT-ID.
16     03 CLASS-NO     PIC 9(2).
17     03 STUDENT-NO    PIC 9(2).
18     03 STUDENT-NAME PIC X(20).
19     02 SCORE        PIC 9(3) OCCURS 5.

```

```

20 01 READ-STATUS PIC X(1) VALUE SPACE.
21     88 AT-END VALUE "E".
22 01 K PIC 9(1).
23 PROCEDURE DIVISION.
24 SORT-PROCEDURE.
25     SORT SORT-FILE DESCENDING KEY TOTAL
26     INPUT PROCEDURE IS READ-DATA
27     GIVING OUT-FILE.
28     STOP RUN.
29 READ-DATA.
30     OPEN INPUT IN-FILE.
31     PERFORM UNTIL AT-END
32     READ IN-FILE AT END
33     
34     NOT AT END
35     MOVE IN-REC TO W-IN-REC
36     PERFORM RELEASE-DATA
37     END-READ
38     END-PERFORM.
39     CLOSE IN-FILE.
40 RELEASE-DATA.
41     .
42     MOVE ZERO TO TOTAL.
43     PERFORM VARYING K FROM 1 BY 1 UNTIL K > 5
44     COMPUTE TOTAL = TOTAL + SCORE(K)
45     END-PERFORM.
46     RELEASE SORT-REC.

```

COBOL

設問1 プログラム中の に入れる正しい答えを、解答群の中から選べ。

解答群

- ア INITIALIZE SORT-REC
- イ MOVE SPACE TO READ-STATUS
- ウ MOVE SPACE TO SORT-REC
- エ MOVE STUDENT-ID TO SORT-REC
- オ PERFORM RELEASE-DATA
- カ SET AT-END TO TRUE

設問2 プログラムの最後で生徒数及び各科目の学年平均点を表示するようにプログラムを変更する。変更内容を示す次の表中の に入れる正しい答えを、解答群の中から選べ。

なお、生徒の人数は1～9,999とし、平均点は小数第2位以下を切り捨てる。

処置	プログラムの変更内容
行番号22と23の間に追加	01 STATISTICS. 02 STUDENT-TOTAL PIC 9(4) VALUE ZERO. 02 SUB-TOTAL PIC 9(8) VALUE ZERO OCCURS 5. 02 AVERAGE PIC 999.9.
行番号27と28の間に追加	DISPLAY "STUDENT:". DISPLAY STUDENT-TOTAL. DISPLAY "AVERAGE:". PERFORM VARYING K FROM 1 BY 1 UNTIL K > 5 COMPUTE AVERAGE = SUB-TOTAL(K) / STUDENT-TOTAL DISPLAY AVERAGE END-PERFORM.
行番号44と45の間に追加	<input type="text" value="c"/>
行番号46の後に追加	<input type="text" value="d"/> .

COBOL

解答群

- ア COMPUTE STUDENT-TOTAL = STUDENT-TOTAL + 1
- イ COMPUTE STUDENT-TOTAL = STUDENT-TOTAL + K
- ウ COMPUTE SUB-TOTAL(K) = SUB-TOTAL(K) + SCORE(K)
- エ MOVE K TO STUDENT-TOTAL
- オ MOVE SCORE(K) TO SUB-TOTAL(K)
- カ MOVE TOTAL TO SUB-TOTAL(K)

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。

〔プログラムの説明〕

英文のテキストを処理し、単語の出現回数を数えるプログラムの一部である。テキストは、単語、空白、コンマ及びピリオドで構成され、単語はアルファベットだけで構成される。単語中の大文字はすべて小文字に変換して取り扱う。さらに、単語の出現回数だけでなく、アルファベットの各文字が単語の先頭に現れた回数を数えられるようにする。

- (1) `WordTable` は、与えられた文字列から単語を切り出し、出現回数を数えるクラスである。実際に単語を数える処理は、コンストラクタで指定する別のオブジェクトで行う。
- (2) `Counter` は、単語を数える処理のインターフェースである。
- (3) `WordCounter` は、単語の出現回数を数えるクラスである。
- (4) `FirstLetterCounter` は、アルファベットの各文字が単語の先頭に現れた回数を数えるクラスである。
- (5) `Test` は、テスト用のメインプログラムである。実行例を図に示す。

```
wordCountTable:
java(2)
hello(1)
wonderful(1)
world(1)
is(1)

firstLetterCountTable:
h(1)
i(1)
j(2)
w(2)
```

図 クラス `Test` の実行例

`java.util.StringTokenizer` は、指定された区切り文字で、文字列を字句単位（トークン）に分解するクラスである。次のメソッドをもつ。

```
public boolean hasMoreTokens()
```

文字列に利用できるトークンがまだあるかどうかを判定し、結果を返す。

```
public String nextToken()
```

次のトークンを返す。

`java.util.Map` は、キーと値を関連付けて管理するインタフェースであり、`java.util.HashMap` は、そのインタフェースを実装したクラスである。キーを指定して、そのキーに値を関連付けたり、そのキーに関連付けられた値を取り出したりすることができる。次のメソッドをもつ。

```
public Object get(Object key)
```

`key` に関連付けられた値を返す。

```
public Object put(Object key, Object value)
```

`key` に `value` を関連付ける。

```
public boolean containsKey(Object key)
```

`key` に関連付けられた値がある場合に `true` を返す。

```
public Set keySet()
```

マップに含まれるキーの集合を返す。

`java.util.Iterator` は、要素を順番に取り出すための操作を提供するインタフェースである。次のメソッドをもつ。

```
public boolean hasNext()
```

次の要素がある場合に `true` を返す。

```
public Object next()
```

次の要素を返す。

[プログラム1]

```
import java.util.StringTokenizer;

public class WordTable {
    private                      a                      counter;
    public WordTable(                     a                      counter) {
        this.counter = counter;
    }
    public void put(String line) {
        StringTokenizer st = new StringTokenizer(line, " ,.");
        while (st.hasMoreTokens())
            counter.put(st.nextToken().toLowerCase());
    }
    public String toString() {
        return counter.toString();
    }
}
```

[プログラム2]

```
public interface Counter {
    void put(String str);
}
```

[プログラム3]

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

public class WordCounter                      b                      {
    private Map freq = new HashMap();
    public void put(String str) {
        int count =                      c                     ;
        if (freq.containsKey(str))
            count += ((Integer) freq.get(str)).intValue();
        freq.put(str, new Integer(count));
    }
    public String toString() {
        StringBuffer buf = new StringBuffer();
        for (Iterator it = freq.keySet().iterator();
             it.hasNext(); ) {
            String word = (String) it.next();
            buf.append(word + "(" + freq.get(word) + ")\n");
        }
        return buf.toString();
    }
}
```

[プログラム 4]

```
public class FirstLetterCounter b {
    private int[] flFreq = new int[26];
    public void put(String str) {
        d++;
    }
    public String toString() {
        StringBuffer buf = new StringBuffer();
        for (int i = 0; i < flFreq.length; i++) {
            if (flFreq[i] != 0)
                buf.append((char) ('a' + i)
                    + "(" + flFreq[i] + ")\n");
        }
        return buf.toString();
    }
}
```

[プログラム 5]

```
public class Test {
    public static void main(String[] args) {
        String text = "Hello java world. Java is wonderful.";
        WordTable wordCountTable =
            new WordTable(new WordCounter());
        WordTable firstLetterCountTable =
            new WordTable(new FirstLetterCounter());
        wordCountTable.put(text);
        firstLetterCountTable.put(text);
        System.out.println("wordCountTable:\n" +
            wordCountTable);
        System.out.println("firstLetterCountTable:\n" +
            firstLetterCountTable);
    }
}
```

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 Counter | イ FirstLetterCounter |
| ウ Object | エ WordCounter |
| オ WordTable | |

bに関する解答群

- | | |
|---------------------|------------------------|
| ア extends Counter | イ extends Object |
| ウ extends WordTable | エ implements Counter |
| オ implements Object | カ implements WordTable |

cに関する解答群

- ア -1
- イ 0
- ウ 1
- エ ((Integer) freq.get(str)).intValue()
- オ str.length()

dに関する解答群

- ア flFreq['a' + str.charAt(0)]
- イ flFreq['a' + str.charAt(1)]
- ウ flFreq[str.charAt(0)]
- エ flFreq[str.charAt(1)]
- オ flFreq[str.charAt(0) - 'a']
- カ flFreq[str.charAt(1) - 'a']

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1の説明〕

プログラム1 (SFT1) は、マスクを右にシフトしながら1語に格納されているデータの中の1であるビットの個数を数えて、GR0に設定する副プログラムである。

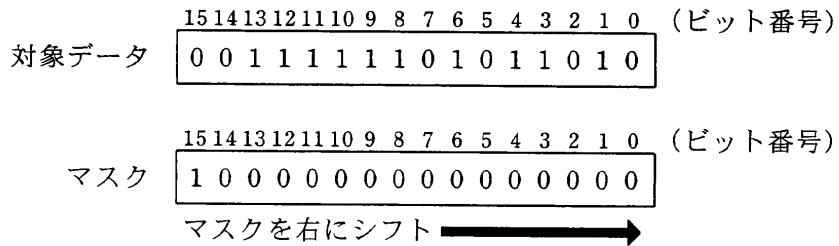


図1 プログラム1の処理方法

- (1) 主プログラムは、GR1に対象データを格納して副プログラムを呼ぶ。
- (2) 対象データの第15ビットから第0ビットまでを順にマスクと照合し、1の個数を数えて、GR0に設定する。
- (3) 副プログラムから戻るとき、汎用レジスタGR1～GR7の内容は元に戻す。

〔プログラム1〕

```

SFT1  START
      RPUSH
      LD   GR2, MASK
      LAD  GR0, 0
LOOP  LD   GR3, GR2
      AND  GR3, GR1
      JZE  SKIP
      ADDA GR0, =1
SKIP  SRL  GR2, 1
      a
      RPOP
      RET
MASK  DC   #8000
      END

```

← α

7E7E7E

〔プログラム 2 の説明〕

プログラム 2 (SFT2) は、プログラム 1 と比べて、命令の実行回数が少なくなるように、対象データを右にシフトしながら常に最下位ビットをマスクと照合し、1 の個数を数えて、GR0 に設定する副プログラムである。

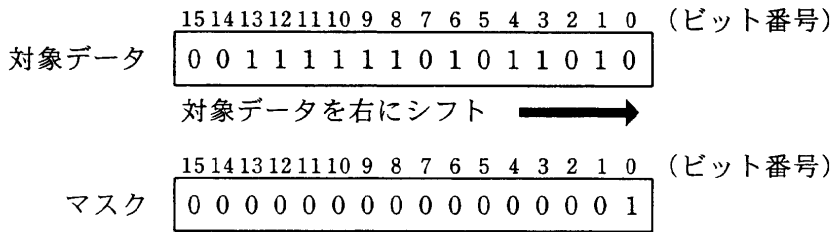


図 2 プログラム 2 の処理方法

〔プログラム 2〕

```

SFT2  START
      RPUSH
      LAD  GR0,0
LOOP  LD   GR3,GR1      ← β
      b
      ADDA GR0,GR3
      SRL  GR1,1
      a
      RPOP
      RET
      END
    
```

設問 1 プログラム 1, 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。

a に関する解答群

- | | |
|-------------|------------|
| ア JMI LOOP | イ JNZ LOOP |
| ウ JUMP LOOP | エ JZE LOOP |

bに関する解答群

- | | | | | | |
|---|-----|------------|---|----|------------|
| ア | AND | GR3,=#0001 | イ | OR | GR3,=#0001 |
| ウ | AND | GR3,=#1000 | エ | OR | GR3,=#1000 |
| オ | AND | GR3,=#FFFF | カ | OR | GR3,=#FFFE |

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。

主プログラムが GR1 に #0555 を格納して、プログラム1及びプログラム2を呼んだ場合、プログラム1の命令 α の実行回数よりもプログラム2の命令 β の実行回数の方が c 回少なくなる。

プログラム1の命令 α の実行回数とプログラム2の命令 β の実行回数が等しくなるのは、主プログラムが GR1 に格納したデータのビット番号 d のビットの値が e のときである。

解答群

- | | | | | | | | | | |
|---|---|---|---|---|----|---|----|---|----|
| ア | 0 | イ | 1 | ウ | 5 | エ | 6 | オ | 7 |
| カ | 8 | キ | 9 | ク | 10 | ケ | 14 | コ | 15 |

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 1～3 に答えよ。

[プログラムの説明]

ある大学における成績出力用プログラムである。

- (1) この大学では、学期終了時に学生の履修科目成績一覧を図 1 に示すレコード様式でファイルに出力する。学生数は 15,000、科目数は 2,000 であり、得点は 0～100 の整数値である。

学生キー	学生氏名	科目名	得点	科目名	得点	科目名	得点	
(例) 02498	基本五郎	情報処理 2	25	化学実験	50	...	英会話基礎 1	67

図 1 履修科目成績一覧のレコード様式

- (2) 出力に必要なデータは、図 2 に示す 3 種類のファイルに記述されている。

科目キー	科目名	学生キー	学生氏名	科目キー	学生キー	得点
0000	情報処理 1	00000	設計花子	1932	00472	34
0001	情報処理 2	00001	情報太郎	1932	09755	79
0002	数値解析	00002	開発次郎	0357	10253	41
⋮	⋮	⋮	⋮	⋮	⋮	⋮
0752	化学実験	02498	基本五郎	0752	02498	50
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1997	英会話基礎 3	14997	試験守	0173	13712	13
1998	英会話基礎 4	14998	単元三郎	0173	00638	78
1999	英語応用	14999	言語大輔	0173	03581	61

科目ファイル course.txt 学生ファイル student.txt 成績情報ファイル record.txt

図 2 読み込みファイル群の例

(3) プログラムは、はじめに関数 `init` を呼ぶことで、図2に示した3種類のファイルから図3に示すようなリスト構造をメモリ上に構成する。科目キー `courseKey` の科目名は、`char` 型の配列 `courseName[courseKey]` に格納される。また、学生キー `studentKey` の学生情報は `STUDENT` 型の構造体 `student[studentKey]` で表現され、メンバ `studentName` には学生氏名が、メンバ `rFirstCourse` にはその学生の成績情報を表すリストの先頭へのポインタが格納される。学生の各履修科目についての成績情報は `RECORD` 型の構造体で表現し、メンバ `score` には得点、メンバ `courseName` には科目名へのポインタ、メンバ `rNextCourse` には他の履修科目の成績情報へのポインタが格納される。

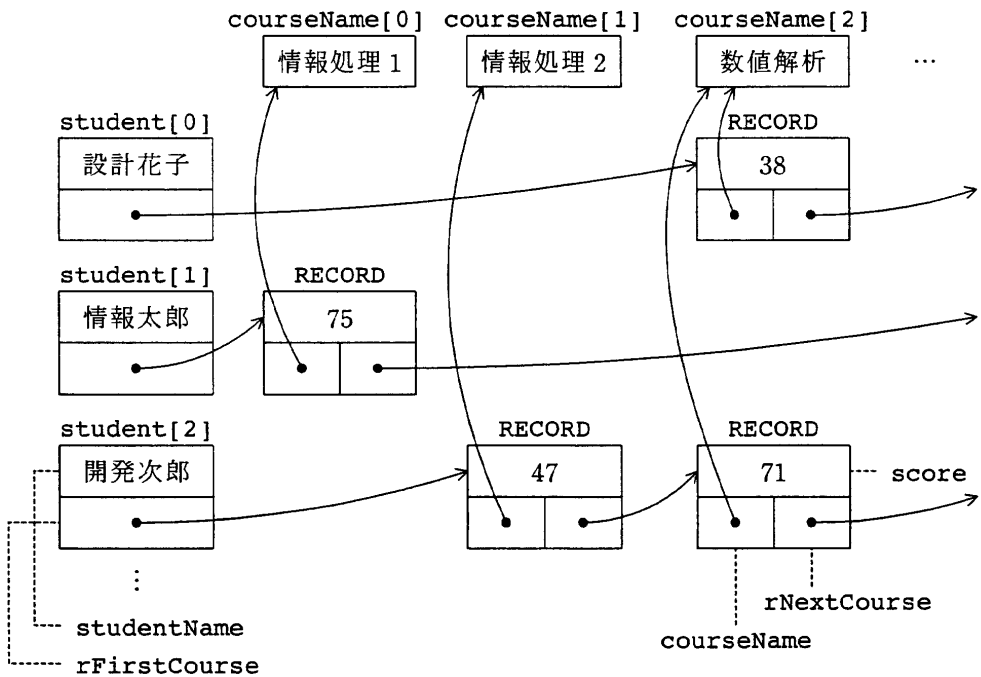


図3 データ構造例

(4) プログラム中で定義されているその他の関数の説明は次のとおりである。

```
void regist(int courseKey, int studentKey, short s);
```

機能：新たに RECORD 型の構造体を生成し、メンバ score に得点 s を代入する。また、この得点が学生キー studentKey の科目キー courseKey についてのものであるという情報を設定する。

```
void writeCourses();
```

機能：全学生分の履修科目成績一覧をファイルに出力する。

(5) このプログラムでは、次のライブラリ関数を用いる。

```
void *malloc(size_t size)
```

機能：size バイトの領域をメモリ上に割り付け、割り付けられた領域へのポインタを返す。メモリ割付けに失敗した場合は NULL を返す。

[プログラム]

```
#include <stdio.h>
#include <stdlib.h>
#define NUM_STUDENT 15000 /* 学生数 */
#define NUM_COURSE 2000 /* 科目数 */
#define MAX_WORD_LENGTH 21

struct RECORD {
    /* 成績情報 */
    char *courseName; /* 科目名へのポインタ */
    short score; /* 得点 */
    struct RECORD *pNextCourse;
    /* 次の履修科目の成績情報へのポインタ */
};

struct STUDENT {
    /* 学生情報 */
    char studentName[MAX_WORD_LENGTH];
    /* 学生氏名 */
    struct RECORD *pFirstCourse;
    /* 一つ目の履修科目の成績情報へのポインタ */
} student[NUM_STUDENT];

char courseName[NUM_COURSE][MAX_WORD_LENGTH];
/* 科目名の配列 */

void init();
void regist(int, int, short);
void writeCourses();
```

```

void init(){
    FILE *fStudent = fopen("student.txt", "r");
    FILE *fCourse = fopen("course.txt", "r");
    FILE *fRecord = fopen("record.txt", "r");
    int studentKey, courseKey, score;
    while(fscanf(fCourse, "%d", &courseKey) != EOF){
        fscanf(fCourse, "%s", courseName[courseKey]);
    }
    while(fscanf(fStudent, "%d", &studentKey) != EOF){
        fscanf(fStudent, "%s", student[studentKey].studentName);
        student[studentKey].rFirstCourse = NULL;
    }
    while(fscanf(fRecord, "%d %d %d", &courseKey, &studentKey,
        &score) != EOF){
        regist(courseKey, studentKey, (short)score);
    }
    fclose(fStudent);
    fclose(fCourse);
    fclose(fRecord);
}

void regist(int courseKey, int studentKey, short s){
    struct RECORD *p; ← α
    if ((p = (struct RECORD *)malloc(sizeof(struct RECORD)))
        == NULL){
        printf("メモリエラーです\n");
        exit(-1);
    }
    p->rNextCourse = student[studentKey].rFirstCourse;
    student[studentKey].rFirstCourse = a; ← β
    p->courseName = b;
    p->score = s;
}

void writeCourses(){
    FILE *fOutput = fopen("output.txt", "w");
    struct RECORD *p;
    int i;
    for (i = 0; i < NUM_STUDENT; i++){
        fprintf(fOutput, "%05d %-24s ", i,
            student[i].studentName);
        p = student[i].rFirstCourse;
        while(p != NULL){
            fprintf(fOutput, "%-24s %3d ", c );
            p = p->rNextCourse;
        }
        fprintf(fOutput, "\n");
    }
    fclose(fOutput);
}

```


設問3 次の記述中の に入れる正しい答えを、解答群の中から選べ。

図1に例示した履修科目成績一覧において、得点の降順に出力されるように、関数 `regist` の α , β を次のとおりに変更した。

なお、 a には正しい答えが既に入っているものとする。

処置	プログラムの変更内容
α を置換	<pre>struct RECORD *p; struct RECORD *q;</pre>
β を置換	<pre>q = student[studentKey].rFirstCourse; if (<input type="text"/> d <input type="text"/>) { p->rNextCourse = q; student[studentKey].rFirstCourse = <input type="text"/> a <input type="text"/> ; } else { while(<input type="text"/> e <input type="text"/>){ q = q->rNextCourse; } p->rNextCourse = q->rNextCourse; q->rNextCourse = <input type="text"/> a <input type="text"/> ; }</pre>

解答群

- ア `q != NULL && q->score < s`
- イ `q != NULL && q->score > s`
- ウ `q == NULL || q->score < s`
- エ `q == NULL || q->score > s`
- オ `q->rNextCourse != NULL && q->rNextCourse->score < s`
- カ `q->rNextCourse != NULL && q->rNextCourse->score > s`
- キ `q->rNextCourse == NULL || q->rNextCourse->score < s`
- ク `q->rNextCourse == NULL || q->rNextCourse->score > s`

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, 2 に答えよ。

[プログラムの説明]

渡された会員 ID を基に、会員ファイルから会員情報を検索し、画面データとして返却する呼ばれるプログラム（以下、副プログラムという）である。

(1) 呼ぶプログラム（以下、主プログラムという）は、会員 ID を読み取り、検索結果を画面に表示する。主プログラムが副プログラムへ渡す情報は、次のとおりである。現在の年月は、YYYYMM の形式である。

会員 ID 6 けた	現在の年月 6 けた
---------------	---------------

(2) 副プログラムでは、主プログラムから受け取った会員 ID で会員ファイルを検索し、見つかった会員情報を画面データに格納して主プログラムに返す。画面データに格納する情報は、見つかった会員のレコード項目のうち会員 ID 以外のすべての項目である。このとき、新規入会サービスの対象者であるかどうかを調べ、フラグをセットする。新規入会サービスの対象者は、特別会員及び一般会員のうち、会員期間が、入会月を基準に月単位で数えて、12 か月以内の会員である。例えば、10 月入会の場合は、翌年の 9 月までがサービス対象期間となる。

(3) 会員ファイルのレコード様式は、次のとおりである。

会員 ID 6 けた	氏名 20 けた	区分 1 けた	入会年月日 8 けた	その他の会員情報 205 けた
---------------	-------------	------------	---------------	--------------------

① 会員 ID をキーとした索引ファイルであり、会員 ID は数字 0～9 の組合せからなる。

② 区分には会員区分が記録されている。会員区分には、特別会員、一般会員、学生会員があり、次のとおりコード化されている。

1：特別会員 2：一般会員 3：学生会員

③ 入会年月日には、その会員の入会した年月日が YYYYMMDD の形式で記録されている。

(4) 画面データ (L-SCREEN) のデータ様式は、次のとおりである。

フラグ1	フラグ2	会員情報
1けた	1けた	234けた

フラグ1及びフラグ2は、次の決定表に従って値が決まる。

フラグ設定の決定表					
条件部	会員ファイルが検索できた	Y	Y	Y	N
	入会1年未満である	Y	Y	N	-
	区分が学生会員である	Y	N	-	-
動作部	a	X	X	X	-
		-	-	-	X
		X	-	X	X
		-	X	-	-

注 条件部での“Y”は条件が真，“N”は条件が偽，“-”は真偽に関係ないことを表す。動作部での“X”は条件がすべて満たされたとき、その行で指定した動作の実行を表し，“-”は動作を実行しないことを表す。

[プログラム]

(行番号)

```

1 DATA DIVISION.
2 FILE SECTION.
3 FD MEMBER-F.
4 01 MEMBER-R.
5     03 M-ID          PIC X(6).
6     03 M-DATA.
7         05 M-NAME    PIC X(20).
8         05 M-CLASS   PIC X(1).
9         05 M-DATE    PIC X(8).
10        05           PIC X(205).
11 WORKING-STORAGE SECTION.
12 

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


13
```

```

14 LINKAGE SECTION.
15 01 L-ID          PIC X(6).
16 01 L-YEAR       PIC 9(6).
17 01 L-SCREEN.
18     03 S-F1     PIC X(1).
19     03 S-F2     PIC X(1).
20     03 S-DATA   PIC X(234).
21 PROCEDURE DIVISION USING L-ID, L-YEAR, L-SCREEN.
22 MAIN-RTN.
23     OPEN INPUT MEMBER-F.
24     MOVE SPACE TO L-SCREEN.
25     MOVE L-ID TO M-ID.
26     READ MEMBER-F INVALID
27         MOVE "1" TO S-F1
28     NOT INVALID
29         MOVE M-DATA TO S-DATA
30     PERFORM CHECK-RTN
31     END-READ.
32     CLOSE MEMBER-F.
33     EXIT PROGRAM.
34 CHECK-RTN.
35     MOVE M-DATE TO W-YEAR1.
36     IF L-YEAR - W-YEAR2 < C THEN
37         IF M-CLASS = "1" OR "2" THEN
38             MOVE "1" TO S-F2
39         END-IF
40     END-IF.

```

設問1 決定表及びプログラム中の に入れる正しい答えを、解答群の中から選べ。ここで、決定表のフラグ1、フラグ2は、プログラムの S-F1、S-F2 に対応している。

aに関する解答群

ア	フラグ1に“1”を転記
	フラグ1に空白を転記
	フラグ2に“1”を転記
	フラグ2に空白を転記

イ	フラグ1に“1”を転記
	フラグ2に“1”を転記
	フラグ1に空白を転記
	フラグ2に空白を転記

ウ	フラグ1に空白を転記
	フラグ2に空白を転記
	フラグ1に“1”を転記
	フラグ2に“1”を転記

エ	フラグ1に空白を転記
	フラグ1に“1”を転記
	フラグ2に空白を転記
	フラグ2に“1”を転記

bに関する解答群

```
ア 01 W-YEAR1 PIC X(6).
   01 W-YEAR2 REDEFINES W-YEAR1 PIC 9(6).

イ 01 W-YEAR1 PIC 9(6).
   01 W-YEAR2 REDEFINES W-YEAR1 PIC X(6).

ウ 01 W-YEAR1 PIC X(6).
   01 W-YEAR2 PIC 9(6).
```

cに関する解答群

ア 0	イ 1	ウ 12
エ 99	オ 100	カ 101

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。

このシステムでは、会員IDの最後の1文字だけの入力ミスが非常に多いことが経験上分かっている。そこで、主プログラムから渡された会員IDと一致する会員情報が見つからなかったとき、会員IDの最後の1文字だけが異なるすべての会員の会員IDと氏名を修正候補として主プログラムに戻すように、次のとおり変更した。

処置	プログラムの変更内容
行番号 5 を置換	03 M-ID. 05 M-NUM PIC 9 OCCURS 6.
行番号 13 と 14 の間に追加	01 X PIC 9(2). 01 Y PIC 9(2).
行番号 15 を置換	01 L-ID. 03 L-NUM PIC 9 OCCURS 6.
行番号 20 を置換	03 S-DATA. 05 S-CORR OCCURS 9. 07 S-ID PIC X(6). 07 S-NAME PIC X(20).
行番号 27 と 28 の間に追加	PERFORM CORR-RTN
行番号 40 の後に追加	CORR-RTN. MOVE 1 TO Y. PERFORM VARYING X FROM 0 BY 1 UNTIL d IF X NOT = L-NUM(6) THEN e READ MEMBER-F INVALID CONTINUE NOT INVALID MOVE M-ID TO S-ID(f) MOVE M-NAME TO S-NAME(f) COMPUTE Y = Y + 1 END-READ END-IF END-PERFORM.

dに関する解答群

- ア X = 6 イ X > 6 ウ X = 9 エ X > 9 オ X > 10

eに関する解答群

- | | |
|---|---|
| ア MOVE L-ID TO M-ID | イ MOVE L-ID TO M-ID
MOVE X TO M-ID(6:) |
| ウ MOVE L-ID TO M-ID
MOVE X TO M-NUM(6) | エ MOVE L-ID TO M-ID
MOVE X TO M-NUM(X) |
| オ MOVE X TO M-NUM(6) | カ MOVE X TO M-NUM(X) |

fに関する解答群

- ア X イ X + 1 ウ Y エ Y + 1

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。

〔プログラムの説明〕

Aを起点、Dを終点とする四つの駅A, B, C, Dからなる路線の各駅に設置された自動改札機の処理を模したプログラムである。この路線では片道切符（以下、切符という）とプリペイドカード（以下、カードという）の2種類の乗車券が使用できる。

路線の運賃は駅間の距離で決められる。距離が4 km までは120 円（この運賃を初乗り運賃という）で、それを超えると2 km ごとに30 円加算される。2 km 未満は2 km に切り上げる。例えば、距離が7 km のとき、運賃は180 円である。

切符には、乗車駅で自動改札機を通過して入場するとき、乗車駅の情報記録される。降車駅で自動改札機を通過して出場するとき、運賃が計算され、金額が不足しているときはゲートが閉じられ、出場できない。一度使用した切符は無効となる。この路線では切符を発券した駅にかかわらず、どの駅の自動改札機からでも入場し、乗車できる。例えば、A 駅で発券された切符でB 駅の自動改札機から入場できる。

カードには、乗車駅で自動改札機を通過して入場するとき、乗車駅の情報記録される。このとき、カードの残高が0 円の場合は、ゲートが閉じられ、入場できない。降車駅で自動改札機を通過して出場するとき、精算処理が行われる。すなわち、運賃が計算され、カードの残高から引かれる。このとき、カードの残高が運賃に満たない場合は、ゲートが閉じられ、出場できない。

クラス `Line` は路線を表す。メソッド `getFare` は与えられた距離から運賃を計算して返す。

クラス `Gate` は、自動改札機を表す。クラス `Line` のフィールド `A`, `B`, `C`, `D` は `Gate` のインスタンスであり、それぞれ `A`, `B`, `C`, `D` の各駅に設置された自動改札機を表す。コンストラクタ及び各メソッドは、次の処理を行う。

- (1) コンストラクタは、`Gate` のインスタンスを生成する。最初の引数に駅名、2 番目の引数に路線の起点である `A` 駅からの距離を指定する。
- (2) メソッド `enter` は、自動改札機を通過して入場するときの処理を行う。乗車券が適正でない場合はゲートを閉じる。入場処理が正常に行われた場合は、乗車券に乗車駅の情報記録する。
- (3) メソッド `exit` は、自動改札機を通過して出場するときの処理を行う。乗車券の金

額（残高）が不足するなど、適正でない場合はゲートを閉じる。

(4) メソッド `open` 及び `close` は、それぞれゲートの開閉を表すメッセージを出力する。

抽象クラス `Ticket` は、この路線の乗車券を表し、このクラスを継承して切符やカードを定義する。コンストラクタ及び各メソッドは、次の処理を行う。

- (1) コンストラクタは、購入時の金額を初期値として乗車券に保持する。
- (2) メソッド `getValue` は、呼び出された時点での乗車券の金額（残高）を返す。
- (3) メソッド `adjustValue` は、必要であれば精算処理を行う。
- (4) メソッド `deduct` は、引数で指定された金額を乗車券の金額（残高）から差し引いて金額（残高）を更新する。
- (5) メソッド `setOrigin` は、指定された `Gate` を乗車駅として記録する。`null` が指定されたときは、乗車駅の記録を消去する。
- (6) メソッド `getOrigin` は、記録されている乗車駅を返す。記録されていないときは `null` を返す。

クラス `OneWayTicket` は切符を表し、クラス `PrepaidCard` はカードを表す。それぞれの乗車券の処理で、抽象メソッドを実装し、必要に応じて `Ticket` のメソッドをオーバーライドする。

[プログラム 1]

```
public final class Line {
    public static final Gate A = new Gate("A", 0);
    public static final Gate B = new Gate("B", 5);
    public static final Gate C = new Gate("C", 8);
    public static final Gate D = new Gate("D", 14);

    public static int getFare(int distance) {
        return 120 + (Math.max(distance - 3, 0) / 2) * 30;
    }
}
```

[プログラム 2]

```
public class Gate {
    private final String name;
    private final int distance;

    public Gate(String name, int distance) {
        this.name = name;
        this.distance = distance;
    }

    public void enter(Ticket ticket) {
        if (ticket.isValid() && ticket.getOrigin() == null) {
            a;
            open();
        } else {
            close();
        }
    }

    public void exit(Ticket ticket) {
        Gate origin = ticket.getOrigin();
        if (origin != null) {
            int d = Math.abs(origin.distance - distance);
            int fare = Line.getFare(d);
            if (b) {
                ticket.adjustValue(fare);
                ticket.setOrigin(null);
                open();
                return;
            }
        }
        close();
    }

    private void open() { System.out.println(name + ": open"); }
    private void close() {
        System.out.println(name + ": closed");
    }
}
```

[プログラム 3]

```
public abstract class Ticket {
    private Gate origin;
    private int value;

    public Ticket(int value) {
        this.value = value;
    }
}
```

```

public int getValue() { return value; }

public void deduct(int amount) { value -= amount; }

public void setOrigin(Gate gate) { origin = gate; }

public Gate getOrigin() { return origin; }

public abstract void adjustValue(int amount);

public abstract boolean isValid();
}

```

{プログラム 4}

```

public class OneWayTicket extends Ticket {
    private boolean valid = true;

    public OneWayTicket(int value) {
        c;
    }

    public void setOrigin(Gate gate) {
        super.setOrigin(gate);
        if (gate == null)
            valid = false;
    }

    public void adjustValue(int amount) { }

    public boolean isValid() { return valid; }
}

```

{プログラム 5}

```

public class PrepaidCard extends Ticket {
    public PrepaidCard(int value) {
        c;
    }

    public void adjustValue(int amount) { deduct(amount); }

    public boolean isValid() {
        return getValue() > 0;
    }
}

```

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 <code>ticket.setOrigin(Line.A)</code> | イ <code>ticket.setOrigin(Line.D)</code> |
| ウ <code>ticket.setOrigin(null)</code> | エ <code>ticket.setOrigin(this)</code> |
| オ <code>ticket.setOrigin(ticket)</code> | |

bに関する解答群

- | | |
|---|---|
| ア <code>ticket.getValue() < fare</code> | イ <code>ticket.getValue() <= fare</code> |
| ウ <code>ticket.getValue() == fare</code> | エ <code>ticket.getValue() > fare</code> |
| オ <code>ticket.getValue() >= fare</code> | |

cに関する解答群

- | | |
|-----------------------------|--|
| ア <code>super()</code> | イ <code>super(this)</code> |
| ウ <code>super(value)</code> | エ <code>super(); this.value = value</code> |
| オ <code>this(value)</code> | |

設問2 この路線で、新しいタイプの乗車券を発売することになった。この乗車券は発売した時刻から24時間以内は全駅で乗り降り自由な乗車券である。発売から24時間経過すると出場はできるが入場はできなくなる。これに伴い、抽象クラス `Ticket` を継承して新しいクラスを定義し、クラス `Gate` には修正を加えずにこの乗車券をサポートしたい。このクラスのコンストラクタ及びメソッドの処理を次の表にまとめた。表中の に入れる正しい答えを、解答群の中から選べ。ここで、解答群中の `value` は、クラス `Ticket` のフィールド `value` であり、コンストラクタで初期値を設定し、値はメソッド `getValue` で得るものとする。

コンストラクタ及びメソッド	処理
コンストラクタ	d
メソッド <code>getValue</code>	e
メソッド <code>setOrigin</code>	スーパークラスで定義されたとおり
メソッド <code>getOrigin</code>	スーパークラスで定義されたとおり
メソッド <code>adjustValue</code>	何もしない（メソッド本体が文を含まない）。
メソッド <code>isValid</code>	f

解答群

- ア `value` が路線の最高運賃（全運賃の最大値）以上のときだけ `true` を返す。それ以外は `false` を返す。
- イ `value` が路線の初乗り運賃以上のときだけ `true` を返す。それ以外は `false` を返す。
- ウ `value` の初期値に、理論的に 24 時間かかっても使い切れない金額を設定する。
- エ `value` の初期値に、路線の最高運賃（全運賃の最大値）を設定し、発券時刻を記録する。
- オ スーパークラスで定義されたとおり
- カ 常に 0 を返す。
- キ 常に路線の初乗り運賃の値を返す。
- ク 何もしない（メソッド本体が文を含まない）。
- ケ メソッド `deduct` を、`amount` を引数として呼び出す。
- コ 呼び出されたときの時刻がインスタンスに格納されている発券時刻から 24 時間以内のときだけ `true` を返す。それ以外は `false` を返す。

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1～3に答えよ。

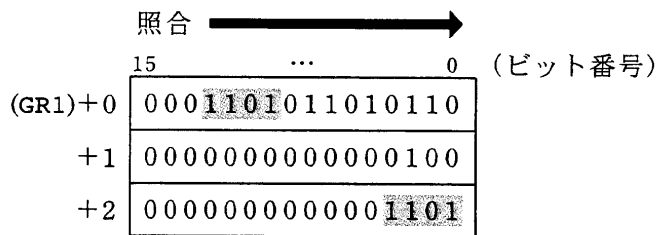
〔プログラムの説明〕

1語の中から指定されたビットパターンを検索する副プログラム BPSRH である。

- (1) 主プログラムは、パラメタ領域の先頭番地を GR1 に設定して、BPSRH を呼ぶ。
 パラメタの形式は次のとおりである。

(GR1)+0	検索対象語	
+1	n	n: ビットパターンのビット数
+2	ビットパターン (右詰め)	(1 ≤ n ≤ 16)

- (2) BPSRH は、検索対象語の上位ビットからビットパターンと照合し、最初に一致した部分の最上位のビット番号を GR0 に設定して主プログラムに戻る。一致した部分がない場合は、-1 を GR0 に設定して主プログラムに戻る。次の例では、GR0 には、一致した部分の最上位のビット番号 12 が設定される。



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

{プログラム}

(行番号)

```
1  BPSRH  START
2      RPUSH
3      LD   GR6,1,GR1
4      LAD  GR7,16
5      SUBA GR7,GR6 ; GR7 ← (16 - n)
6      LD   GR2,2,GR1
7      SLL  GR2,0,GR7 ; ビットパターンを左詰めに
8      LAD  GR4,-1
9      a ; マスクパターンの生成
10     LAD  GR0,-1 ; 戻り値の初期化
11     LAD  GR3,0 ; 照合位置ポインタの初期化
12     LD   GR5,0,GR1
13  LOOP LD   GR6,GR5 ; GR6 は作業用として使用
14     AND  GR6,GR4
15     CPL  GR6,GR2 ; ビットパターンとの照合
16     JZE  FIND
17     LAD  GR3,1,GR3 ; 次の照合位置を設定
18     CPA  GR3,GR7 ; 未照合部分が n ビット以上あるか?
19     JPL  EXIT
20     b
21     JUMP LOOP
22  FIND LAD  GR0,15 ; ビット番号の算出
23     SUBA GR0,GR3
24  EXIT RPOP
25     RET
26     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 に関する解答群

- | | | | | | |
|---|-----|-----------|---|-----|-----------|
| ア | SLL | GR4,0,GR6 | イ | SLL | GR4,0,GR7 |
| ウ | SRA | GR4,0,GR6 | エ | SRA | GR4,0,GR7 |
| オ | SRL | GR4,0,GR6 | カ | SRL | GR4,0,GR7 |

b に関する解答群

- | | | | | | |
|---|-----|-----------|---|-----|-----------|
| ア | LD | GR5,1,GR2 | イ | SLL | GR3,1 |
| ウ | SLL | GR5,1 | エ | SRL | GR3,1 |
| オ | SRL | GR5,0,GR2 | カ | SRL | GR5,0,GR3 |
| キ | SRL | GR5,1 | | | |

設問2 次のパラメタが渡され、ラベル FIND に制御が移ったときの、GR5 の値の16進表記として正しい答えを、解答群の中から選べ。

(GR1)+0	0001101011010110
+1	00000000000000100
+2	00000000000001101

解答群

- | | | |
|--------|--------|--------|
| ア 000D | イ 1AD6 | ウ AD60 |
| エ D000 | オ D6B0 | |

設問3 1の連続するビットパターンに特化した検索を行う副プログラム BP1SRH を使用して、映画館の指定席を予約する副プログラム RESERVE を作成した。

RESERVE 中の に入れる正しい答えを、解答群の中から選べ。

- (1) 映画館の指定席は1,024席あり、座席番号は0～1023である。また、指定席は座席番号順に16席ごとにグループ化されている。指定席管理表は連続する64語からなり、先頭の語のビット番号15が座席番号0の状態を、末尾の語のビット番号0が座席番号1023の状態を表す。指定席管理表の対応するビットが1のとき空席を、0のとき予約済を表す。

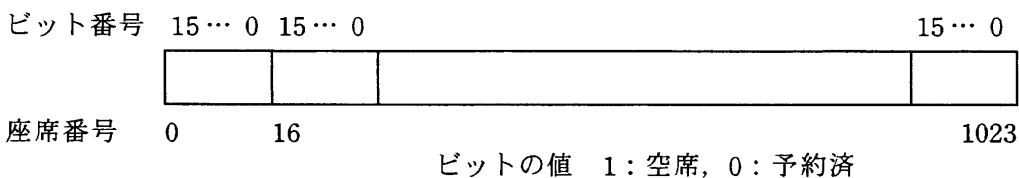


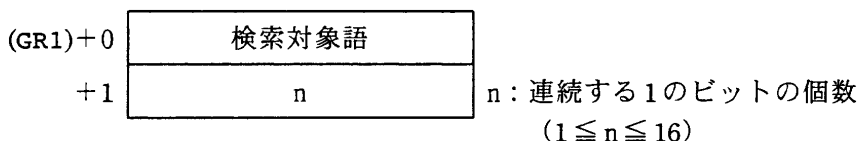
図 指定席管理表 (64 語) の形式

- (2) 主プログラムは、予約すべき席数 n ($1 \leq n \leq 16$) を GR1 に、指定席管理表の開始アドレスを GR2 に設定して、RESERVE を呼ぶ。
- (3) RESERVE は、指定された数の席を連続して、かつグループをまたがらないで確

保する。座席番号 0 から順に空席を探し、見つければ“予約済”の状態にして、確保された一番若い番号の座席番号を GR0 に設定し、主プログラムに戻る。確保できなければ、-1 を GR0 に設定して主プログラムに戻る。

(4) RESERVE から戻るとき、汎用レジスタ GR1 ~ GR7 の内容は元に戻る。

(5) BP1SRH に与えるパラメタは、次の形式とする。



BP1SRH は、BPSRH の行番号 6 ~ 9 を次の三つの命令で置き換えたプログラムである。

```
LAD GR2,#8000
SRA GR2,-1,GR6
LD GR4,GR2
```

[プログラム]

```
RESERVE START
RPU SH
LD GR6,GR1 ; n の保存
LD GR1,PARAM
ST GR6,1,GR1 ; BP1SRH 呼出しパラメタ準備 (1)
ST GR2,TBLADD ; 指定席管理表の開始アドレス保存
LD GR4,64,GR2
LD GR0,-1 ; 戻り値の初期化
LOOP CPL GR2,GR4 ; 検索終了?
JZE EXIT
LD GR5,0,GR2 ; 指定席管理表から 1 語取出し
ST GR5,0,GR1 ; BP1SRH 呼出しパラメタ準備 (2)
CALL BP1SRH ; 1 語中の空席を検索
CPA GR0,=-1
JNZ FIND
LD GR2,1,GR2 ; 次の語を検索へ
JUMP LOOP
FIND LAD GR3,15
SUBA GR3,GR0 ; GR3 ← (15 - GR0)
LD GR7,#8000
SRA GR7,-1,GR6
C
XOR GR7,#FFFF ; GR7 ← 1110000111111111 (GR0=12, n=4 の場合)
```

```

AND    GR5,GR7    ; 予約済に設定
ST     GR5,0,GR2
SUBL   GR2,TBLADD ; 座席番号の算出
      d
ADDA   GR2,GR3
LD     GR0,GR2
EXIT   RPOP
      RET
TBLADD DS    1
PARAM  DS    2    ; BP1SRH 呼出し用パラメタ領域
END

```

cに関する解答群

ア	AND	GR5,GR3	イ	AND	GR5,GR7
ウ	OR	GR5,GR3	エ	OR	GR5,GR7
オ	SRA	GR7,0,GR3	カ	SRL	GR7,0,GR3

dに関する解答群

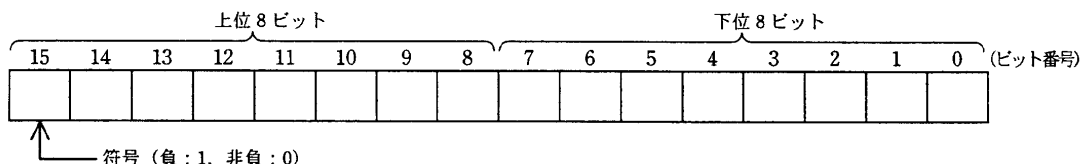
ア	SLL	GR2,1	イ	SLL	GR2,2
ウ	SLL	GR2,4	エ	SLL	GR3,1
オ	SLL	GR3,2	カ	SLL	GR3,4

■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	LD	$r1, r2$	$r1 \leftarrow (r2)$	○*1
Load		$r, adr [, x]$	$r \leftarrow (\text{実効アドレス})$	
ストア	ST	$r, adr [, x]$	実効アドレス $\leftarrow (r)$	-
STore				
ロードアドレス	LAD	$r, adr [, x]$	$r \leftarrow \text{実効アドレス}$	
Load Address				

(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	$r1, r2$ $r, \text{adr} [,x]$	$r1 \leftarrow (r1) \text{ AND } (r2)$ $r \leftarrow (r) \text{ AND } (\text{実効アドレス})$	○*1
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	$r1, r2$ $r, \text{adr} [,x]$	(r1) と (r2), 又は (r) と (実効アドレス) の算術比較又は論理比較を行い, 比較結果によって, FR に次の値を設定する。	○*1		
論理比較 ComPare Logical	CPL	$r1, r2$ $r, \text{adr} [,x]$	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				

(4) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	$r, \text{adr} [,x]$	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。	○*2
算術右シフト Shift Right Arithmetic	SRA	$r, \text{adr} [,x]$	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。	
論理左シフト Shift Left Logical	SLL	$r, \text{adr} [,x]$	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。	
論理右シフト Shift Right Logical	SRL	$r, \text{adr} [,x]$	シフトの結果, 空いたビット位置には 0 が入る。	

(5) 分岐命令

正分岐 Jump on Plus	JPL	$\text{adr} [,x]$	FR の値によって, 実効アドレスに分岐する。分岐しないときは, 次の命令に進む。	—																												
負分岐 Jump on Minus	JMI	$\text{adr} [,x]$	<table border="1"> <thead> <tr> <th>命令</th> <th colspan="3">分岐するときの FR の値</th> </tr> <tr> <td></td> <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																															
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) - _L 1, (SP) ← 実効アドレス	—
ポップ POP	POP r	r ← (SP), SP ← (SP) + _L 1	

(7) コール, リターン命令

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

(8) その他

スーパーバイザコール 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。
 +_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 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 (≥ 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 ($0000 \leq h \leq FFFF$)。
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 (≥ 0) が 2 進数で格納される。ファイルの終わり (end of file) を検出した場合は、-1 が格納される。IN 命令を実行すると、GR の内容は保存されるが、FR の内容は不定となる。

(2)

OUT	出力領域, 出力文字長領域
-----	---------------

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

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

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

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

(3)

RPUSH	
-------	--

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

(4)

RPOP	
------	--

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

2.5 機械語命令

機械語命令のオペランドは、次の形式で記述する。

r, r1, r2 GR は、記号 GR0 ~ GR7 で指定する。

x 指標レジスタとして用いる GR は、記号 GR1 ~ GR7 で指定する。

adr アドレスは、10 進定数、16 進定数、アドレス定数又はリテラルで指定する。

リテラルは、一つの 10 進定数、16 進定数又は文字定数の前に等号 (=) を付けて記述する。CASL II は、等号の後の定数をオペランドとする DC 命令を生成し、そのアドレスを adr の値とする。

2.6 その他

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

3. プログラム実行の手引

3.1 OS

プログラムの実行に関して、次の取決めがある。

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

10. 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、選択欄の問題番号の(選)をマークしてください。マークがない場合は、採点の対象になりません。

[問6と問10を選択した場合の例]

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

(6) 解答は、次の例題にならって、解答欄にマークしてください。

[例題] 次の に入れる正しい答えを、解答群の中から選べ。

秋の情報処理技術者試験は、 a 月に実施される。

解答群

ア 8 イ 9 ウ 10 エ 11

正しい答えは“ウ 10”ですから、次のようにマークしてください。

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

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、白紙であっても提出してください。
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。

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。