

平成 16 年度 春期

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

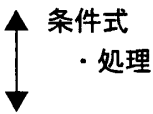
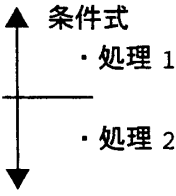
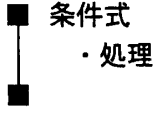
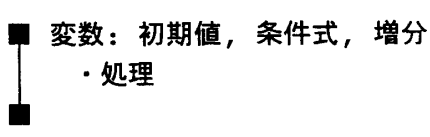
6. 問題に関する質問にはお答えできません。文意どおり解釈してください。
7. 問題冊子の余白などは、適宜利用して構いませんが、どのページも切り離さないでください。
8. アセンブラ言語の仕様は、この冊子の末尾を参照してください。
9. 電卓は、使用できません。

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、この記述形式が適用されているものとする。

〔記述形式〕

記述形式	説明
○	手続、変数などの名前、型などを宣言する。
・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。
/* 文 */	文に注釈を記述する。
 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、処理を繰返し実行する。
 変数: 初期値, 条件式, 増分 ・処理	繰返し処理を示す。 開始時点で変数に初期値 (式で与えられる場合がある) が格納され、条件式が真の間、処理を繰り返す。また、繰り返すごとに、変数に増分 (式で与えられる場合がある) を加える。

〔演算子〕

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	

〔論理型の定数〕

true, false

注 整数同士の除算では、商を結果として返す。%演算子は、剰余算を表す。

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 2進数の加算に関する次の記述を読んで、設問1, 2に答えよ。

2けた以上の2進数を1けたずつ加算するとき、下位けたからの繰り上がりを考慮すると、1けたの2進数三つを加算する処理（以下、全加算処理という）が必要になる。表は、全加算処理における入力と出力の組合せを示す。

例えば、表中の  $(c, x, y, c', s) = (1, 0, 1, 1, 0)$  の組合せは、次の計算を表す。

$$\begin{array}{r}
 1 \leftarrow \text{下位けたからの繰り上がり } c \\
 0 \leftarrow \text{入力された1けたの2進数 } x \\
 + 1 \leftarrow \text{入力された1けたの2進数 } y \\
 \hline
 10 \\
 \uparrow \uparrow \\
 \text{加算結果の1けた目 } s \\
 \text{加算結果の繰り上がり } c'
 \end{array}$$

表 全加算処理の入出力の組合せ

入力			出力	
c	x	y	c'	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

設問1 2けた以上の2進数に対して、下位けたから1けたずつ全加算処理を行う様子を状態遷移図で示すと、図1のように書くことができる。図1では、入力xとy、出力sの関係を、“/”を使って“xy/s”と表している。また、図1中の $q_0$ 及び $q_1$ は、それぞれ、加算処理によって繰り上がりの起こらなかった状態と、繰り上がりの起こった状態を表している。初期状態は $q_0$ である。

例えば、“01 + 01”の計算を図1を使って考えると、次のようになる。

- (1) 1けた目の加算“1 + 1”を行うとき、下位けたからの繰り上がりがないので、cは0であり、状態は $q_0$ である。この状態において、入力x, yは1, 1なので、関係“11/0”が適合し、出力sは0となる。また、このときc'は1と

なり、状態は  $q_0$  から  $q_1$  へ移る。

- (2)  $q_1$  の状態で、2 けた目の加算 “0 + 0” を行う。このときの入力  $x, y$  は 0, 0 であり、 $c$  は 1 なので、関係 “00/1” が適合し、出力  $s$  は 1 となる。また、このとき  $c'$  は 0 となり、状態は  $q_1$  から  $q_0$  へ移る。

図 1 の状態遷移図を使って、“0011 + 0101” の計算を行ったとき、状態は  $q_0$  と  $q_1$  をどのように遷移するか。正しい答えを、解答群の中から選べ。

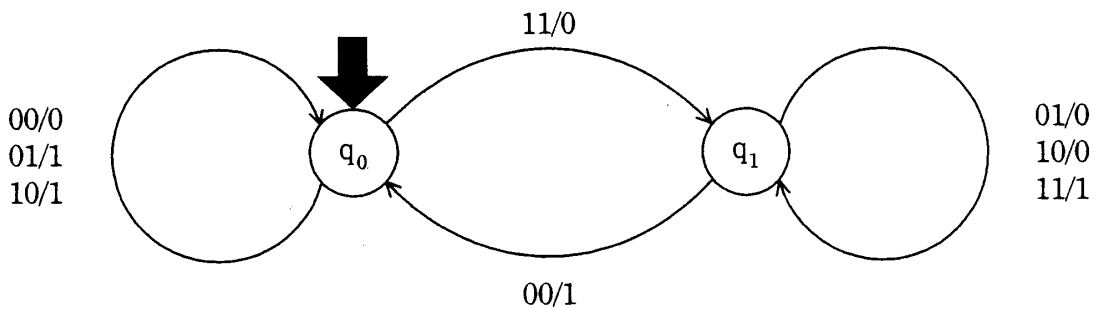


図 1 全加算処理の状態遷移図

解答群

- |   |   |
|---|---|
| ア $q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$ | イ $q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_1 \rightarrow q_0$ |
| ウ $q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_0 \rightarrow q_0$ | エ $q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_0$ |
| オ $q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$ |   |

設問 2 次のプログラムの説明を読んで、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。

〔プログラムの説明〕

図 1 で示した状態遷移図を基に、16 ビットの加算処理を行うプログラムである。ここで、図 2 に示すように、16 ビットの 2 進数を格納するための 16 個の要素をもつ論理型の配列（添字は 0 から始まる） $x, y, s$  を用いる。プログラム中の論理型の変数及び配列では、1 を true, 0 を false として扱う。また、プログラム中の xor は、排他的論理和演算を表している。

プログラムの実行後には、 $x$  と  $y$  の加算結果の下位 16 ビット分が  $s$  に格納される。

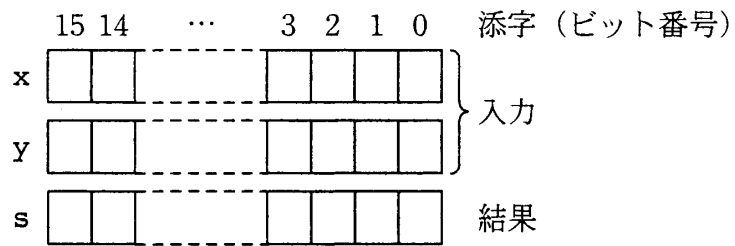


図2 配列 x, y, s の説明

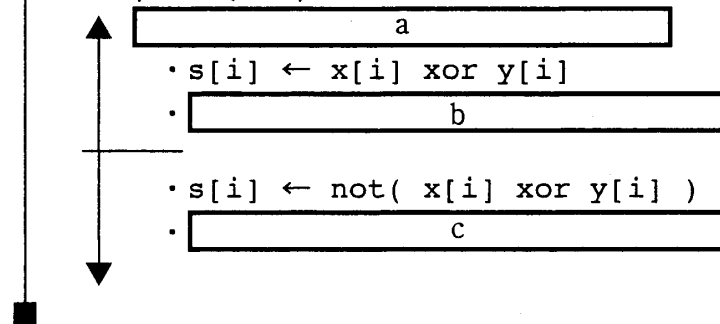
[プログラム]

○ 論理型:  $x[16], y[16], s[16], sw$

○ 整数型:  $i$

・  $sw \leftarrow 0$

■  $i: 0, i < 16, 1$



解答群

ア  $sw = 0$

イ  $sw = 1$

ウ  $sw = s[i]$

エ  $sw \leftarrow 0$

オ  $sw \leftarrow 1$

カ  $sw \leftarrow s[i]$

キ  $sw \leftarrow x[i] \text{ and } y[i]$

ク  $sw \leftarrow x[i] \text{ or } y[i]$

問2 データベースの障害回復機能に関する次の記述を読んで、設問1～4に答えよ。

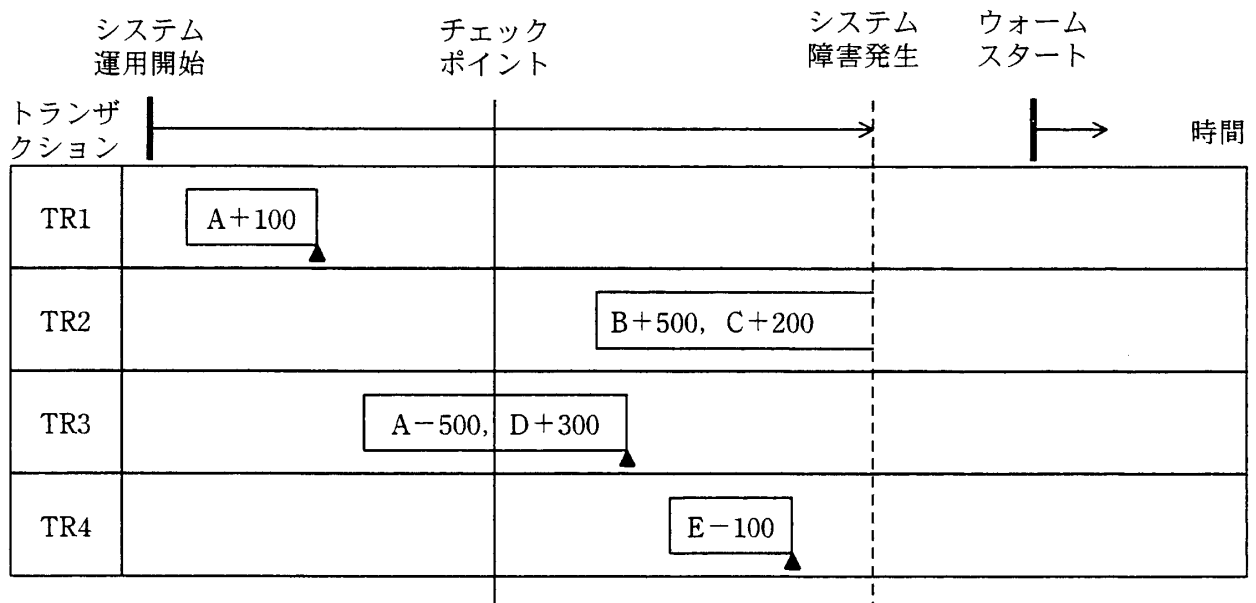
あるデータベースシステムでは、定期的にバックアップをとるとともに、更新前情報及び更新後情報をログファイルに記録している。システム障害が発生した場合は、ロールバック処理、ロールフォワード処理及び再処理を組み合わせたウォームスタートで再始動する運用を行っている。

このデータベースシステムでは、チェックポイントで、更新前ログ及び更新後ログへの情報書出しとデータベースへの書出しを一致させている。したがって、例えばロールバック処理は、ログファイル上の更新前情報を使って、データベースを直前のチェックポイントの時点まで戻すことによって、整合性を保つ。

今、表の値の状態でシステムの運用が開始され、図に示すように各トランザクションを実行していたとき、システム障害が発生したとする。

表 システム運用開始時の値

項目	値
A	1,000
B	1,500
C	3,000
D	2,000
E	500



注 A+100 は、その時点での A の値に 100 を加えることを表す。

▲は、コミットの発行を表す。

図 データベースシステムの運用状況

設問1 システム障害が発生した時点で、障害復旧の対象になるトランザクションとして正しい答えを、解答群の中から選べ。

解答群

ア TR1, TR2, TR3, TR4

イ TR2

ウ TR2, TR3, TR4

エ TR3, TR4

設問2 ウォームスタートによる再始動で回復を行うとき、ロールバックが完了した時点のA～Eの値として正しい答えを、解答群の中から選べ。ここで、ログファイルはすべて正常に記録されていたとする。

解答群

ア

項目	値
A	600
B	1,500
C	3,000
D	2,000
E	500

イ

項目	値
A	600
B	1,500
C	3,000
D	2,300
E	400

ウ

項目	値
A	1,000
B	1,500
C	3,000
D	2,000
E	500

エ

項目	値
A	1,100
B	1,500
C	3,000
D	2,000
E	500



設問3 ロールフォワード処理が終わった時点における A～E の値として正しい答えを、解答群の中から選べ。

解答群

ア

項目	値
A	600
B	1,500
C	3,000
D	2,300
E	400

イ

項目	値
A	600
B	2,000
C	3,200
D	2,300
E	400

ウ

項目	値
A	1,100
B	1,500
C	3,000
D	2,300
E	400

エ

項目	値
A	1,100
B	1,500
C	3,000
D	2,300
E	500

設問4 このデータベースシステムでは、障害回復のとき、ロールバック処理及びロールフォワード処理を組み合わせることによって、可能な限りトランザクションを再処理しないで済む復旧を行っている。ウォームスタート時、再処理をしなければならないトランザクションとして正しい答えを、解答群の中から選べ。

解答群

ア TR2

イ TR2, TR3

ウ TR2, TR3, TR4

エ TR4

問3 データ圧縮の方法の一つであるハフマン符号化に関する次の記述を読んで、設問1、2に答えよ。

ハフマン符号化は、圧縮対象の文字列を構成する文字の種類に注目し、その出現回数の偏りを利用した圧縮方法である。

今、図1に示す100文字からなる文字列の圧縮を考える。全角1文字を16ビット(2バイト)で表現する方式の場合、この文字列は、100文字なので、1,600ビットの長さである。ここで、この文字列を構成する文字の種類とその出現回数は、図2に示すとおりであるとする。

あんたがたどこさ△ひごさ△ひごどこさ△ ~ (中略) ~  
くってさ△それをこのはでちょっとかくせ

図1 対象となる文字列 (△は空白を示す)

文字の種類	△	さ	て	っ	く	こ	が	せ	た	と	ど	ま	ん
出現回数	12	11	6	5	4	4	3	3	3	3	3	3	3

う	ご	そ	で	に	は	ば	ひ	も	や	よ	れ
2	2	2	2	2	2	2	2	2	2	2	2

を	あ	い	お	か	き	し	ち	ぬ	の	ぽ	り
2	1	1	1	1	1	1	1	1	1	1	1

図2 文字の種類と出現回数

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。

対象となる文字列は、図2に示すように、37種類の文字から構成されている。37種類の文字を固定長のビット列を用いて識別するためには、1種類の文字に

- a ビットのビット列を対応させればよく、対象となる文字列は、  
a ビット/文字×100文字で計算される長さで表現できる。このとき、対

象となる文字列の長さは、表に示すように、全角1文字を16ビットで表現する方式の場合を100%とすると、b %となる。

表 文字の表現方式と対象文字列の長さ

表現の方式	1文字の長さ ビット	対象の文字列の長さ ビット	圧縮の割合 %
全角1文字を16ビットで表現	16	1,600	100
37種類の文字を固定長で表現	<span style="border: 1px solid black; padding: 2px;">a</span>	(省略)	<span style="border: 1px solid black; padding: 2px;">b</span>

aに関する解答群

ア 1            イ 6            ウ 7            エ 16            オ 20  
カ 33           キ 37           ク 47           ケ 100

bに関する解答群

ア 2.3            イ 6.0            ウ 6.3            エ 16.2            オ 16.7  
カ 37.0           キ 37.5           ク 43.2

[ハフマン符号化の手順]

文字の出現回数の偏りを利用したハフマン符号化を用いれば、この文字列を更に短いビット列で表現することができる。

この文字列をハフマン符号化する手順は、次のとおりである。

- (1) 文字列を構成する文字の種類を、その出現回数の多い順に並べ替える。同じ出現回数の場合は、文字コードの昇順とする。また、(2)で説明する仮想の文字の場合、出現回数と同じであるときの順序は、並べ替える直前の順序に従うものとする。
- (2) (1)の並べ替えの結果、最下位になった文字とその一つ上位の文字の2文字をグループ化し、これを仮想の1文字として扱い、最下位に位置付ける。この仮想の文字の出現回数は、グループ化した2文字の出現回数の和とする。
- (3) (2)の結果、すなわち、文字の種類が一つ減った状態のものに対し、(1)、(2)を、文字の種類が2種類になるまで繰り返す。
- (4) (3)で、最後に残った2種類の文字のそれぞれに、長さ1のビット列1又は0を

対応させる。

(5) (4) の 2 種類の文字から戻りながら各文字にビット列を対応させる。

(i) 1 段階戻ったときの 2 種類の文字に対しては、直前の段階のビット列の右端に 1 又は 0 を付加したビット列をそれぞれ対応させる。

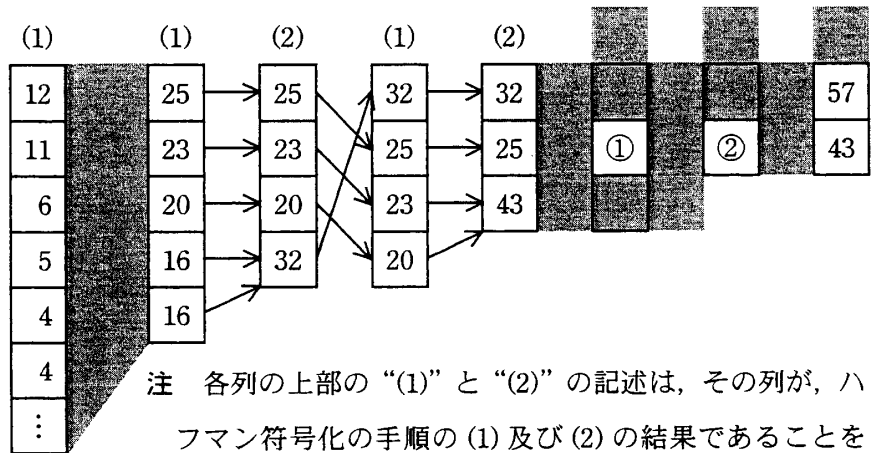
(ii) 元の 37 種類すべての文字にビット列が対応するまで (i) を行う。

(6) この 37 種類のビット列を用いて、対象となる文字列を表現する。

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。

(1)～(3) の実際の操作は、図 3 に示すとおりである。ここで、①は、 c であり、②は、 d である。最終的に各文字に対応するビット列は、図 4 に示すとおりとなり、100 文字からなる圧縮対象の文字列の長さは、全角 1 文字を 16 ビットで表現する方式の場合の約  e % のビット数で表現できることになる。

文字の種類	出現回数
△	12
さ	11
て	6
っ	5
く	4
こ	4
⋮	⋮



注 各列の上部の“(1)”と“(2)”の記述は、その列が、ハフマン符号化の手順の(1)及び(2)の結果であることを示す。また、の部分には表示を省略している。

図 3 実際の操作

このように、ハフマン符号化では、各文字に対応するビット列の長さが、その出現回数によって変わるが、符号化したビット列を先頭から見ていくことで、各文字に対応するビット列が一意に識別できるので、その復元は単純なアルゴリズムで実現できる。

文字の種類	△	さ	て	っ	く	こ	が
出現回数	12	11	6	5	4	4	3
対応ビット列	100	010	1010	0010	0000	11101	10110
ビット長	3	3	4	4	4	5	5
出現ビット長(注)	36	33	24	20	16	20	15
出現ビット累計	36	69	93	113	129	149	164

せ	た	と	ど	ま	ん	う	ご	そ	で
3	3	3	3	3	3	2	2	2	2
00111	00110	01101	01100	01111	01110	00010	101111	101110	110101
5	5	5	5	5	5	5	6	6	6
15	15	15	15	15	15	10	12	12	12
179	194	209	224	239	254	264	276	288	300

に	は	ば	ひ	も	や	よ	れ	を	あ
2	2	2	2	2	2	2	2	2	1
110100	110111	110110	110001	110000	110011	110010	111101	111100	000110
6	6	6	6	6	6	6	6	6	6
12	12	12	12	12	12	12	12	12	6
312	324	336	348	360	372	384	396	408	414

い	お	か	き	し	ち	ぬ	の	ぼ	り
1	1	1	1	1	1	1	1	1	1
0001111	0001110	1110001	1110000	1110011	1110010	1111101	1111100	1111111	1111110
7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7
421	428	435	442	449	456	463	470	477	484

注 出現ビット長は、ビット長×出現回数で計算する。

図4 文字の出現回数と対応するビット列に関する情報

c, dに関する解答群

ア 25            イ 32            ウ 43            エ 57            オ 100

eに関する解答群

ア 3.0            イ 7.0            ウ 18.8            エ 30.3            オ 43.8  
カ 50.0            キ 80.7            ク 85.7

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。

〔プログラムの説明〕

1次元配列に連続して格納されている $2^n$ 個の整数型データ ( $n$ は整数で $n > 0$ ) に対して、併合を $n$ 回繰り返すことによって整列を行う副プログラム `mergeSort` である。

- (1) 配列 `input[]` の要素を昇順に並べて配列 `output[]` に格納する。
- (2) 各配列の添字は0から始まる。
- (3) 副プログラム `mergeSort` の引数の仕様を表に示す。

表 `mergeSort` の引数の仕様

変数	型	入力/出力	意味
<code>input[]</code>	整数型	入力	整列するデータ
<code>output[]</code>	整数型	出力	整列結果を格納する領域
<code>size</code>	整数型	入力	配列の要素数

図は、1次元配列に格納されている8個のデータを3回の併合で整列する例である。

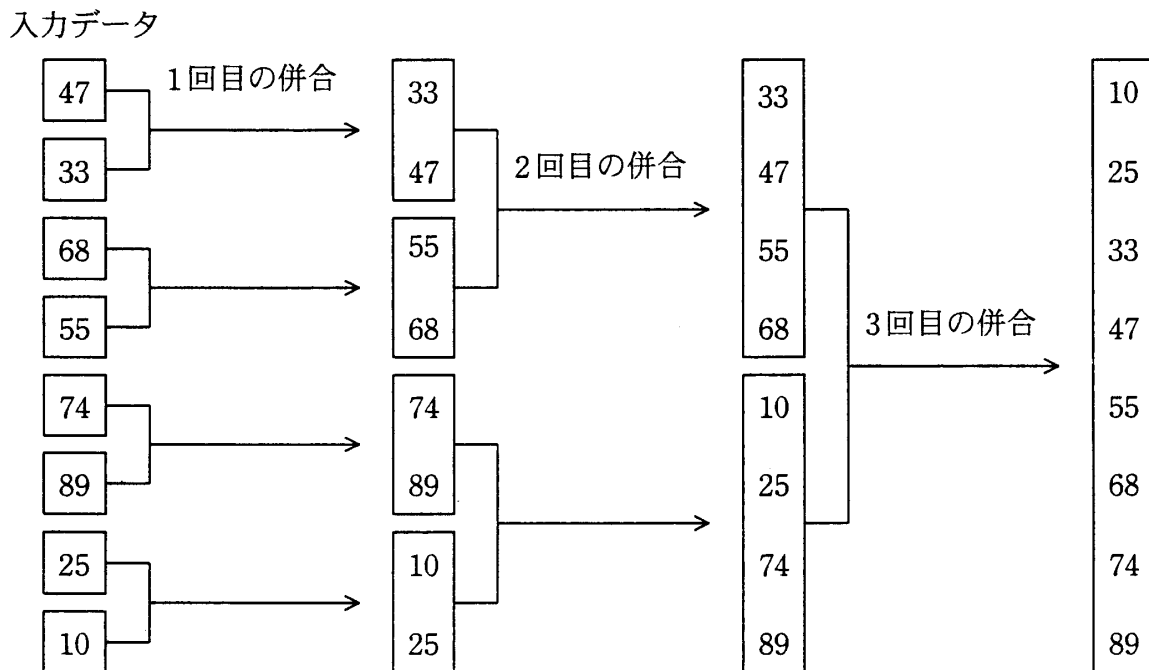
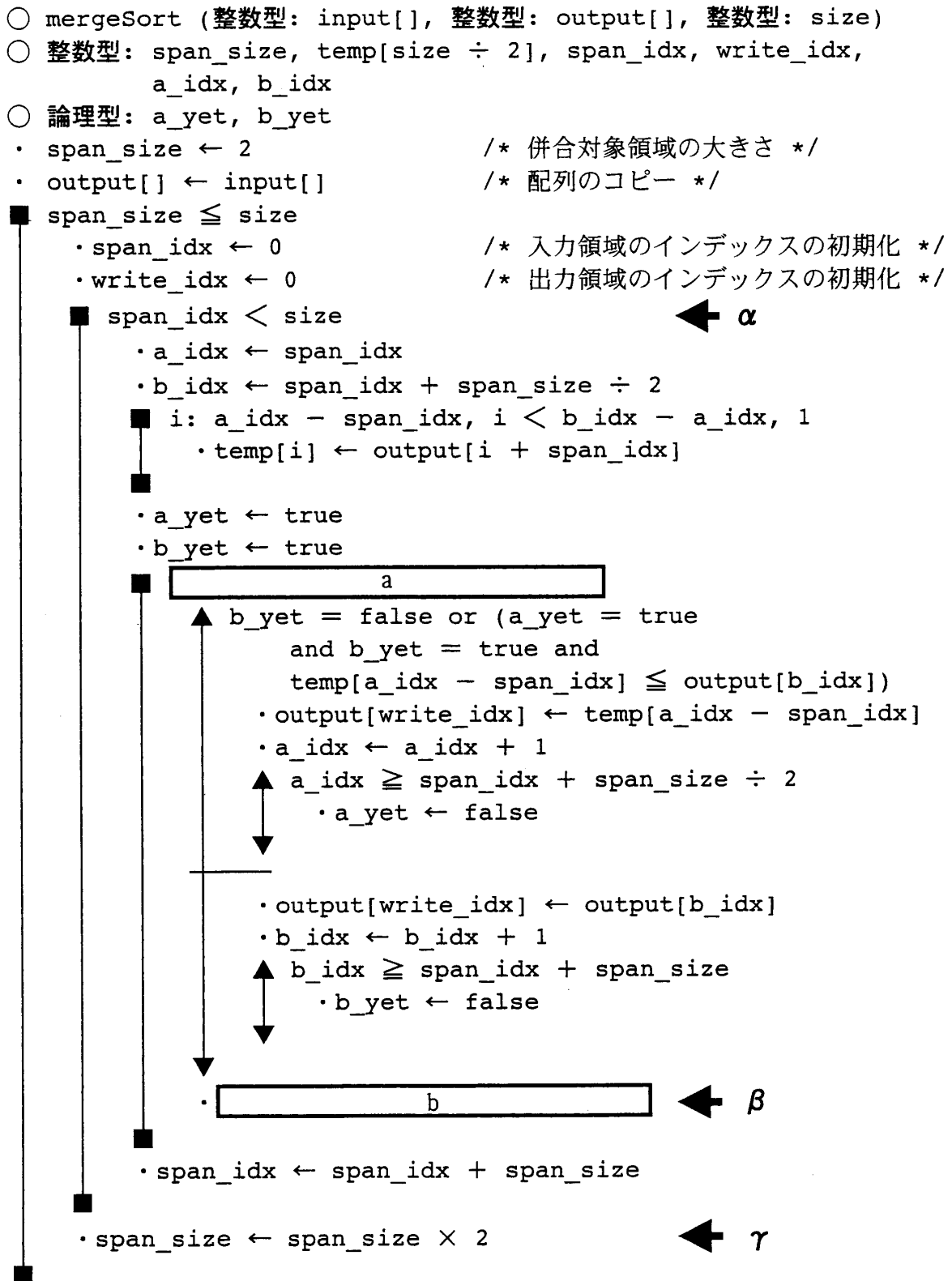


図 併合による整列の例

[プログラム]





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 `a_yet = false and b_yet = false`
- イ `a_yet = false or b_yet = false`
- ウ `a_yet = true and b_yet = true`
- エ `a_yet = true or b_yet = true`

bに関する解答群

- ア `b_idx ← span_idx`
- イ `b_idx ← span_idx + span_size`
- ウ `b_idx ← span_idx + span_size ÷ 2`
- エ `b_idx ← span_idx + span_size × 2`
- オ `write_idx ← 1`
- カ `write_idx ← a_idx + 1`
- キ `write_idx ← b_idx + 1`
- ク `write_idx ← write_idx + 1`

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。

次のデータを入力データとして、副プログラム `mergeSort` を実行すると、 c  回の併合の処理を実行するが、実際は  d  回目の併合が終了した時点でデータは昇順に並んでしまう。

副プログラム `mergeSort` をデータが昇順に並んだ時点で終了させるためには、次の変更1～3を加えればよい。ただし、変更後のプログラム中の変数 `ordered` は、論理型で宣言されているものとする。



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。

利用者の好みに合わせて国内ニュースを提供するシステムを設計する。このシステムの概要は、次のとおりである。

[システムの概要]

- (1) 提供するニュースは、ニュースファイルに次の様式で格納されている。ニュースファイルは、1件のニュースを1レコードとする順ファイルであり、ニュースが登録された順に記録されている。

ニュースファイルのレコード様式

登録年月日	登録時刻	発生年月日	発生時刻	分類	見出し	概要	詳細	画像ファイルのアドレス
-------	------	-------	------	----	-----	----	----	-------------

登録年月日は、ニュースが登録された年月日を表す8けたの文字列である。例えば、2004年4月1日の場合、“20040401”となる。登録時刻は、ニュースが登録された時分を表す4けたの文字列である。例えば、午後6時30分の場合、“1830”となる。発生年月日は、ニュースが発生した年月日であり、登録年月日と同様の様式である。発生時刻は、ニュースが発生した時刻であり、登録時刻と同様の様式である。分類は、ニュースの分類であり、“医療”、“教育”、“経済”、“芸能”、“サイエンス”、“社会”、“スポーツ”、“政治”のいずれかが格納される。更にニュースの見出し、概要、詳細及び関連する画像ファイルのアドレスが含まれる。

- (2) 利用者の情報は、利用者ファイルに次の様式で登録されている。利用者ファイルは、利用者IDをキーとする索引ファイルである。

利用者ファイルのレコード様式

利用者ID	分類1	分類2	分類3	分類4	分類5	前回利用年月日	前回利用時刻
-------	-----	-----	-----	-----	-----	---------	--------

利用者ファイルには、利用者の好みのニュースの分類が一つ以上五つまで分類1から順に登録されている。分類が五つに満たない場合、残りの分類には“NIL”

(空を示す)が格納される。システムが提供するニュースは、登録されている分類ごとに、発生年月日、発生時刻の新しいものから最大10件ずつである。

前回利用年月日と前回利用時刻は、利用者が最後にシステムを利用した日時である。前回利用年月日、前回利用時刻は、ニュースファイルの登録年月日、登録時刻とそれぞれ同じ様式の文字列である。

- (3) 利用者がシステムを利用する日時が、最後に利用した日時から24時間を超える場合、現在時刻の24時間前から登録されたすべてのニュースを対象に抽出を行う。利用者がシステムを利用する日時が、最後に利用した日時から24時間以内の場合、最後に利用した日時以降に登録されたニュースを対象に抽出を行う。
- (4) システムは、利用者の端末種別に基づき、提供する抽出結果の様式を変更する。利用者の端末が携帯電話などの場合、端末種別は“簡易”であり、ニュースの概要を提供する。パソコンなどの場合、端末種別は“詳細”であり、ニュースの詳細や画像を提供する。

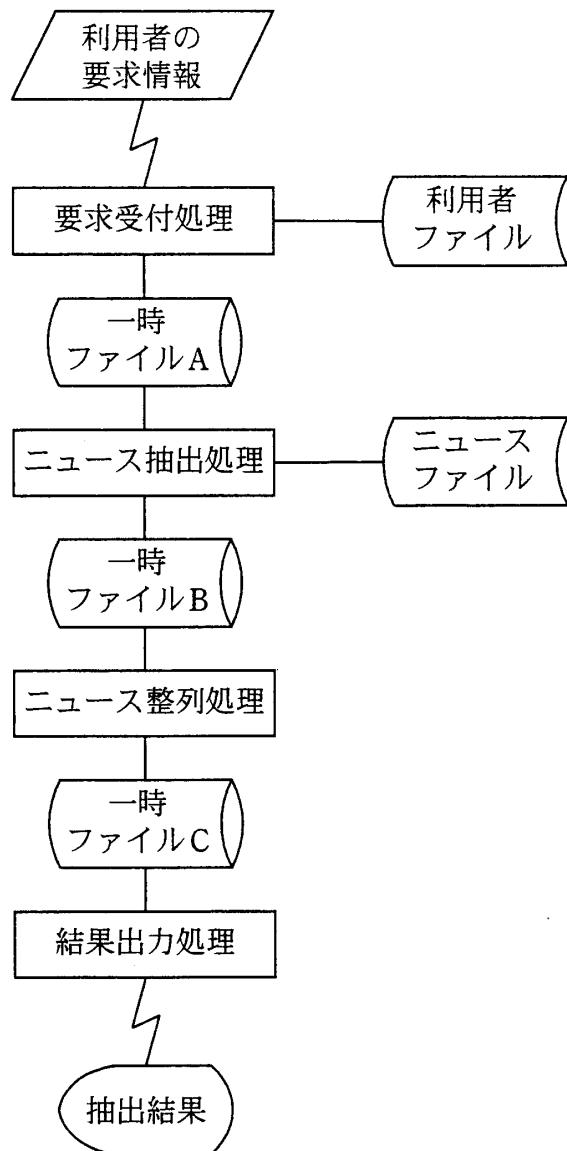


図 処理の流れ

[処理の説明]

(1) 要求受付処理では、次の様式の利用者の要求情報に含まれる利用者 ID を基に、利用者ファイルから利用者 ID の一致するレコードを選択する。

なお、様式中の端末種別には、“簡易”か“詳細”かの区分が格納される。

要求情報の様式

利用者 ID	端末種別
--------	------

さらに、要求情報と選択したレコードから、次の様式のレコードを一時ファイル A に書き出す。

一時ファイル A のレコード様式

分類 1	分類 2	分類 3	分類 4	分類 5	抽出開始 年月日	抽出開始 時刻	端末種別
------	------	------	------	------	-------------	------------	------

最後に、利用者ファイルの該当利用者レコードの前回利用年月日、前回利用時刻の値を、それぞれ現在の年月日、時刻に更新する。

(2) ニュース抽出処理では、一時ファイル A を基に、ニュースファイルから次の二つの条件を同時に満たすレコードを抽出する。

① ニュースファイルのレコードの登録年月日、登録時刻が、一時ファイル A の抽出開始年月日、抽出開始時刻以降である。

② ニュースファイルのレコードの分類が、一時ファイル A の分類 1 ～ 5 のどれかと一致する。

これらの条件を満たすレコードに、必要な情報を加えて一時ファイル B に書き出す。

(3) ニュース整列処理では、一時ファイル B を整列して、一時ファイル C へ書き出す。

(4) 結果出力処理では、一時ファイル C から分類ごとに最大 10 件を抽出し、その結果を、端末種別に従って次の様式で書き出す。

端末種別が“簡易”の場合の様式

発生年月日	発生時刻	分類	見出し	概要
-------	------	----	-----	----

端末種別が“詳細”の場合の様式

発生年月日	発生時刻	分類	見出し	詳細	画像ファイル のアドレス
-------	------	----	-----	----	-----------------

(5) ここで、各処理間の情報受渡しは、図に示すファイルだけで行われるものとする。



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。

ニュース整列処理では、ニュース抽出処理で出力されたレコードを対象に整列処理を行う。このとき、第1整列キーは  e  で文字コードの昇順とし、第2整列キーは  f  で降順、第3整列キーは  g  で降順とする。

解答群

- |        |               |
|--------|---------------|
| ア 概要   | イ 画像ファイルのアドレス |
| ウ 詳細   | エ 抽出開始時刻      |
| オ 登録時刻 | カ 登録年月日       |
| キ 発生時刻 | ク 発生年月日       |
| ケ 分類   |               |

設問4 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。

ある利用者の利用者ファイルのレコードが次のとおりであった。

DT512	政治	経済	スポーツ	NIL	NIL	20040401	0400
-------	----	----	------	-----	-----	----------	------

2004年4月1日の11時00分における、分類ごとのニュースの登録件数が次の表のとおりであったとする。この利用者が同年月日の同時刻にシステムを利用した場合、ニュース整列処理で一時ファイルCに書き出されるレコードの件数は  h  件であり、結果出力処理で表示されるニュースの件数は  i  件である。



表 ニュースの登録件数

分類	1時間ごとの登録件数										
	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00
	00:59	01:59	02:59	03:59	04:59	05:59	06:59	07:59	08:59	09:59	10:59
医療	0	0	0	0	0	0	2	0	1	0	0
教育	0	0	0	0	0	1	0	1	0	0	0
経済	0	0	0	1	0	2	2	6	8	2	4
芸能	1	0	1	0	1	2	0	1	2	1	1
サイエンス	1	0	1	0	1	1	3	2	3	2	5
社会	0	1	2	0	1	0	2	3	5	4	3
スポーツ	0	0	0	1	0	0	1	1	0	3	2
政治	0	1	0	0	0	1	4	2	3	6	3

解答群

ア 27

イ 28

ウ 50

エ 53

オ 60

カ 98

キ 108

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]

図に示すように、2次元平面の中に  $n$  個 ( $3 \leq n \leq 20$ ) の点を与えられたときに、これらの点すべてを含む円のうち、半径が最小である円の中心座標と半径を求めるプログラムである。

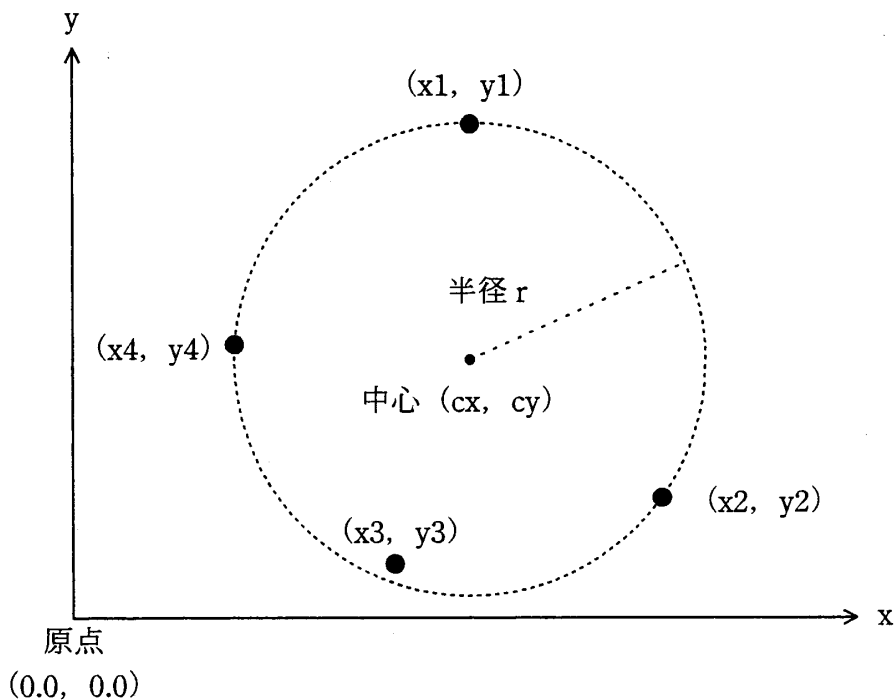


図 四つの点  $(x_1, y_1) \sim (x_4, y_4)$  を含む円

- (1) 各点の位置は、 $x$  座標と  $y$  座標で与えられる。各座標値はともに  $10^4$  を超えない正の値である。
- (2) ここでは、次のアルゴリズムによって、求める円の中心を探し出している。
  - ① 仮中心点を原点  $(0.0, 0.0)$  に、移動係数を  $0.5$  に設定する。
  - ②  $n$  個の点の中で、仮中心点から最も遠い位置にある点 (最遠点) の方向に向か

って、一定の距離（以下、移動距離と呼ぶ）だけ仮中心点を移動する。このときの移動距離は、移動前の仮中心点から最遠点までの距離に移動係数を乗じた値とする。これを一定回数（ここでは100回）繰り返した後に③へ移る。

なお、繰返しの途中で、仮中心点の移動距離が一定の値（ $10^{-9}$ ）以下になった場合には、移動前の仮中心点を円の中心座標とみなし、処理を終了する。

③ 移動係数を  $\frac{1}{2}$  倍にして、②へ戻る。

(3) 関数 `solvcircle` の仕様は、次のとおりである。

```
void solvcircle( double pt_x[], double pt_y[],
                double *cx, double *cy, double *r );
```

・機能：引数で受け取った点の座標値を基に、これらの点を内包する最小円の中心座標（x座標とy座標）と半径を求める。

・引数：`pt_x` 対象となる各点のx座標を格納した配列変数  
（終端には-1.0が格納されている。）

`pt_y` 対象となる各点のy座標を格納した配列変数  
（終端には-1.0が格納されている。）

`cx` 求まった円の中心点のx座標を格納する変数へのポインタ

`cy` 求まった円の中心点のy座標を格納する変数へのポインタ

`r` 求まった円の半径を格納する変数へのポインタ

(4) 関数 `distance` の仕様は、次のとおりである。

```
double distance( double x1, double y1,
                 double x2, double y2 );
```

・機能：引数で指定された2点間の距離を求める。

・引数：`x1, y1` 一方の点のx座標（x1）とy座標（y1）

`x2, y2` 他方の点のx座標（x2）とy座標（y2）

・返却値：2点（x1, y1）と（x2, y2）の距離

(5) 関数 `main` は、4個の点座標が与えられたときの関数 `solvcircle` の使用例を示している。

(6) プログラムでは、解を  $10^{-8}$  までの精度で求める。

[プログラム]

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

#define CONVRGV 1.0e-9
#define EOD -1.0
#define LOOPMAX 100
#define ON 1
#define OFF 0

void solvcircle( double[], double[], double*, double*, double* );
double distance( double, double, double, double );

main()
{
    double pos_x[] = { 10.0, 50.0, 50.0, 100.0, EOD };
    double pos_y[] = { 50.0, 10.0, 100.0, 50.0, EOD };
    double x, y, r;

    solvcircle( pos_x, pos_y, &x, &y, &r );
    printf( " 円の中心座標 ( %12.8f , %12.8f ) 半径 = %12.8f\n",
           x, y, r );
}

void solvcircle( double pt_x[], double pt_y[],
                double *cx, double *cy, double *r )
{
    double mvrate=0.5, maxdist, dist;
    double xsv=0.0, ysv=0.0;
    int k, i, t, cvgflg;

    *cx = *cy = 0.0; /* 仮中心点の初期座標 */
    cvgflg = OFF;
    while ( cvgflg == OFF ) {
        for ( t=0; t < LOOPMAX; t++ ) {
            maxdist = 0.0;
            for ( i=0; pt_x[i] != EOD; i++ ) {
                dist = distance( *cx, *cy, pt_x[i], pt_y[i] );
                if ( dist > maxdist ) {
                    maxdist = dist;
                    k = i;
                }
            }
            *cx += ( pt_x[k] - a ) * mvrate;
            *cy += ( pt_y[k] - b ) * mvrate;
            if ( c <= CONVRGV ) {
                *cx = xsv;
                *cy = ysv;
                cvgflg = ON;
                break;
            }
        }
    }
}
```

```

        xsv = *cx;
        ysv = *cy;
    }
    mvrate /= 2.0;
}
*r = ;
}

double distance( double x1, double y1, double x2, double y2 )
{
    return sqrt( (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) );
}

```

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, dに関する解答群

- ア \*cx
- イ \*cy
- ウ dist
- エ distance( pt\_x[0], pt\_y[0], \*cx, \*cy )
- オ distance( pt\_x[0], pt\_y[0], xsv, ysv )
- カ maxdist
- キ pt\_x[0]
- ク pt\_y[0]

cに関する解答群

- ア distance ( pt\_x[k], pt\_y[k], \*cx, \*cy )
- イ distance ( pt\_x[k], pt\_y[k], xsv, ysv )
- ウ distance ( xsv, ysv, \*cx, \*cy )
- エ maxdist
- オ maxdist - dist

問7 次の COBOL プログラムの説明及びプログラムを読んで、設問に答えよ。

〔プログラムの説明〕

ある会社では、社員が英語と情報処理技術についてどのような能力を保有しているかを、資格技能ファイルで管理している。年に一度、英語テストの最新の点数及び過去1年間に取得した情報処理技術の資格情報を資格技能ファイルに反映させる更新処理を行う。更新処理では、まず、英語テストの最新の点数を英語点数ファイルに記録し、資格情報を情報処理技術認定ファイルに記録する。その上で、旧資格技能ファイルに英語点数ファイルと情報処理技術認定ファイルとを突き合わせ、新資格技能ファイルを作成する。

(1) 新旧の資格技能ファイル N-SHIKAKUF, O-SHIKAKUF のレコード様式は、次のとおりである。

社員番号 6けた	英語の点数 3けた	情報処理技術資格情報			
		区分領域1 2けた	区分領域2 2けた	...	区分領域13 2けた

- ① 全社員のレコードが存在し、社員番号の昇順に整列された順ファイルである。
- ② 英語テストは入社時に全員受験しており、英語の点数は必ず記録されている。  
情報処理技術資格情報は、資格を取得している場合に記録され、取得していない場合は空白が記録されている。
- ③ 英語点数ファイルにデータがあれば、英語の点数を更新する。点数は0～999点の範囲である。
- ④ 情報処理技術資格は、全部で13区分あり、合格した区分を2けたのコードで区分領域1～13の先頭から記録する。区分が記録されていない区分領域は空白である。
- ⑤ 情報処理技術認定ファイルにデータがあれば、新資格技能ファイルの最初の空白の区分領域に追加する。

(2) 英語点数ファイル EIGOF のレコード様式は、次のとおりである。

社員番号 6 けた	英語の点数 3 けた
--------------	---------------

- ① 社員番号の昇順に整列された順ファイルである。
- ② 受験した社員のレコードだけが存在する。また、1年間に複数回受験していても、過去のデータを無条件で置き換えているので、最後に受験したときの点数を記録したレコードだけが存在する。
- ③ 旧資格技能ファイルに存在しない社員番号のレコードがあったときは、エラーメッセージを出力する。

(3) 情報処理技術認定ファイル JOHOF のレコード様式は、次のとおりである。

社員番号 6 けた	区分 2 けた
--------------	------------

- ① 社員番号の昇順に整列された順ファイルである。
- ② 1年間に新たに資格を取得した社員のレコードだけが存在する。また、1人につき複数レコードが存在することがあるが、区分は異なる。
- ③ 旧資格技能ファイルに存在しない社員番号のレコードがあったときは、エラーメッセージを出力する。
- ④ 旧資格技能ファイルに、同一区分が記録済みであったときは、エラーメッセージを出力する。

[プログラム]

```
DATA DIVISION.  
FILE SECTION.  
FD O-SHIKAKUF.  
01 O-R PIC X(35).  
FD N-SHIKAKUF.  
01 N-R PIC X(35).  
FD EIGOF.  
01 E-R.  
03 E-SBANGO PIC X(6).  
03 E-TEN PIC 9(3).
```

```

FD JOHOF.
01 J-R.
    03 J-SBANGO          PIC X(6).
    03 J-KUBUN           PIC X(2).
WORKING-STORAGE SECTION.
01 W-R.
    03 W-SBANGO          PIC X(6).
    03 W-EIGO            PIC 9(3).
    03 W-JOHO.
        05 W-KUBUN-A     PIC X(2) OCCURS 13 INDEXED BY N.
01 W-E-SBANGO           PIC X(6).
01 W-J-SBANGO           PIC X(6).

```

PROCEDURE DIVISION.

HAJIME.

```

OPEN INPUT O-SHIKAKUF EIGOF JOHOF OUTPUT N-SHIKAKUF.
PERFORM READ-O-SHIKAKUF.
PERFORM READ-EIGOF.
PERFORM READ-JOHOF.

```

PERFORM UNTIL

EVALUATE TRUE

WHEN W-SBANGO = W-E-SBANGO AND NOT = HIGH-VALUE

WHEN W-SBANGO < W-E-SBANGO

WHEN W-SBANGO > W-E-SBANGO

END-EVALUATE

EVALUATE TRUE

WHEN W-SBANGO = W-J-SBANGO AND NOT = HIGH-VALUE

PERFORM J-RTN

WHEN W-SBANGO < W-J-SBANGO

CONTINUE

WHEN W-SBANGO > W-J-SBANGO

DISPLAY "ERROR JOHO ", W-J-SBANGO

PERFORM READ-JOHOF

END-EVALUATE

IF W-SBANGO < W-E-SBANGO AND W-J-SBANGO

WRITE N-R FROM W-R

PERFORM READ-O-SHIKAKUF

END-IF

END-PERFORM.

CLOSE O-SHIKAKUF EIGOF JOHOF N-SHIKAKUF.

STOP RUN.

E-RTN.

MOVE E-TEN TO W-EIGO.

J-RTN.

PERFORM UNTIL

PERFORM VARYING N FROM 1 BY 1

UNTIL N > 13 OR J-KUBUN = W-KUBUN-A(N)

OR W-KUBUN-A(N) = SPACE

CONTINUE

END-PERFORM



```

        IF N < 14 AND W-KUBUN-A(N) = SPACE
            MOVE J-KUBUN TO W-KUBUN-A(N)
        ELSE
            DISPLAY "ERROR KUBUN ", J-R
        END-IF
        PERFORM READ-JOHOF
    END-PERFORM.
READ-O-SHIKAKUF.
    READ O-SHIKAKUF
        AT END MOVE HIGH-VALUE TO W-SBANGO
        NOT AT END MOVE O-R TO W-R
    END-READ.
READ-EIGOF.
    READ EIGOF
        AT END MOVE HIGH-VALUE TO W-E-SBANGO
        NOT AT END MOVE E-SBANGO TO W-E-SBANGO
    END-READ.
READ-JOHOF.
    READ JOHOF
        AT END MOVE HIGH-VALUE TO W-J-SBANGO
        NOT AT END MOVE J-SBANGO TO W-J-SBANGO
    END-READ.

```

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 W-SBANGO = HIGH-VALUE
- イ W-SBANGO = HIGH-VALUE AND W-E-SBANGO = HIGH-VALUE  
AND W-J-SBANGO = HIGH-VALUE
- ウ W-SBANGO = HIGH-VALUE OR W-E-SBANGO = HIGH-VALUE  
AND W-J-SBANGO = HIGH-VALUE
- エ W-SBANGO = HIGH-VALUE OR W-E-SBANGO = HIGH-VALUE  
OR W-J-SBANGO = HIGH-VALUE

b～dに関する解答群

- ア CONTINUE
- イ DISPLAY "ERROR EIGO " , W-E-SBANGO
- ウ DISPLAY "ERROR EIGO " , W-E-SBANGO  
PERFORM READ-EIGOF
- エ DISPLAY "ERROR EIGO " , W-E-SBANGO  
PERFORM READ-JOHOF
- オ PERFORM E-RTN
- カ PERFORM E-RTN  
PERFORM READ-EIGOF
- キ PERFORM E-RTN  
PERFORM READ-EIGOF  
PERFORM READ-JOHOF
- ク PERFORM E-RTN  
PERFORM READ-JOHOF
- ケ PERFORM READ-EIGOF
- コ PERFORM READ-EIGOF  
PERFORM READ-JOHOF

eに関する解答群

- ア W-J-SBANGO = E-SBANGO
- イ W-J-SBANGO = J-SBANGO
- ウ W-J-SBANGO = W-SBANGO
- エ W-J-SBANGO NOT = W-SBANGO

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]

このプログラムは、数量の単位変換を行う共通機能を提供するクラス群と、それらのテストプログラムからなる。テストプログラムでは、セルシウス温度（セ氏温度，°C）及びカ氏温度（°F）の変換を行うクラスを利用する。

- (1) インタフェース `Converter` は、単位変換を行うクラスが実装すべきインタフェースを定義する。
- (2) クラス `ConverterRunner` は、単位変換クラスを利用するプログラムからアクセスするためのクラスである。
  - ① メソッド `setConverter` は、単位変換クラスのインスタンスを設定する。
  - ② メソッド `run` は、単位変換クラスのメソッド `convert` を呼び出す。
- (3) クラス `CtoF` は、セ氏温度の値を、カ氏温度に変換する処理を実装するクラスである。
- (4) クラス `FtoC` は、カ氏温度の値を、セ氏温度に変換する処理を実装するクラスである。
- (5) クラス `TestConverter` は、単位変換クラスを利用するテストプログラムである。メソッド `main` は、実行すべき単位変換処理（1:カ氏→セ氏変換，2:セ氏→カ氏変換，q:終了）の選択を促してから、入力された値を指定された処理に従って変換する。実行例を図に示す。

```
% java TestConverter
Type 1(FtoC), 2(CtoF), or q(Quit): 1
Type input value: 68
20.0
```

図 クラス `TestConverter` の実行例

[プログラム 1]

```
public interface Converter {  
      
}
```

[プログラム 2]

```
public class ConverterRunner {  
    private  conv;  
  
    public void setConverter( conv) {  
        this.conv = conv;  
    }  
  
    public void run(String input) {  
        if (conv == null) return;  
        try {  
            double in = Double.parseDouble(input);  
            double out = conv.convert(in);  
            System.out.println(out);  
        } catch (NumberFormatException e) {  
            System.err.println("invalid input " + input);  
        }  
    }  
}
```

[プログラム 3]

```
public class CtoF  {  
    public double convert(double input) {  
        return 9.0 / 5.0 * input + 32.0;  
    }  
}
```

[プログラム 4]

```
public class FtoC  {  
    public double convert(double input) {  
        return 5.0 / 9.0 * (input - 32.0);  
    }  
}
```

[プログラム 5]

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class TestConverter {
    public static void main(String[] args) {
        // 標準入力から入力を受け取る reader を作成する
        BufferedReader reader =
            new BufferedReader(new InputStreamReader(System.in));
        ConverterRunner runner = new ConverterRunner();
        try {
            while (true) {
                System.out.print("Type 1(FtoC), 2(CtoF), " +
                    "or q(Quit): ");
                // 標準入力から 1 行読み込む
                String choice = reader.readLine();
                if ("1".equals(choice)) {
                    runner.setConverter(new FtoC());
                } else if ("2".equals(choice)) {
                    runner.setConverter(new CtoF());
                } else if ("q".equals(choice)) {
                    break;
                } else continue;

                System.out.print("Type input value: ");
                // 標準入力から 1 行読み込む
                String value = reader.readLine();
                d
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

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 abstract public void convert();
- イ public double convert(double input) { }
- ウ public double convert(double input);
- エ public double convert(String input);

bに関する解答群

- ア Converter
- イ ConverterRunner
- ウ CtoF
- エ FtoC

cに関する解答群

- ア extends Converter
- イ extends ConverterRunner
- ウ implements Converter
- エ implements ConverterRunner

dに関する解答群

- ア runner.conv.convert(Double.parseDouble(value));
- イ runner.conv.run(value);
- ウ runner.convert(Double.parseDouble(value));
- エ runner.run(value);

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]

偶数パリティを設定する副プログラム EPAR である。

(1) EPAR は、GR1 中の 1 であるビットの個数が全体として偶数となるように、GR1 のビット番号 15 にパリティビットを設定する。

すなわち、GR1 のビット番号 0 ~ 14 の中で 1 であるビットの個数が奇数ならば 1 を、偶数ならば 0 をビット番号 15 に設定する。

(2) 主プログラムは、GR1 のビット番号 0 ~ 14 にビットデータ列を、ビット番号 15 に 0 を設定して EPAR を呼ぶ。

(3) 副プログラムから戻るとき、GR2 の内容は元に戻す。

[プログラム]

```
EPAR  START
      PUSH  0,GR2
      ST    GR1,TMP
      LAD   GR2,0           ; カウンタの初期化
LP1   PUSH  0,GR1
      AND   GR1,=1
      JZE   NCNT
      ADDL  GR2,=1
NCNT  POP   GR1
      SRL  GR1,1
      JNZ  LP1
      LD   GR1,TMP
      
      JZE   FIN
      OR   GR1,=#8000      ; ← α
FIN   POP   GR2
      RET
TMP   DS   1
      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。

解答群

- |   |     |        |   |     |        |   |     |       |
|---|-----|--------|---|-----|--------|---|-----|-------|
| ア | AND | GR1,=1 | イ | AND | GR2,=1 | ウ | SLL | GR1,1 |
| エ | SLL | GR2,1  | オ | SRL | GR1,1  | カ | SRL | GR2,1 |

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。

誤って GR1 のビット番号 15 が 1 に設定されて主プログラムから呼ばれたとき、この副プログラムは正しい結果を返さない場合がある。GR1 に #A010 が設定されて呼ばれた場合、主プログラムに戻るときの GR1 の内容は  a  となる。

GR1 のビット番号 15 が 1 に設定されて主プログラムから呼ばれた場合でも、ビット番号 15 にパリティビットが正しく設定されるようにしたい。このためには、 $\alpha$  で示される 1 行を  b  と修正すればよい。

a に関する解答群

- |   |       |   |       |   |       |   |       |
|---|-------|---|-------|---|-------|---|-------|
| ア | #2010 | イ | #4020 | ウ | #5008 | エ | #A010 |
| オ | #C020 | カ | #D008 |   |       |   |       |

b に関する解答群

- |   |     |            |   |     |            |
|---|-----|------------|---|-----|------------|
| ア | AND | GR1,=#8000 | イ | OR  | GR1,=#4000 |
| ウ | XOR | GR1,=#8000 | エ | SLL | GR1,1      |
| オ | SRL | GR1,1      |   |     |            |



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 1 の説明]

C 言語の仕様で記述された空でないソースプログラムを標準入力から読み込んで、注釈を除去して標準出力へ出力するプログラムである。

(1) ソースプログラムの表記に関する説明

① このプログラムで扱う注釈は、文字定数、文字列リテラル又は注釈の中を除いて、“/\*” で始まり “\*/” で終わる文字の列である。

② 使用可能文字の種類は、JIS X 0201 (7 ビット及び 8 ビットの情報交換用符号化文字集合) である。ただし、“¥” は “\” と表記する。

③ 次のような記述はない。

(i) 注釈の入れ子記述

例 /\* aaaaa /\* bbbbbb \*/ ccccc \*/

(ii) 図形文字の 3 文字表記

??= ??( ??/ ??' ??< ??> ??) ??! ??-

④ 文法上の誤りはない。

(2) プログラム 1 では、次の手順で注釈除去を行っている。ただし、文字定数、文字列リテラル及び注釈の解析を単純に処理しているので、ソースプログラムの記述によっては、これらを正しく認識できずに誤った動作を起こす場合がある。

① 一重引用符又は二重引用符を検出すると、文字定数又は文字列リテラルの開始と解釈し、関数 `quote` を使用して、対応する一重引用符又は二重引用符を検出するまで、文字の列を読み込んでそのまま出力する。

② “/\*” を検出すると、注釈の開始と解釈し、最初に現れる “\*/” までの文字の列を読み飛ばす。

- (3) プログラム 1 による注釈除去の実行例を図に示す。

入力ソースプログラム

```
/* This program uses fgets to display
 * a line from a file on the screen. */
#include <stdio.h>
int main( void )
{
    FILE *stream; /* file pointer */
    char line[100]; /* input stream */

    if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
    {
        if( fgets( line, 100, stream ) == NULL)
            printf( "fgets error\n" ); /* error message */
        else
            printf( "%s", line);
        fclose( stream );
    }
}
```

注釈を除去した後の出力結果

```
#include <stdio.h>
int main( void )
{
    FILE *stream;
    char line[100];

    if( (stream = fopen( "crt_fgets.txt", "r" )) != NULL )
    {
        if( fgets( line, 100, stream ) == NULL)
            printf( "fgets error\n" );
        else
            printf( "%s", line);
        fclose( stream );
    }
}
```

図 注釈除去の実行例

[プログラム 1]

```
#include <stdio.h>
void quote( char );

main()
{
    int c1, c2;

    while ( (c1 = getchar()) != EOF ) {
        /* 一重引用符の検出 */
        if ( c1 == '\\' ) quote( '\\' );
        /* 二重引用符の検出 */
        else if ( c1 == '\"' ) quote( '\"' );
        /* 斜線の検出 */
        else if ( c1 == '/' ) {
            c2 = getchar();
            /* 次の文字がアスタリスクのとき */
            if ( c2 == '*' ) {
                /* 注釈文字列の除去 */
                while ( 1 ) {
                    while ( (c1 = getchar()) != '*' );
                    c2 = getchar();
                    if ( c2 == '/' ) break;
                }
            }
            /* その他の場合 */
            else {
                putchar(c1);
                putchar(c2);
            }
        }
        else putchar(c1); /* 読み込んだ1文字をそのまま出力*/
    }
}

void quote( char c )
{
    /* 文字定数及び文字列リテラルの抽出 */
    char cc;

    putchar(c);
    while ( (cc = getchar()) != c ) putchar(cc);
    putchar(cc);
}
}
```

設問1 プログラム1に入力すると誤った動作を起こす記述を，解答群の中から選べ。

解答群

```
ア /* "aaaaaaa" */  
イ /* aaa 'a' */  
ウ if ( c == '\\' ) {  
エ printf( " \'  " );  
オ printf( "aaa /* comment */ \n" );
```

〔プログラム2の説明〕

プログラム2は，プログラム1の説明(2)で指摘した問題点を解決するために，次のように書き換えたものである。

- (1) 文字定数，文字列リテラル，注釈の三つのモードに分けて処理する。
- (2) “文字定数モード”は，一重引用符の出現によって，モードのオンとオフが切り替わる。ただし，“\”による拡張表記として記述されたもの，文字列リテラル内及び注釈内に記述されたものは該当しない。
- (3) “文字列リテラルモード”は，二重引用符の出現によって，モードのオンとオフが切り替わる。ただし，“\”による拡張表記として記述されたもの，文字定数内及び注釈内に記述されたものは該当しない。
- (4) “注釈モード”は，“/\*”及び“\*/”の出現によって，モードのオンとオフが切り替わる。ただし，文字定数内及び文字列リテラル内に記述されたものは該当しない。

[プログラム 2]

```
#include <stdio.h>
main()
{
    int c1, c2;
    int c_mode = 0;          /* 注釈モードをオフに初期化      */
    int quotel = 0;         /* 文字定数モードをオフに初期化  */
    int quote2 = 0;        /* 文字列リテラルモードをオフに初期化 */

    for ( c1 = getchar(); ( c2 = getchar() ) != EOF; c1 = c2 ) {

        if ( !c_mode ) { /* 注釈モードがオフのとき      */
            /* 文字定数又は文字列リテラルの中で文字 \ を検出      */
            if (  && c1 == '\\ ' ) {
                putchar(c1);
                putchar(c2);
                c2 = getchar();
                continue;
            }
            /* 文字列リテラル以外のところで一重引用符を検出      */
            else if ( !quote2 && c1 == '\'' )
                ;
            /* 文字定数以外のところで二重引用符を検出      */
            else if ( !quotel && c1 == '\" ' )
                ;
            /* 文字定数及び文字列リテラル以外で / と * を検出      */
            else if (  && c1 == '/' && c2 == '*' ) {
                ;
                c2 = getchar();
                continue;
            }
            putchar(c1);
        }

        else {
            if ( c1 == '*' && c2 == '/' ) { /* 注釈の終端か? */
                ;
                c2 = getchar();
            }
        }
        putchar(c1);
    }
}
```

設問2 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, dに関する解答群

ア !quote1

イ !quote2

ウ (!quote1 || !quote2)

エ (!quote1 && !quote2)

オ (quote1 || quote2)

カ (quote1 && quote2)

b, c, eに関する解答群

ア c\_mode = !c\_mode

イ c\_mode = quote1 && quote2

ウ quote1 = !quote1

エ quote1 = !quote2

オ quote1 = quote2

カ quote2 = !quote1

キ quote2 = !quote2

ク quote2 = quote1

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

[プログラムの説明]

社員別業務別の1か月分の作業日報ファイルを読み込み、社員別業務別の作業時間月計ファイルと社員別の超過時間月計ファイルを作成するプログラムである。

(1) 作業日報ファイル NIPPO-FILE のレコード様式は、次のとおりである。

社員コード	作業日	作業日種別	業務コード	作業時間
6けた	2けた	1けた	3けた	2けた

- ① 作業日種別の値は、0 のときは平日出勤日を示し、1 のときは休日出勤日を示す。
  - ② 作業時間は、1 時間単位とする。
  - ③ 平日出勤日1日の作業時間の合計は8時間以上である。
  - ④ 平日出勤日1日の作業時間のうち8時間を超えた分と、休日出勤日の作業時間を超過時間とみなす。
  - ⑤ 各作業日の作業時間の合計は24時間以内である。
  - ⑥ 同一社員コードのレコードに現れる業務コードは、100種類以下である。
  - ⑦ 1人の社員が1日の間に複数の業務を行ったときは、行った業務ごとにレコードがある。
  - ⑧ 1人の社員が連続して複数日にまたがって同一業務を行ったときは、作業日ごとにレコードがある。
  - ⑨ 作業日報ファイルは、社員コード及び作業日の昇順に整列された順ファイルである。
  - ⑩ ファイルの内容は、すべて正しいものとする。
- (2) 作業時間月計ファイル SAGYO-FILE のレコード様式は、次のとおりである。

社員コード	業務コード	作業時間
6けた	3けた	3けた

社員ごとに業務ごとの作業時間を集計した順ファイルである。

(3) 超過時間月計ファイル CHOKA-FILE のレコード様式は、次のとおりである。

社員コード	超過時間
6けた	3けた

- ① 社員ごとに各作業日の超過時間を求め、1か月分を集計した順ファイルである。
  - ② 集計した超過時間が0の場合でも、超過時間月計ファイルに書き出す。
- (4) 処理の手順は、次のとおりである。
- ① 社員ごとに1日分の作業時間を求める。
  - ② 社員ごとに業務ごとの作業時間を集計する。
  - ③ 社員ごとに①で求めた1日分の作業時間のうち、平日出勤日のときは8時間を越えた分を超過時間とし、休日出勤日のときはすべての作業時間を超過時間として合計する。
  - ④ 社員ごとに②で求めた業務ごとの作業時間を、作業時間月計ファイルに書き出す。
  - ⑤ 社員ごとに③で求めた超過時間の合計を、超過時間月計ファイルに書き出す。
- (5) このプログラムは未完成で、幾つかの文を挿入したり条件式を修正したりする必要がある。

[プログラム]

(行番号)

```
1      DATA                DIVISION.
2      FILE                  SECTION.
3      FD  NIPPO-FILE.
4      01  NIPPO-REC.
5          03  N-SHAIN-CODE    PIC  X(06).
6          03  N-SAGYO-BI      PIC  9(02).
7          03  N-SHUBETSU      PIC  9(01).
8          03  N-GYOMU-CODE    PIC  X(03).
9          03  N-SAGYO-JIKAN   PIC  9(02).
10     FD  SAGYO-FILE.
11     01  SAGYO-REC.
12         03  S-SHAIN-CODE    PIC  X(06).
13         03  S-GYOMU-CODE    PIC  X(03).
14         03  S-SAGYO-JIKAN   PIC  9(03).
15     FD  CHOKA-FILE.
16     01  CHOKA-REC.
17         03  C-SHAIN-CODE    PIC  X(06).
```



```

18         03 C-CHOKA-JIKAN PIC 9(03).
19 WORKING-STORAGE SECTION.
20 01 EOF PIC X(01).
21 01 SHAIN-CODE PIC X(06).
22 01 SAGYO-BI PIC 9(02).
23 01 SHUBETSU PIC 9(01).
24 01 SAGYO-JIKAN-KEI PIC 9(03).
25 01 GYOMU-SU PIC 9(03).
26 01 GYOMU.
27 03 FILLER OCCURS 100 INDEXED BY N.
28 05 GYOMU-CODE PIC X(03).
29 05 SAGYO-JIKAN PIC 9(03).
30 PROCEDURE DIVISION.
31 MAIN-RTN.
32 OPEN INPUT NIPPO-FILE
33 OUTPUT CHOKA-FILE SAGYO-FILE.
34 MOVE SPACE TO EOF.
35 PERFORM READ-RTN.
36 PERFORM UNTIL EOF = "E"
37 INITIALIZE CHOKA-REC
38 INITIALIZE GYOMU
39 MOVE N-SHAIN-CODE TO SHAIN-CODE
40 PERFORM UNTIL EOF = "E" OR
41 N-SHAIN-CODE NOT = SHAIN-CODE
42 MOVE N-SAGYO-BI TO SAGYO-BI
43 MOVE N-SHUBETSU TO SHUBETSU
44 PERFORM UNTIL N-SAGYO-BI NOT = SAGYO-BI
45 COMPUTE SAGYO-JIKAN-KEI =
46 SAGYO-JIKAN-KEI + N-SAGYO-JIKAN
47 PERFORM VARYING N FROM 1 BY 1
48 UNTIL GYOMU-CODE(N) = N-GYOMU-CODE
49 CONTINUE
50 END-PERFORM
51 IF N > GYOMU-SU THEN
52 MOVE N-GYOMU-CODE TO GYOMU-CODE(N)
53 SET GYOMU-SU TO N
54 END-IF
55 COMPUTE SAGYO-JIKAN(N) =
56 SAGYO-JIKAN(N) + N-SAGYO-JIKAN
57 END-PERFORM
58 IF SHUBETSU = 0 THEN
59 COMPUTE C-CHOKA-JIKAN =
60 C-CHOKA-JIKAN + SAGYO-JIKAN-KEI - 8
61 ELSE
62 COMPUTE C-CHOKA-JIKAN =
63 C-CHOKA-JIKAN + SAGYO-JIKAN-KEI
64 END-IF
65 END-PERFORM
66 PERFORM WRITE-RTN
67 END-PERFORM.
68 CLOSE NIPPO-FILE CHOKA-FILE SAGYO-FILE.
69 STOP RUN.

```

```

70     READ-RTN.
71         READ NIPPO-FILE AT END
72             MOVE "E" TO EOF
73     END-READ.
74     WRITE-RTN.
75         MOVE SHAIN-CODE TO C-SHAIN-CODE S-SHAIN-CODE.
76         PERFORM VARYING N FROM 1 BY 1
77             UNTIL N = GYOMU-SU
78             MOVE GYOMU-CODE(N) TO S-GYOMU-CODE
79             MOVE SAGYO-JIKAN(N) TO S-SAGYO-JIKAN
80             WRITE SAGYO-REC
81     END-PERFORM.
82     WRITE CHOKA-REC.

```

設問 プログラムを完成させるためには、プログラム中の条件式を表1に示す三つの条件式で置き換えるとともに、表2に示す三つの文をプログラムに挿入する必要がある。表1、表2の  に入れる置換位置及び挿入位置を、解答群の中から選べ。ただし、挿入した場合でも、プログラムの行番号は付け直さないものとする。

表1 置き換える条件式

置換位置	条件式
<input type="text" value="a"/>	N > GYOMU-SU
<input type="text" value="b"/>	N > GYOMU-SU OR GYOMU-CODE(N) = N-GYOMU-CODE
<input type="text" value="c"/>	EOF = "E" OR N-SHAIN-CODE NOT = SHAIN-CODE OR N-SAGYO-BI NOT = SAGYO-BI

表2 挿入する文

挿入位置	文
<input type="text" value="d"/>	PERFORM READ-RTN
<input type="text" value="e"/>	MOVE ZERO TO SAGYO-JIKAN-KEI
<input type="text" value="f"/>	MOVE ZERO TO GYOMU-SU

a～cに関する解答群

ア 行番号 36

イ 行番号 40, 41

ウ 行番号 44

エ 行番号 48

オ 行番号 51

カ 行番号 58

キ 行番号 77

d～fに関する解答群

ア 行番号 36 と 37 の間

イ 行番号 41 と 42 の間

ウ 行番号 44 と 45 の間

エ 行番号 48 と 49 の間

オ 行番号 49 と 50 の間

カ 行番号 56 と 57 の間

キ 行番号 64 と 65 の間

ク 行番号 66 と 67 の間

ケ 行番号 77 と 78 の間

コ 行番号 80 と 81 の間

問12 次のJavaプログラムの説明及びプログラムを読んで、設問に答えよ。

[プログラムの説明]

インタフェース `CharIterator` は、データ構造に依存せずにそのインスタンスから文字 (`char`) を順番に取り出すための操作を定義する。`CharIterator` には、次のメソッドが定義されている。

```
public char next()
```

次の文字があればそれを返し、なければ

`java.util.NoSuchElementException` を投げる。例えば、この `CharIterator` インスタンスが文字を  $n$  個もっているとき、最初の呼出しで1番目の文字を返し、2回目の呼出しで2番目の文字を返し、…というように、 $n$  回目の呼出しまで文字を順番に返していく。 $n + 1$  回目以降の呼出しでは、`NoSuchElementException` を投げる。

なお、`NoSuchElementException` は `java.lang.RuntimeException` のサブクラスである。

```
public boolean hasNext()
```

次の文字があれば `true` を返し、なければ `false` を返す。

クラス `CharIteratorFactory` は、引数に指定したデータ型に一致した `CharIterator` を返すメソッドを定義する。

`CharIteratorFactory` には、次のクラスメソッドが定義されている。

```
public static CharIterator getCharIterator(String data)
```

引数に指定した `String` から文字を順番に取り出す `CharIterator` を返す。  
引数に `null` が指定されたときは、`NullPointerException` を投げる。

```
public static CharIterator getCharIterator(char[][] data)
```

引数に指定した `char` を要素型とする配列の配列 (2次元の文字配列) から文字を順番に取り出す `CharIterator` を返す。引数に `null` が指定されたときは、`NullPointerException` を投げる。文字を取り出す順序は、文字配列とその配列のそれぞれのインデックス値の小さい順とする。すなわち、`char[][]` 型の  $m$  に対し、その要素 `m[i][j]` を  $i$  の小さい順、同じ  $i$  に対

しては j の小さい順に取り出す。

クラス CharIteratorTest は、CharIteratorFactory で定義されたメソッドをテストするためのプログラムである。メソッド main を実行すると、図の実行結果が得られる。

'H' '1' '6'
'2' '0' '0' '4'

図 CharIteratorTest.main の実行結果

[プログラム 1]

```
public interface CharIterator {
    public boolean hasNext();
    public char next();
}
```

[プログラム 2]

```
import java.util.NoSuchElementException;

public class CharIteratorFactory {
    public static CharIterator getCharIterator(String data) {
        if (data == null)
            throw new NullPointerException();
        // String データから文字を順番に返す CharIterator のインスタンス
        // を生成して返す。
        return a;
    }

    public static CharIterator getCharIterator(char[][] data) {
        if (data == null)
            throw new NullPointerException();
        // 2次元の文字配列から文字を順番に返す CharIterator のインスタンス
        // を生成して返す。
        return b;
    }
}

class StringCharIterator implements CharIterator {
    private String data;
    private int index = 0;
```

```
StringCharIterator(String data) {
    this.data = data;
}
// dataに次の文字があるかどうかをチェックする。
public boolean hasNext() {
    return c;
}

public char next() {
    // 次の文字がないときは, NoSuchElementException を投げる。
    if (index >= data.length())
        throw new NoSuchElementException();
    // dataの次の文字を返し, インデックス値を更新する。
    return d;
}
}

class Char2DArrayCharIterator implements CharIterator {
    private char[][] data;
    private int index1 = 0, index2 = 0;

    Char2DArrayCharIterator(char[][] data) {
        this.data = data;
    }
    public boolean hasNext() {
        // data[index1][index2]の要素があれば true を返し, なければ次
        // に定義されている要素を探す。次の要素がなければ false を返す。
        for (; index1 < data.length; index1++) {
            if (data[index1] != null
                && index2 < data[index1].length) {
                return true;
            }
            e;
        }
        return false;
    }
    public char next() {
        // メソッド hasNext を呼び出して次の要素があるかどうかを調べ,
        // あればその要素を返し, インデックス値を更新する。なければ,
        // NoSuchElementException を投げる。
        if (hasNext()) {
            return f;
        }
        throw new NoSuchElementException();
    }
}
}
```

[プログラム 3]

```
public class CharIteratorTest {
    public static void main(String[] args) {
        CharIterator itr =
            CharIteratorFactory.getCharIterator("H16");
        printIterator(itr);

        itr = CharIteratorFactory.getCharIterator(
            new char[][] {{ '2' },
                          { '0' },
                          null,
                          { '0', '4' }});

        printIterator(itr);
    }
    private static void printIterator(CharIterator itr) {
        while (itr.hasNext()) {
            System.out.print("'" + itr.next() + "' ");
        }
        System.out.println();
    }
}
```

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 `getCharIterator((char[][] data)`
- イ `getCharIterator((String) data)`
- ウ `new Char2DArrayCharIterator()`
- エ `new Char2DArrayCharIterator(data)`
- オ `new StringCharIterator()`
- カ `new StringCharIterator(data)`

c に関する解答群

- |  |  |
|--|--|
| ア <code>index &lt; data.length()</code>    | イ <code>index &lt;= data.length()</code>   |
| ウ <code>index &gt;= data.length()</code>   | エ <code>index++ &lt; data.length()</code>  |
| オ <code>index++ &lt;= data.length()</code> | カ <code>index++ &gt;= data.length()</code> |

dに関する解答群

- |   |                                     |   |                                   |
|---|-------------------------------------|---|-----------------------------------|
| ア | <code>data.charAt(++index)</code>   | イ | <code>data.charAt(--index)</code> |
| ウ | <code>data.charAt(index + 1)</code> | エ | <code>data.charAt(index++)</code> |
| オ | <code>data.charAt(index--)</code>   | カ | <code>data.charAt(index)</code>   |

eに関する解答群

- |   |                                   |   |                                   |
|---|-----------------------------------|---|-----------------------------------|
| ア | <code>index1 = 0</code>           | イ | <code>index1 = data.length</code> |
| ウ | <code>index1 = index2</code>      | エ | <code>index2 = 0</code>           |
| オ | <code>index2 = data.length</code> | カ | <code>index2 = index1</code>      |

fに関する解答群

- |   |                                       |   |                                     |
|---|---------------------------------------|---|-------------------------------------|
| ア | <code>data[index1++][index2++]</code> | イ | <code>data[index1++][index2]</code> |
| ウ | <code>data[index1][++index2]</code>   | エ | <code>data[index1][--index2]</code> |
| オ | <code>data[index1][index2 + 1]</code> | カ | <code>data[index1][index2++]</code> |



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〕

10進数の文字列を、編集して出力する副プログラム EDOUT である。

- (1) 主プログラムは、GR1 にパラメタの先頭番地を格納して EDOUT を呼び出す。パラメタの内容は、次のとおりである。

アドレス	
(GR1) + 0	文字列の先頭アドレス
+ 1	文字列の長さ

- (2) 文字列は、“0”～“9”の数字だけからなる。
- (3) 文字列中の各文字は、連続する語の下位8ビットに格納され、上位8ビットにはすべて0が格納されている。
- (4) EDOUT は文字列に対して次の編集を行い、18語の連続する領域 OUTB に右詰め  
で格納する。
- ① 文字列の右端から3文字ごとに、“,”を挿入する。
  - ② ①の結果を、1文字ずつ OUTB の各語の下位8ビットに格納する（上位8ビットには、すべて0を格納する）。
  - ③ ①の結果が18文字未満の場合、OUTB の残りの語には、間隔文字を格納する。

編集前の文字列

1 2 3 4
---------

出力する文字列

△△△△△△△△△△△△△△△△ 1 , 2 3 4
----------------------------

△：間隔文字を表す。

- ④ ①の結果が19文字以上の場合は、右から18文字分だけを格納する。

編集前の文字列

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
---------------------------------

出力する文字列

3 4 , 5 6 7 , 8 9 0 , 1 2 3 , 4 5 6
-------------------------------------

(5) 副プログラムから戻るとき、GR1 ~ GR7 の内容は元に戻す。

[プログラム]

(行番号)

```

1 EDOUT   START
2         RPUSH
3         LD    GR0,0,GR1           ; 先頭の数字のアドレスを GR0 へ
4         ST    GR0,ADDRS
5         LD    GR1,1,GR1           ; 文字列の長さを GR1 へ
6         ADDL  GR1,ADDRS
7         LAD   GR1,-1,GR1          ; 最後の数字のアドレス
8         LD    GR2,OUTLNG          ; 出力領域の添字
9         LAD   GR3,3                ; 位取りカウンタ初期化
10 LOOP   SUBA  GR2,=1              ; 出力領域の最後か?
11         a
12         CPL  GR1,ADDRS           ; 先頭の数字か?
13         JMI  SPACIN
14         SUBA  GR3,=1              ; 位取りの判断
15         b
16         LD    GR0,0,GR1
17         LAD   GR1,-1,GR1
18         JUMP  MOVCHR
19 RANK    c
20         LAD   GR3,3                ; 位取りカウンタ再初期化
21         JUMP  MOVCHR
22 SPACIN  LD    GR0,SPACE
23 MOVCHR  d
24         JUMP  LOOP
25 OUTPUT  OUT   OUTB,OUTLNG
26         RPOP
27         RET
28 OUTB    DS    18
29 ADDR    DS    1
30 COMMA   DC    ','
31 SPACE   DC    ' '
32 OUTLNG  DC    18
33         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	JMI	MOVCHR	イ	JMI	OUTPUT	ウ	JMI	RANK
エ	JNZ	MOVCHR	オ	JNZ	OUTPUT	カ	JNZ	RANK
キ	JZE	MOVCHR	ク	JZE	OUTPUT	ケ	JZE	RANK

c, d に関する解答群

ア	LAD	GR0,0,GR1	イ	LD	GR0,COMMA
ウ	LD	GR0,SPACE	エ	ST	GR0,OUTB,GR1
オ	ST	GR0,OUTB,GR2	カ	ST	GR1,OUTLNG
キ	ST	GR2,OUTLNG			

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。

入力文字列を“1 2 3 4 5 6 7 8 9 0”とし、プログラムの行番号 16 の命令を実行する直前の GR1 が指すメモリの内容が“6”のとき、GR0 の値は  e  , GR2 の値は  f  , GR3 の値は  g  である。

e に関する解答群

ア	#0030	イ	#0031	ウ	#0032	エ	#0033	オ	#0034
カ	#0035	キ	#0036	ク	#0037	ケ	#0038	コ	#0039

f に関する解答群

ア	#0009	イ	#000A	ウ	#000B	エ	#000C	オ	#000D
カ	#000E	キ	#000F	ク	#0010	ケ	#0011	コ	#0012

g に関する解答群

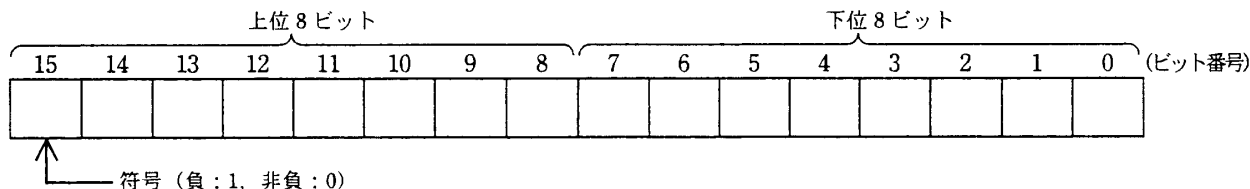
ア	#0000	イ	#0001	ウ	#0002	エ	#0003	オ	#FFFF
---	-------	---	-------	---	-------	---	-------	---	-------

## ■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	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r2)$ $r \leftarrow (\text{実効アドレス})$	○*1
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	$r1, r2$ $r, \text{adr} [, x]$	$r1 \leftarrow (r1) \text{ XOR } (r2)$ $r \leftarrow (r) \text{ XOR } (\text{実効アドレス})$	○*1

(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)			
			(r1) = (r2)		0	1
			(r) = (実効アドレス)			
(r1) < (r2)	1	0				
(r) < (実効アドレス)						

(4) シフト演算命令

算術左シフト 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。シフトの結果, 空いたビット位置には, 左シフトのときは 0, 右シフトのときは符号と同じものが入る。	○*2
算術右シフト Shift Right Arithmetic	SRA	$r, \text{adr} [, x]$		
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	$\text{adr} [, x]$	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	$\text{adr} [, x]$			
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。
					分岐するときの FR の値
					命令
			OF	SF	ZF
			JPL	0	0
			JMI	1	
			JNZ		0
			JZE		1
			JOV	1	

(6) スタック操作命令

プッシュ PUSH	PUSH    adr [,x]	SP ← (SP) - <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 ]

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、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。

〔例題〕 次の  に入れる正しい答えを、解答群の中から選べ。

春の情報処理技術者試験は、 a  月に実施される。

解答群

ア 2            イ 3            ウ 4            エ 5

正しい答えは“ウ 4”ですから、次のようにマークしてください。

例題	a	<input checked="" type="radio"/> ア	<input type="radio"/> イ	<input type="radio"/> ウ	<input 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。