平成17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

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。

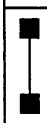
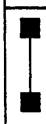
注意事項は問題冊子の裏表紙に続きます。
こちら側から裏返して，必ず読んでください。

C
COBOL
Java
アセンブラ

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。
/* 文 */		文に注釈を記述する。
処理	・変数 ← 式	変数に式の値を代入する。
	・手続(引数, …)	手続を呼び出し，引数を受け渡す。
		単岐選択処理を示す。 条件式が真のときは処理を実行する。
		双岐選択処理を示す。 条件式が真のときは処理 1 を実行し，偽のときは処理 2 を実行する。
		前判定繰返し処理を示す。 条件式が真の間，処理を繰り返し実行する。
		後判定繰返し処理を示す。 処理を実行し，条件式が真の間，処理を繰り返し実行する。
		繰返し処理を示す。 開始時点で変数に初期値（定数又は式で与えられる）が格納され，条件式が真の間，処理を繰り返す。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 ↑
乗除演算	×，÷，%	
加減演算	+，−	
関係演算	＞，＜，≧，≦，＝，≠	
論理積	and	
論理和	or	↓ 低

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　リストに関する次の記述を読んで，設問1～3に答えよ。

リストの構造は図1のとおりとする。

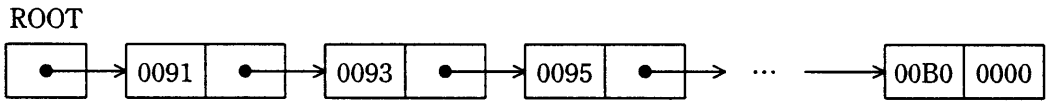


図1　リストの構造

(1) ROOTは，リストの先頭を指す。

(2) リストの要素は連続する2語からなる。第1語には値が，第2語には次の要素へのポインタが格納されている。

(3) リストの各要素は値の昇順に連結されていて，値はすべて異なる。最後の要素の第2語にはポインタとして0000が格納されている。

(4) 図1の構造をもつリストが，図2のとおりに主記憶の00FF番地から0117番地までに格納されている。00FF番地はROOTである。

(5) 1語は16ビットからなり，語単位で番地が付いている。

番地	内容
⋮	⋮
00FF	0100
0100	0091
0101	010A
0102	009F
0103	0106
0104	0095
0105	0116

番地	内容
0106	00A0
0107	010C
0108	00A9
0109	0112
010A	0093
010B	0104
010C	
010D	0114

番地	内容
010E	00B0
010F	0000
0110	009B
0111	0102
0112	00AB
0113	010E
0114	00A2
0115	0108

番地	内容
0116	0099
0117	0110
0118	00A7
0119	0000
011A	009C
011B	011C
011C	00A5
011D	0118

図2　主記憶の状態

設問1　010C 番地の内容として正しい答えを，解答群の中から選べ。

解答群

ア　0099　　イ　00A1　　ウ　00A3　　エ　00A4　　オ　00A5

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。

0110 番地及び 0111 番地からなる要素をリストから削除するには，［ a ］番地の内容を［ b ］に変えればよい。

解答群

ア　0101　　イ　0102　　ウ　0103　　エ　0104　　オ　0105

カ　0113　　キ　0114　　ク　0115　　ケ　0116　　コ　0117

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。

0118 番地から 011D 番地に格納されている 3 要素からなるサブリストと，設問 2 で要素を削除する前のリストを併合するには，［ c ］番地の内容を 011A に，［ d ］番地の内容を 0102 に，［ e ］番地の内容を 011C に，［ f ］番地の内容を 0108 にそれぞれ変えればよい。

解答群

ア　0109　　イ　010B　　ウ　010D　　エ　010F　　オ　0111

カ　0113　　キ　0115　　ク　0117　　ケ　0119　　コ　011B

問2　次のプログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

副プログラム Replace は，文字列を置換するプログラムである。

(1) Replace は，文字列 A を先頭からコピーして文字列 B を作る。このとき，文字列 A 中に文字列 S と一致する文字列を見つけるたびに，文字列 D の内容に置き換える。

(2) このプログラムで扱う文字列は，1 文字ずつ文字型配列の要素に順に格納されていて，最後の文字の次に定数 EOS が格納されている。配列の添字は 0 から始まる。

(3) 文字列 A，S，D は，それぞれ文字型配列 A[ ]，S[ ]，D[ ] に格納されていて，文字列 B は文字型配列 B[ ] に格納する。文字型配列 B[ ] は十分に大きいものとする。

(4) 副プログラムの引数の仕様を表に，実行例を図に示す。

表　Replace の引数の仕様

引数名	データ型	入力／出力	意味
A[ ]	文字型	入力	対象文字列
S[ ]	文字型	入力	照合文字列
D[ ]	文字型	入力	置換文字列
B[ ]	文字型	出力	結果文字列

添字	0	1	2	3	4	5	6	7	8	9
A[ ]	“a”	“a”	“b”	“c”	“a”	“b”	“b”	EOS		
S[ ]	“a”	“b”	EOS							
D[ ]	“A”	“B”	“C”	EOS						
B[ ]	“a”	“A”	“B”	“C”	“c”	“A”	“B”	“C”	“b”	EOS

図　Replace の実行例

〔プログラム〕

```
○Replace(文字型: A[], 文字型: S[], 文字型: D[], 文字型: B[])
○整数型: Aidx, Sidx, Didx, Bidx, Idx
・Aidx ← 0
・Bidx ← 0
■ A[Aidx] ≠ EOS
    ▲ A[Aidx] = S[0]
        ・Idx ← Aidx
        ・ a
        ■                                   /* 後判定繰返し処理を示す */
            ・Sidx ← Sidx + 1               /* 処理を実行する */
            ・Aidx ← Aidx + 1               /* 条件式が真の間，処理を繰り返す */
        ■ b     and  A[Aidx] ≠ EOS
        ▲ c
            ・Didx ← 0
            ■ D[Didx] ≠ EOS
                ・B[Bidx] ← D[Didx]
                ・Didx ← Didx + 1
                ・Bidx ← Bidx + 1
            ■
        ―――――――――――――――
            ・ d
            ・Aidx ← Idx + 1
            ・Bidx ← Bidx + 1
        ▼
    ―――――――――――――――
        ・B[Bidx] ← A[Aidx]
        ・Aidx ← Aidx + 1
        ・Bidx ← Bidx + 1
    ▼
■
・B[Bidx] ← EOS
```

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，d に関する解答群

ア　B[Bidx] ← A[Aidx]　　イ　B[Bidx] ← S[0]

ウ　Idx ← 0　　エ　Idx ← 1

オ　Sidx ← 0　　カ　Sidx ← 1

b, c に関する解答群

ア A[Aidx] ≠ EOS

イ A[Aidx] = S[Sidx]

ウ A[Aidx] ≠ S[Sidx]

エ A[Aidx] ≠ S[Sidx] and S[Sidx] ≠ EOS

オ S[Sidx] = EOS

カ S[Sidx] ≠ EOS

問3　オンラインシステムに関する次の記述を読んで，設問1，2に答えよ。

このシステムは，Webサブシステム，アプリケーション（AP）サブシステム及びデータベース（DB）サブシステムの3階層の構成をとる。これらのサブシステムの現在のサーバ構成は，次のとおりである。

(1) Webサブシステムは，2台の同一機種のWebサーバで構成されている。

(2) APサブシステムは，2台の同一機種のアプリケーション（AP）サーバで構成されている。

(3) DBサブシステムは，1台のデータベース（DB）サーバで構成されている。

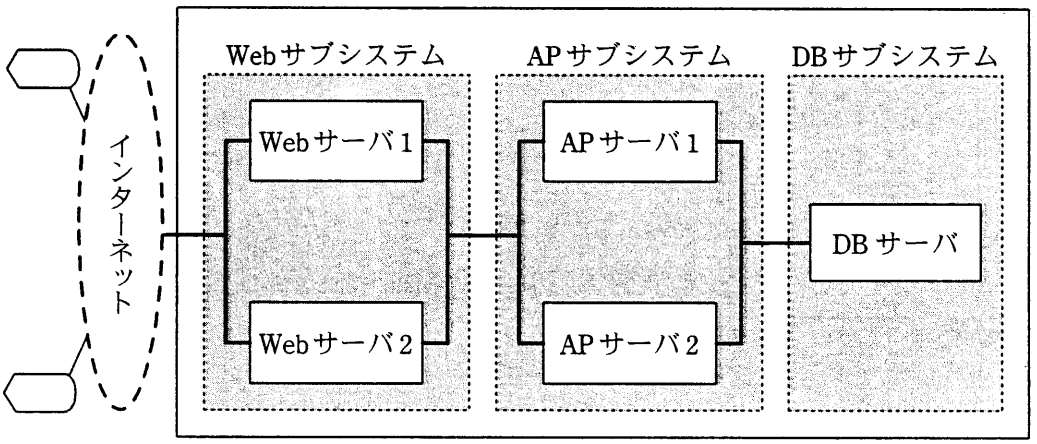


図　オンラインシステムの現在のシステム構成

このシステムは，アクセス数の増加に対処するために，各サブシステムにサーバを追加できるように設計されている。システムの処理の概要及び各サブシステムが稼働するための必要条件は，次のとおりである。

(1) 利用者からの正しい要求はすべて，Webサブシステム → APサブシステム → DBサブシステムの順に処理し，DBサブシステム → APサブシステム → Webサブシステムの順に応答を生成して利用者に返す。

(2) 利用者からの要求は，Webサブシステム中のいずれか1台のサーバによって処理される。すなわち，Webサブシステム中のサーバが少なくとも1台稼働していれば，Webサブシステムは稼働しているとみなされる。

(3) 利用者からの要求1件に対して，Webサブシステムは，APサブシステム中の2台のサーバを同時に使用する。すなわち，APサブシステム中のサーバが少なくとも2台稼働していれば，APサブシステムは稼働しているとみなされる。

(4) すべての処理要求は，DBサブシステム中の1台のサーバによって処理される。すなわち，DBサブシステム中のサーバが少なくとも1台稼働していれば，DBサブシステムは稼働しているとみなされる。

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。

(1) Webサーバ単体の平均故障間隔（MTBF）をF_W，平均修理時間（MTTR）をR_Wとする。

(2) APサーバ単体の平均故障間隔をF_A，平均修理時間をR_Aとする。

(3) DBサーバ単体の平均故障間隔をF_D，平均修理時間をR_Dとする。

Webサブシステムの稼働率は［ a ］であり，APサブシステムの稼働率は［ b ］である。また，Webサブシステム，APサブシステム及びDBサブシステムの稼働率をそれぞれ，U_W，U_A及びU_Dとすると，このシステム全体としての稼働率は，［ c ］である。

aに関する解答群

ア $\left(1-\dfrac{F_W}{F_W+R_W}\right)^2$　　イ $\left(\dfrac{F_W}{F_W+R_W}\right)^2$

ウ $1-\left(1-\dfrac{F_W}{F_W+R_W}\right)^2$　　エ $\dfrac{F_W}{F_W+R_W}$

bに関する解答群

ア $\left(1-\dfrac{F_A}{F_A+R_A}\right)^2$　　イ $\left(\dfrac{F_A}{F_A+R_A}\right)^2$

ウ $1-\left(1-\dfrac{F_A}{F_A+R_A}\right)^2$　　エ $\dfrac{F_A}{F_A+R_A}$

c に関する解答群

ア $\frac{U_W+U_A+U_D}{3}$　　イ $1-(1-U_W)\times(1-U_A)\times(1-U_D)$

ウ $U_W\times U_A\times U_D$　　エ U_W, U_A, U_D の最小値

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。

このシステムにおいて，すべてのサーバの稼働率は等しく 0.9 であるとする。ここで，稼働率 0.9 のサーバを 1 台，いずれかのサブシステムに並列構成で追加する。Web サブシステムに追加したときのシステム全体の稼働率を W，AP サブシステムに追加したときのシステム全体の稼働率を A，DB サブシステムに追加したときのシステム全体の稼働率を D とすると，W，A，D の大小関係は［　　］になる。

解答群

ア $A>D>W$　　イ $A>W>D$　　ウ $D>A>W$

エ $D>W>A$　　オ $W>A>D$　　カ $W>D>A$

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〕

副プログラム HeapSort は，配列に格納されている整数値をヒープソートで昇順に整列するプログラムである。

(1) 整列する Num 個（Num ≧ 2）の整数値は，大域変数の配列 A[1]，A[2]，…，A[Num] に格納されている。

(2) ヒープソートは，2分木を用いてデータを整列する。2分木を配列で表現するには，ある節が A[i] に対応するとき，その左側の子の節を A[2×i] に，右側の子の節を A[2×i+1] に対応させる。図では，丸が節を，丸中の数字が節の値を示し，実際に値が格納されている配列の要素を丸の脇(わき)に示している。

(3) ヒープとは，図のように，各節の値が自分の子の節の値以上になっている2分木である。

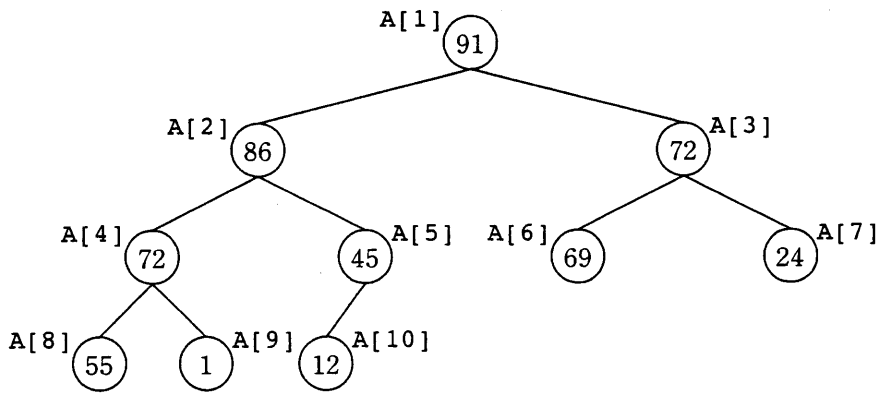


図　ヒープの例

(4) 整列の手順は，次のとおりである。

① 配列 A の中の，A[1]，A[2]，…，A[Num] を整列対象とする。

② 整列対象の各要素が，(2) で述べたような2分木を表現しているものとして，要素の値を入れ換えてヒープを作る。その結果，木の根（A[1]）には整列対象の要素の最大値が入る。

③ 木の根（A[1]）の値と，木の最後の節（整列対象の最後の要素に対応する節）の値とを交換する。

④　木の最後の節を2分木から取り除く（整列対象の要素を一つ減らす）。

⑤　④の結果が，木の根だけになるまで，②～④の処理を繰り返す。

(5) ヒープを作り直す手順は，次のとおりである。

①　木の根を親の節とする。

②　子の節がなければ終了する。

③　二つの子の節のうち値が大きい方と親の節の値を比較し，親の節の方が小さいときは，値を交換する。親の節の値が子の節の値以上のときは，何もしないで終了する。

④　値を交換した子の節を根とする部分木に対して，①～③の処理を繰り返す。

(6) 副プログラムの引数の仕様を表1～4に示す。

表1　HeapSort の引数の仕様

引数名	データ型	入力／出力	意味
Num	整数型	入力	木の最後の節に対応する配列要素の添字

表2　InitHeap の引数の仕様

引数名	データ型	入力／出力	意味
Last	整数型	入力	最初にヒープを作る木の，最後の節に対応する配列要素の添字

表3　MakeHeap の引数の仕様

引数名	データ型	入力／出力	意味
Top	整数型	入力	ヒープを作り直す部分木の，木の根に対応する配列要素の添字
Last	整数型	入力	ヒープを作り直す部分木の，最後の節に対応する配列要素の添字

表4　Swap の引数の仕様

引数名	データ型	入力／出力	意味
X	整数型	入力	A[Y] と交換する配列要素の添字
Y	整数型	入力	A[X] と交換する配列要素の添字

〔プログラム〕

```
○整数型: A[1000000]                          /* 大域変数として用いる */

○HeapSort(整数型: Num)
○整数型: Idx
/* 最初にヒープを作成 */
・InitHeap(Num)
/* 並べ替え */
■ Idx: Num, Idx ＞ 1, −1
    ・Swap(1, Idx)
    ・MakeHeap(1, Idx−1)
■

○MakeHeap(整数型: Top, 整数型: Last)
○整数型: L, R
・ a
・R ← L + 1
▲ R ≦ Last                                   /* 3個比較 */
    ▲ A[L] ＜ A[R]                           /* 右が大きい */
        ▲ A[Top] ＜ A[R]
            ・Swap(Top, R)
            ・MakeHeap(R, Last)
    ――――                                   /* 左が大きい */
        ▲ A[Top] ＜ A[L]
            ・Swap(Top, L)
            ・MakeHeap(L, Last)
――――――
    ▲  b                                     /* 2個比較 */
        ▲ A[Top] ＜ A[L]
            ・Swap(Top, L)
            ・MakeHeap(L, Last)
▼

○Swap(整数型: X, 整数型: Y)
○整数型: Tmp
・Tmp ← A[X]
・A[X] ← A[Y]
・A[Y] ← Tmp
```

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　L ← Top　　イ　L ← Top ＋ 1

ウ　L ← Top × 2　　エ　L ← Top × 2 ＋ 1

b に関する解答群

ア　L ≦ Last　　イ　L ＜ Last

ウ　R ≦ Last － 1　　エ　R ≦ Last － 2

設問2　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

図のヒープを用いて，〔プログラムの説明〕の(4)の③，④を1回だけ実行して，②が終了したとき，最初 A[10] に入っていた値 12 が格納されている配列要素の添字は［　c　］である。また，それまでに節の値を交換した回数は［　d　］である。

解答群

ア　2　　イ　3　　ウ　4　　エ　5　　オ　6

カ　7　　キ　8　　ク　9

設問3　最初にヒープを作成する副プログラム InitHeap は MakeHeap を使って作ることができる。次のプログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

```
○InitHeap(整数型: Last)
○整数型: Idx
■［　　］
  ・MakeHeap(Idx, Last)
■
```

解答群

ア　Idx: 1, Idx ≦ Last, 2

イ　Idx: 1, Idx ≦ Last ÷ 2, 1

ウ　Idx: Last, Idx ≧ 1, −2

エ　Idx: Last ÷ 2, Idx ≧ 1, −1

問5　プログラム設計に関する次の記述を読んで，設問1，2に答えよ。

ある大学では，既存システムから出力される授業の時間割ファイルを，新システムの入力とするために，ファイル編集プログラムを作成することになった。

この大学の時間割では，同じ年次，曜日，時限に複数の科目が存在する場合がある。

既存システムで出力ファイルを作成した後に，時間割の変更が生じた場合，担当者が出力ファイルを修正してから，編集プログラムに入力することも想定されている。さらに，新システムでは，入力レコードの各項目の値の範囲及び順序の誤りを検査する機能をもっていないので，これらの検査も編集プログラムの機能に含めることにした。

この大学の時間割の例，既存システムの出力ファイル，新システムの入力ファイル及び編集プログラムの流れを次に示す。

〔時間割の例〕

曜日	時限	1年次		2年次		3年次		4年次	
月	1	哲学	井上紘司	社会学	瀬川譲治	国際政治	鈴木響子		
		政治学	小林慎一						
	2	英語Ⅰ	間宮剛士	現代経済	竹上松次				
				情報倫理	城乃守衛				
	3	歴史学	新田敦	憲法	桑原堅次				
	4			行動科学	水田隆雄				
	5	環境論	田中武彦						
火	1	体育	岡忠	会計Ⅰ	島田幸作	会計Ⅱ	藤原俊夫		
		体育	下山恵	独語Ⅱ	森田五郎	仏語Ⅲ	山田陽一		
	2					認知科学	星野玉子	経営学	中沢晃子
⋮	⋮	⋮		⋮		⋮		⋮	

図1　時間割の例

〔既存システムの出力ファイル〕

既存システムは，次の様式で，年次，曜日，時限，表示順をキーとして，昇順に時間割ファイルのレコードを出力する。

年次：1～4年次を，数字1～4で表す。

曜日：日～土の曜日を，数字1～7で表す。

時限：1～9時限（最大9時限まで）を，数字1～9で表す。

表示順：同一年次の同一曜日，同一時限の科目の表示の順番を，1から始まる連続した数字で表す。

科目名：“哲学”，“政治学”などを，文字列で記述する。

教員名：“井上紘司”，“小林慎一”などを，文字列で記述する。

レコード様式は，次のとおりである。

年次	曜日	時限	表示順	科目名	教員名

〔新システムの入力ファイル〕

年次，曜日，時限，科目名，教員名は，それぞれ1レコードとして入力する。ただし，既存システムの出力と同じ順番を維持する。

複数の科目を，同一年次の同一曜日，同一時限に指定する場合は，登録する順に，科目名と教員名の組を連続して入力する。

年次，曜日，時限のレコードは，それぞれが変わるときにだけ入力する。

レコードの内容は，次に示す年次，曜日，時限，科目名，教員名のいずれかである。年次，曜日，時限は，先頭の半角英字1文字と，それに続く半角数字1文字で指定する。科目名と教員名は，全角の文字列で指定する。

年次：1～4年次を，記号G1～G4で表す。

曜日：日～土の曜日を，記号D1～D7で表す。

時限：1～9時限（最大9時限まで）を，記号T1～T9で表す。

科目名：“哲学”，“政治学”などを，文字列で記述する。

教員名：“井上紘司”，“小林慎一”などを，文字列で記述する。

既存システムの出力例の一部と対応する新システムへの入力を図2に示す。

既存システムの出力

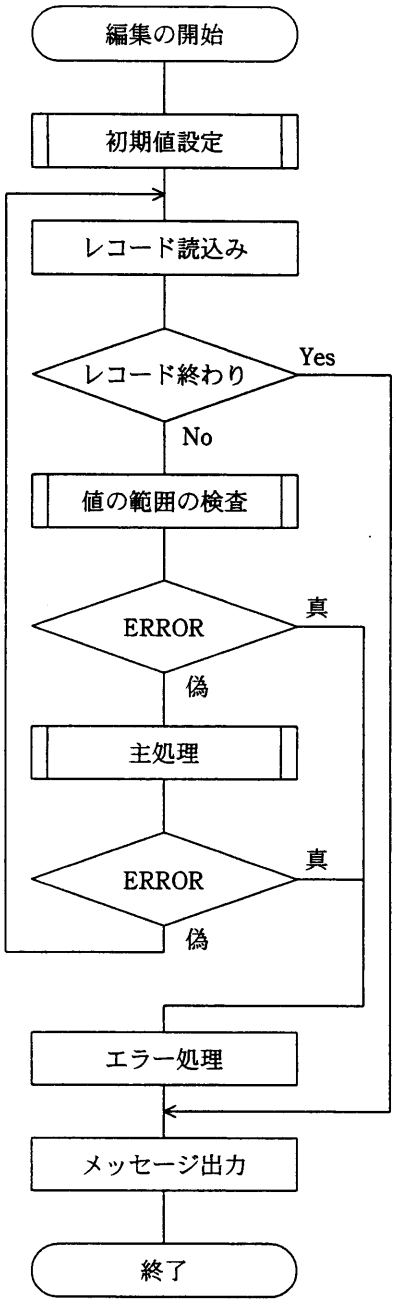
	年次	曜日	時限	表示順	科目名	教員名
1	1	2	1	1	哲学	井上紘司
2	1	2	1	2	政治学	小林慎一
3	1	2	2	1	英語Ⅰ	間宮剛士
4	1	2	3	1	歴史学	新田敦
5	1	2	5	1	環境論	田中武彦
6	1	3	1	1	体育	岡忠
7	1	3	1	2	体育	下山恵

新システムへの入力

	レコードの内容
1	G1
2	D2
3	T1
4	哲学
5	井上紘司
6	政治学
7	小林慎一
8	T2
9	英語Ⅰ
10	間宮剛士
11	T3
12	歴史学
13	新田敦
14	T5
15	環境論
16	田中武彦
17	D3
18	T1
19	体育
20	岡忠
21	体育
22	下山恵

図2　各ファイルの内容の例

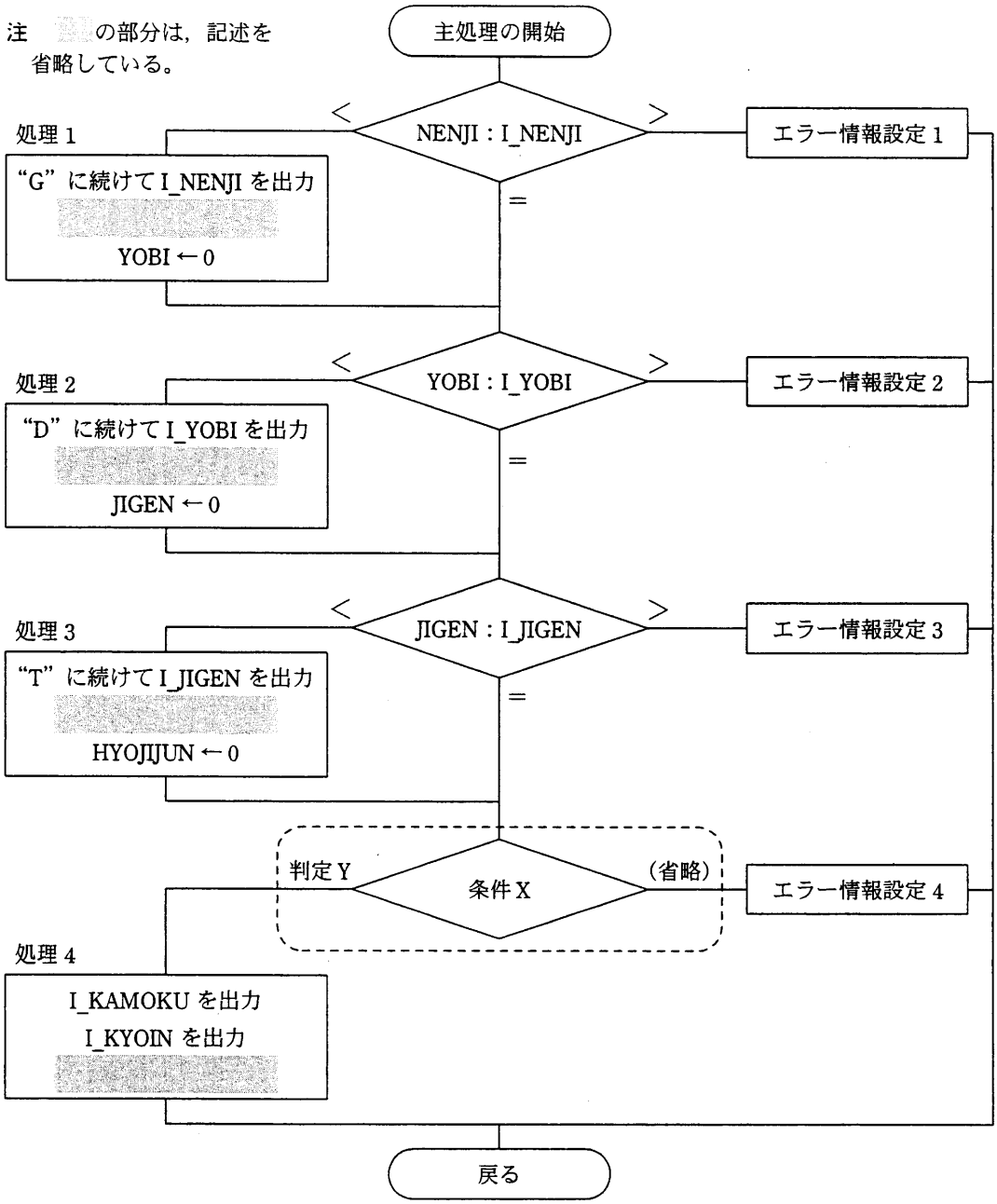
〔プログラムの流れ〕



変数の説明

ERROR： 直前の処理（値の範囲の検査，主処理）で誤りを検出したときに値が真に設定される変数

図３　編集プログラムの流れ



変数の説明

I_NENJI：	入力レコードの年次の値をもつ変数	NENJI：	年次を格納する変数
I_YOBI：	入力レコードの曜日の値をもつ変数	YOBI：	曜日を格納する変数
I_JIGEN：	入力レコードの時限の値をもつ変数	JIGEN：	時限を格納する変数
I_HYOJIJUN：	入力レコードの表示順の値をもつ変数	HYOJIJUN：	表示順を格納する変数
I_KAMOKU：	入力レコードの科目名の値をもつ変数		
I_KYOIN：	入力レコードの教員名の値をもつ変数		

図 4　主処理の流れ

設問1　編集プログラムの振る舞いに関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

既存システムの出力ファイルと，新システムへの入力ファイルの特徴から，編集プログラムが出力するレコード件数は，正しい入力レコード1件に対し，最小で［ a ］，最大で［ b ］となる。

図1の時間割を既存システムが出力し，それをそのまま編集プログラムに入力した場合，3年次の火曜日1時限の会計Ⅱの入力レコードに対して，主処理中の処理1～4のうち，実行されるのは［ c ］及び処理4である。

a，bに関する解答群

ア　1　　イ　2　　ウ　3　　エ　4　　オ　5

カ　6

cに関する解答群

ア　処理1　　イ　処理1，処理2

ウ　処理1，処理2，処理3　　エ　処理1，処理3

オ　処理2　　カ　処理2，処理3

キ　処理3

設問2　図3及び図4中の各処理の内容に関する次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

条件判定に用いる変数（NENJI，YOBI，JIGEN，HYOJIJUN）の新しい値の設定又はこれらの変数の必要最小限の初期化は，主処理中の処理1～4の中で行っている。

このうち処理3の［　　］は，［ d ］である。また，編集プログラムの最初に行う，初期値設定での必要十分な変数の初期化は，［ e ］である。

主処理中の［　　］の部分の条件Xは，［ f ］であり，判定Yは，［ g ］である。

d に関する解答群

ア　JIGEN ← 0

イ　JIGEN ← I_JIGEN

ウ　NENJI ← 0
　　YOBI ← 0
　　JIGEN ← 0

エ　YOBI ← 0
　　JIGEN ← 0

e に関する解答群

ア　ERROR ← 偽

イ　ERROR ← 偽
　　NENJI ← 0

ウ　ERROR ← 偽
　　NENJI ← 0
　　YOBI ← 0

エ　ERROR ← 偽
　　NENJI ← 0
　　YOBI ← 0
　　JIGEN ← 0

オ　ERROR ← 偽
　　NENJI ← 0
　　YOBI ← 0
　　JIGEN ← 0
　　HYOJIJUN ← 0

f に関する解答群

ア　HYOJIJUN－1 : I_HYOJIJUN

イ　HYOJIJUN－1 : I_HYOJIJUN＋1

ウ　HYOJIJUN＋1 : I_HYOJIJUN

エ　HYOJIJUN＋1 : I_HYOJIJUN－1

オ　HYOJIJUN : I_HYOJIJUN

g に関する解答群

ア　＝　　イ　＞　　ウ　＜

エ　≠　　オ　≦　　カ　≧

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 print_string は，欧文ピッチ処理（文字固有の字幅に従って字送りする）を行って印刷するとき，単語の途中で改行されないように英文を出力するプログラムである。

(1) 関数 print_string の引数は，次のとおりである。

line_w	1 行の行幅（ポイント数）
str_list	出力する英文を構成する単語の配列 （最後の要素には，NULL が格納されている）
char_list	出力する単語を構成する文字とその文字幅（ポイント数）のリスト（構造体 CHARPROF の配列）
space_w	空白文字の文字幅（ポイント数）

(2) 文字幅は，文字ごとに次に示す構造体 CHARPROF で定義される。

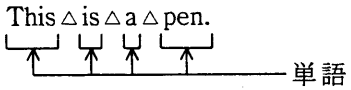
```
typedef struct { char char_p; /* 文字 */
                 int  char_w; /* 文字幅（ポイント数） */
               } CHARPROF;
```

(3) 単語幅は，単語を構成する各文字の文字幅の和である。単語幅を求めるために，関数 word_width を用いる。

(4) 単語を出力しようとしたときに，1 行の行幅を超える場合は，その単語が次の行の先頭になるように出力する。ただし，どの単語幅も行幅を超えることはない。

(5) 単語は，空白を含まない文字列である。単語と単語の間には，1 文字の空白文字を出力する。ただし，行の最後に出力する単語の後には，空白文字は出力しない。

（例）



注　△は空白を示す。

〔プログラム〕

```
#include <stdio.h>

typedef struct { char char_p; /* 文字 */
                 int  char_w; /* 文字幅（ポイント数） */
               } CHARPROF;

void print_string(int, char *[], CHARPROF *, int);
int word_width(char *, CHARPROF *);

void print_string(int line_w, char *str_list[],
                  CHARPROF *char_list, int space_w) {
    int cur_w = 0, str_w, idx;

    for (idx = 0; [ a ]; idx++) {
        str_w = word_width(str_list[idx], char_list);
        [ b ];
        if (cur_w == str_w)     /* 最初の単語？ */
            printf("%s", str_list[idx]);
        else {
            cur_w += space_w;
            if (cur_w <= line_w)
                printf(" %s", str_list[idx]);
            else {
                [ c ];
                printf("\n%s", str_list[idx]);
            }
        }
    }
    putchar('\n');
}

int word_width(char *str, CHARPROF *char_list) {
    int print_w = 0, idx;

    while (*str != '\0') {
        for (idx = 0; [ d ]; idx++);
        print_w += char_list[idx].char_w;
        str++;
    }
    return print_w;
}
```

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　`idx <= line_w`　　イ　`idx < line_w`

ウ　`idx + 1 < line_w`　　エ　`str_list[idx] != NULL`

オ　`str_list[idx] = NULL`　　カ　`str_list[idx] == NULL`

b, c に関する解答群

ア　`cur_w += space_w`　　イ　`cur_w += str_w`

ウ　`cur_w = 0`　　エ　`cur_w = space_w`

オ　`cur_w = space_w + str_w`　　カ　`cur_w = str_w`

d に関する解答群

ア　`*str != *char_list[idx].char_p`

イ　`*str != char_list[idx].char_p`

ウ　`*str == *char_list[idx].char_p`

エ　`*str == char_list[idx].char_p`

オ　`str != char_list[idx].char_p`

カ　`str == char_list[idx].char_p`

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〕

あるイベントの来場者アンケートを集計し，年代別職業別参加人数表を出力するプログラムである。

(1) アンケートファイルのレコード様式は，次のとおりである。

年齢 1けた	職業 1けた	興味 1けた	その他の答え 97けた

① 年齢は，年代別にコード化されており，次のいずれかが記録されている。

1：10～19歳，　2：20～29歳，　3：30～39歳，

4：その他（9歳以下又は40歳以上）

② 職業は，学生，自営業，会社員などに分類してコード化されており，次のいずれかが記録されている。

1：学生，　2：自営業，　3：会社員，　4：その他

③ 興味は，来場の主目的が，体験，セミナ，展示などのどれであるかをコード化したものであり，次のいずれかが記録されている。

1：体験，　2：セミナ，　3：展示，　4：その他

(2) 来場者アンケートを集計して，次の年代別職業別参加人数表を出力する。

行
1→
2→
3→
4→
5→
6→
7→
8→
9→
⋮

年代別職業別参加人数表

	学生	自営業	会社員	その他	合計
10代	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9
20代	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9
30代	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9
その他	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9	zzz,zz9

① 各参加人数は，来場者アンケートを集計した結果で，総参加人数は最大6けたである。

② 年代別，職業別に参加人数を印字し，さらに年代別の合計を印字する。

③ 参加人数及び合計人数以外の文字は，あらかじめ印字されている。

(3) アンケートファイルのレコードの各項目にエラーはない。

〔プログラム〕

```
DATA DIVISION.
FILE SECTION.
FD  ENQUETE-F.
01  ENQUETE-R.
    03  E-NENREI      PIC 9(1).
    03  E-SHOKUGYO    PIC 9(1).
    03  E-KYOMI       PIC 9(1).
    03                 PIC X(97).
FD  PRINT-F.
01  PRINT-R PIC X(100).
WORKING-STORAGE SECTION.
01  HYO.
    03  A              OCCURS 4 INDEXED BY X.
      05  B            OCCURS 4 INDEXED BY Y PIC 9(6).
01  END-SW             PIC X(3) VALUE SPACE.
01  KUHAKU             PIC X(100) VALUE SPACE.
01  MEISAI.
    03                 PIC X(10) VALUE SPACE.
    03                 OCCURS 5 INDEXED BY Z.
      05               PIC X(4) VALUE SPACE.
      05  NINZUU       PIC ZZZ,ZZ9.
01  GOKEI              PIC 9(6) VALUE ZERO.
PROCEDURE DIVISION.
HAJIME.
    OPEN INPUT ENQUETE-F OUTPUT PRINT-F.
    INITIALIZE HYO.
    PERFORM UNTIL END-SW = "END"
      READ ENQUETE-F
        AT END MOVE "END" TO END-SW
        NOT AT END
α ➡       COMPUTE B(E-NENREI E-SHOKUGYO)
                = B(E-NENREI E-SHOKUGYO) + 1
      END-READ
    END-PERFORM.
    WRITE PRINT-R FROM KUHAKU AFTER PAGE.
    WRITE PRINT-R FROM KUHAKU AFTER [ a ].
    PERFORM VARYING X FROM 1 BY 1 UNTIL X > 4
      PERFORM VARYING Y FROM 1 BY 1 UNTIL Y > 4
        [ b ]
        MOVE B(X Y) TO NINZUU(Z)
        [ c ]
      END-PERFORM
```

```
        MOVE GOKEI TO NINZUU(5)
        WRITE PRINT-R FROM MEISAI AFTER 2
        MOVE ZERO TO GOKEI
    END-PERFORM.
    CLOSE ENQUETE-F PRINT-F.
    STOP RUN.
```

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　1　　イ　2　　ウ　3　　エ　4

b，c に関する解答群

ア　COMPUTE GOKEI = GOKEI + B(X Y)

イ　COMPUTE GOKEI = NINZUU(Z) + B(X Y)

ウ　COMPUTE NINZUU(Z) = GOKEI + B(X Y)

エ　MOVE B(X Y) TO GOKEI

オ　MOVE X TO Z

カ　MOVE Y TO Z

キ　MOVE Z TO Y

ク　SET X TO Z

ケ　SET Y TO Z

コ　SET Z TO Y

設問2　出力用紙に次の用紙を使い，職業別目的別参加人数表を出力するとき，プログラム中の**α**をどの文と置換すればよいか，正しい答えを，解答群の中から選べ。

職業別目的別参加人数表

	体験	セミナ	展示	その他	合計
学生					
自営業					
会社員					
その他					

解答群

ア　COMPUTE B(E-KYOMI E-NENREI) = B(E-KYOMI E-NENREI) + 1

イ　COMPUTE B(E-KYOMI E-SHOKUGYO) = B(E-KYOMI E-SHOKUGYO) + 1

ウ　COMPUTE B(E-NENREI E-KYOMI) = B(E-NENREI E-KYOMI) + 1

エ　COMPUTE B(E-NENREI E-SHOKUGYO) = B(E-NENREI E-SHOKUGYO) + 1

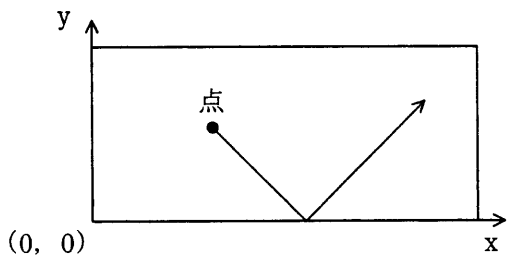
オ　COMPUTE B(E-SHOKUGYO E-KYOMI) = B(E-SHOKUGYO E-KYOMI) + 1

カ　COMPUTE B(E-SHOKUGYO E-NENREI) = B(E-SHOKUGYO E-NENREI) + 1

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1，2に答えよ。

〔プログラムの説明〕

次の図のような長方形領域内を移動する点を描画するプログラムである。



点はクラス Point で表され，点の位置を表す座標（x，y）と速さ（speed）を保持する。速さは，正の値で単位時間にx軸方向及びy軸方向に移動する距離を表す。

図の長方形領域はクラス Space で与えられ，次のクラスメソッドを呼び出すことができる。

(1) `public static int getMaxX()`

長方形領域のx座標の最大値（正の値）を返す。

(2) `public static int getMaxY()`

長方形領域のy座標の最大値（正の値）を返す。

(3) `public static void draw(Point)`

引数で指定された点をその座標位置に描画する。

(4) `public static void erase(Point)`

引数で指定された点を消去する。

抽象クラス Motion は，点が移動する様子を示すために，コンストラクタに Point で与えられた点を初期値とし，点の描画，移動及び消去をこの順番で繰り返す。点の移動後の座標位置は抽象メソッド update で与えられる。

Motion のサブクラス SimpleMotion は，メソッド update と main を実装する。メソッド update は，長方形内では，一定の速さで直線運動し，長方形の境界で反射するような点の動きを表す。メソッド main はプログラムをテストする。

なお，メソッド main で生成される Point の座標の初期値は長方形領域内にある

ものとし，点同士の衝突は考えないものとする。

〔プログラム1〕

```
public class Point {
   private int x, y, speed;
   public Point(int x, int y, int speed) {
      this.x = x; this.y = y;
      this.speed = speed;
   }
   public int getX() { return x; }
   public int getY() { return y; }
   public int getSpeed() { return speed; }
}
```

〔プログラム2〕

```
public abstract class Motion implements Runnable {
   private Point point;

   public Motion(Point point) {
      this.point = point;
   }

   public void run() {
      while (true) {
         Space.draw(point);
         try {
            Thread.sleep(40);
         } catch (InterruptedException e) {}
         Point current = point;
         [          a          ];
         Space.erase(current);
      }
   }

   public abstract Point update(Point point);
}
```

〔プログラム3〕

```
public class SimpleMotion extends Motion {
   private int directionX = 1, directionY = 1;

   public SimpleMotion(Point point) {
      super(point);
   }

   public Point update(Point point) {
      int speed = point.getSpeed();
      int x = point.getX() + directionX * speed;
      int y = point.getY() + directionY * speed;
      if (x <= 0) {
         x = -x;
         directionX *= -1;
      }
      x %= 2 * Space.getMaxX();
      if (x >= Space.getMaxX()) {
         x = 2 * Space.getMaxX() - x;
         directionX *= -1;
      }
      if (y <= 0) {
         y = -y;
         directionY *= -1;
      }
      y %= 2 * Space.getMaxY();
      if (y >= Space.getMaxY()) {
         y = 2 * Space.getMaxY() - y;
         directionY *= -1;
      }
      return new Point(x, y, speed);
   }

   public static void main(String[] args) {
      Point[] points = {
         new Point(10,  20, 3),
         new Point(50,  10, 5),
         new Point(150, 60, 2)
      };
      for (int i = 0; i < points.length; i++) {
         new Thread(          b          ).start();
      }
   }
}
```

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　`current = update(current)`　　イ　`current = update(point)`

ウ　`point = update(point)`　　エ　`update(current)`

オ　`update(point)`

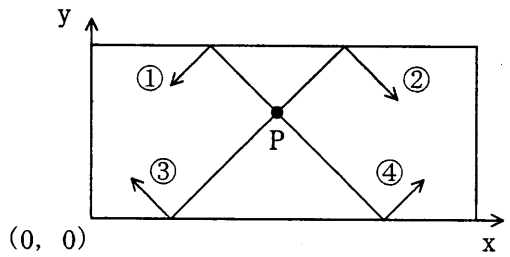
b に関する解答群

ア　`new Motion(points[i])`　　イ　`new Point(points[i])`

ウ　`new Runnable(points[i])`　　エ　`new SimpleMotion(points[i])`

オ　`points[i]`

設問2　クラス `SimpleMotion` のコンストラクタに与えられた `Point` が次の図の点 P で `speed` の値が 1 のとき，メソッド `run` を実行したときの点 P の動きとして正しい答えを，解答群の中から選べ。ただし，プログラム中の［　　］にはすべて正しい答えが入っているものとする。



解答群

ア　矢印①のように進む。

イ　矢印②のように進む。

ウ　矢印③のように進む。

エ　矢印④のように進む。

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〕

ある商店の在庫状況を記録した在庫表から，商品番号をキーとして，商品の在庫量を2分探索法で検索する副プログラム BSEARCH である。

(1) 在庫表は，先頭に扱い商品数N，その後ろに商品番号とその商品の在庫量の対を商品番号順（昇順）に格納したものである。扱い商品数，商品番号，在庫量は，それぞれ0以上の整数値で1語に格納されている。

在庫表の様式

N	商品番号[1]	在庫量[1]	商品番号[2]	在庫量[2]	…	商品番号[N]	在庫量[N]

(2) 主プログラムは，商品番号（検索キー）を GR1 に，在庫表の先頭アドレスを GR2 に設定し，BSEARCH を呼ぶ。

(3) BSEARCH は，検索キーに対応する在庫量を GR0 に設定し，主プログラムに戻る。該当する商品番号が在庫表に存在しない場合は，−1 を GR0 に設定し，主プログラムに戻る。

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

〔プログラム〕

(行番号)

```
 1   BSEARCH   START
 2             RPUSH
 3             LAD    GR3,1          ;  左端標識 LT の初期設定
 4             LD     GR4,0,GR2      ;  右端標識 RT の初期設定
 5             LAD    GR2,-1,GR2     ;  表の基底アドレス
 6             LAD    GR0,-1         ;  戻り値の初期設定
 7   SLOOP     CPL    GR3,GR4        ;  LT > RT ?
 8             JPL    EXIT
 9             LD     GR5,GR3        ;
10             ADDL   GR5,GR4        ;  } LT,RTの中央の位置 MID の計算
11             [   a   ]             ;
12             LD     GR6,GR5        ;
13             SLL    GR6,1          ;  } 商品番号 [MID] のアドレス計算
14             ADDL   GR6,GR2        ;
15             CPL    GR1,0,GR6      ;  KEY = 商品番号[MID] ?
```

```
16              JZE   FIND
17              [      b      ]
18              LAD   GR3,1,GR5     ;  MID+1 → LT
19              JUMP  SLOOP
20   RTSET      LAD   GR4,-1,GR5    ;  MID-1 → RT
21              JUMP  SLOOP
22   FIND       LD    GR0,1,GR6
23   EXIT       RPOP
24              RET
25              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　LAD　GR5,1,GR5　　イ　LAD　GR5,2,GR5　　ウ　SLL　GR5,1

エ　SLL　GR5,2　　オ　SRL　GR5,1

b に関する解答群

ア　JMI　RTSET　　イ　JMI　SLOOP　　ウ　JPL　RTSET

エ　JPL　SLOOP　　オ　LAD　GR2,1,GR2　　カ　LD　GR5,GR6

設問2　GR1 と GR2 を次のように設定して副プログラム BSEARCH を呼び出したとき，行番号 20 の命令は何回実行されるか，正しい答えを，解答群の中から選べ。

GR1 ［　　2　　］

8	1	100	3	300	5	500	7	700	8	800	9	90	11	110	13	130

↑
(GR2)

解答群

ア　0　　イ　1　　ウ　2　　エ　3　　オ　4

カ　7　　キ　8

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

なお，2問以上選択した場合には，はじめの1問について採点します。

問10 次のCプログラムの説明及びプログラムを読んで，設問1，2に答えよ。

〔プログラムの説明〕

関数make_fare_tableは，ある鉄道路線における始発駅から終着駅までの隣接駅間の距離から，その鉄道路線上の全駅間の運賃を求め，運賃表を作成するプログラムである。

(1) 関数make_fare_tableの引数は，次のとおりである。

num	始発駅から終着駅までの駅数
dist_list	隣接駅間の距離（km）を格納した配列（図1参照）
cost_list	距離運賃体系表（構造体COSTUNITの配列）
fare_table	運賃表 fare_table[m][n]（0≦m≦num−1，0≦n≦num−1かつm≠n）には，駅mと駅nの間の運賃を格納する。 fare_table[k][k]（0≦k≦num−1）には，ゼロを格納する。

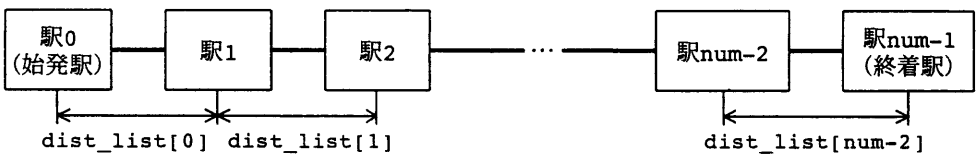


図1 引数numとdist_listの関係

(2) 距離から運賃を求めるために，関数calc_fareを用いる。

関数calc_fareの引数及び返却値は，次のとおりである。

dist	2駅間の距離（km）
cost_list	距離運賃体系表（構造体COSTUNITの配列）
返却値	運賃（円）

(3) 2駅間を幾つかの区間に分け，区間ごとに計算した運賃を合計したものを，2駅間の運賃とする。

(4) 各区間の運賃は，次の①～④のとおりである（図2参照）。

① 最初の20 km以下の区間（区間[0]）の運賃は，5 km増えるごとに100円加算する。

② 20 kmを超え100 km以下の区間（区間[1]）の運賃は，10 km増えるごとに180円加算する。

③ 100 kmを超え500 km以下の区間（区間[2]）の運賃は，50 km増えるごとに850円加算する。

④ 500 kmを超える区間（区間[3]）の運賃は，100 km増えるごとに1,650円加算する。

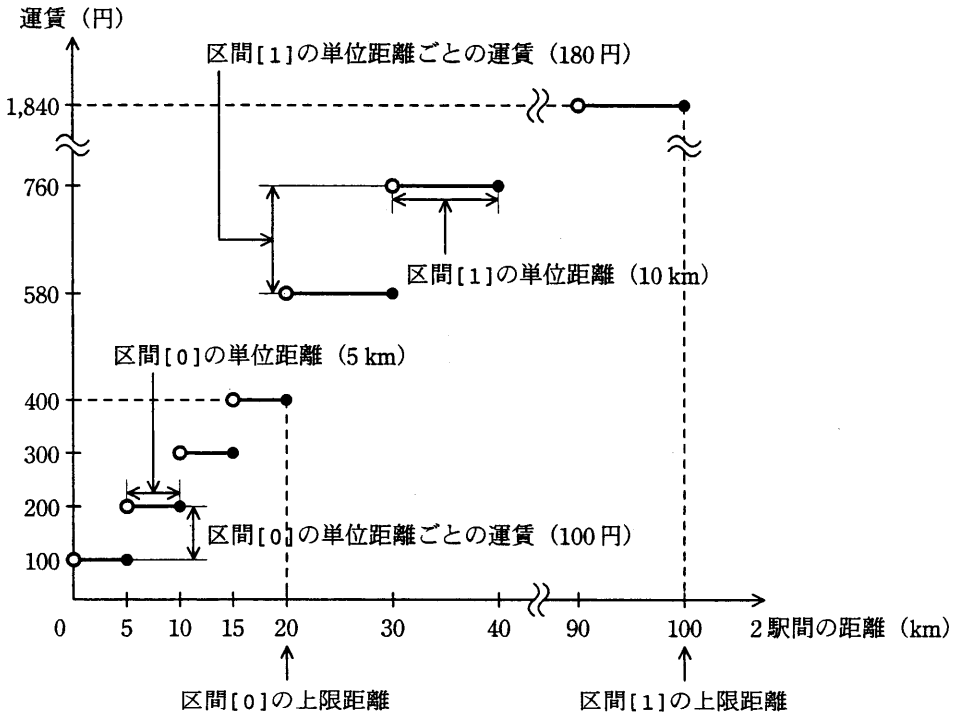


図2　2駅間の距離と運賃の関係

(5) 例えば，距離が 385 km ある駅間の運賃の求め方は，次のとおりである。

① 最初の 20 km 以下の区間（区間 [0]）の運賃は，5 km 増えるごとに 100 円加算されるので，次の式で求められる。

100 円×ceil(20 km÷5 km)=400 円

② 20 km を超え 100 km 以下の区間（区間 [1]）の運賃は，10 km 増えるごとに 180 円加算されるので，次の式で求められる。

180 円×ceil((100 km−20 km)÷10 km)=1,440 円

③ 100 km を超え 385 km 以下の区間（区間 [2]）の運賃は，50 km 増えるごとに 850 円加算されるので，次の式で求められる。

850 円×ceil((385 km−100 km)÷50 km)=5,100 円

運賃は，① ～ ③ の合計で，6,940 円になる。ここで，計算式中の ceil(X) は，X の小数点以下を切り上げた値を表す。

(6) 各区間の上限距離と単位距離は，距離運賃体系表で与える。距離運賃体系表は，上限距離の小さい区間から順に各要素を格納する。最後の要素の上限距離はゼロを格納し，距離の上限がないことを表す。

表　距離運賃体系表

区間 cost_list	上限距離（km） max_dist	単位距離（km） unit_dist	単位距離ごとの運賃（円） unit_cost
[0]	20	5	100
[1]	100	10	180
[2]	500	50	850
[3]	0	100	1,650

(7) 各区間の上限距離，単位距離及び単位距離ごとの運賃は，次に示す構造体 COSTUNIT で定義される。

```
typedef struct { int max_dist;  /* 上限距離（km）  */
                 int unit_dist; /* 単位距離（km）  */
                 int unit_cost; /* 単位距離ごとの運賃（円）  */
} COSTUNIT;
```

〔プログラム〕

```
#include <math.h>

typedef struct { int max_dist;  /* 上限距離 (km)  */
                 int unit_dist; /* 単位距離 (km)  */
                 int unit_cost; /* 単位距離ごとの運賃 (円)  */
               } COSTUNIT;

void make_fare_table(int, double *, COSTUNIT *, int **);
int calc_fare(double, COSTUNIT *);

void make_fare_table(int num, double *dist_list,
                     COSTUNIT *cost_list, int **fare_table) {
    int    idx0, idx1;
    double dist;

    for (idx0 = 0; idx0 < num; idx0++) {
        fare_table[idx0][idx0] = 0;
        dist = 0.0;
        for (idx1 = idx0 + 1; idx1 < num; idx1++) {
            [ a ];
α ➡         fare_table[idx0][idx1] = fare_table[idx1][idx0]
                                   = calc_fare(dist, cost_list);
        }
    }
}

int calc_fare(double dist, COSTUNIT *cost_list) {
    int fare = 0, idx = 0;
    int lower_limit;    /* 区間の下限 (直前の区間の上限距離)  */
    int upper_limit;    /* 区間の上限 (現在の区間の上限距離)  */
    lower_limit = 0;
    upper_limit = cost_list[0].max_dist;
    while ( [ b ] ) {
        fare += ceil( [ c ]
                    / (double)cost_list[idx].unit_dist)
                * cost_list[idx].unit_cost;
        lower_limit = upper_limit;
        upper_limit = cost_list[ [ d ] ].max_dist;
    }
    fare += ceil( [ e ]
                / (double)cost_list[idx].unit_dist)
            * cost_list[idx].unit_cost;
    return fare;
}
```

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　`dist += dist_list[idx0]`　　イ　`dist += dist_list[idx0 - 1]`

ウ　`dist += dist_list[idx1]`　　エ　`dist += dist_list[idx1 - 1]`

オ　`dist = dist_list[idx0]`　　カ　`dist = dist_list[idx0 - 1]`

キ　`dist = dist_list[idx1]`　　ク　`dist = dist_list[idx1 - 1]`

b に関する解答群

ア　`upper_limit != 0 && dist > (double)lower_limit`

イ　`upper_limit != 0 && dist > (double)upper_limit`

ウ　`upper_limit != 0 || dist > (double)lower_limit`

エ　`upper_limit != 0 || dist > (double)upper_limit`

オ　`upper_limit == 0 && dist <= (double)lower_limit`

カ　`upper_limit == 0 && dist <= (double)upper_limit`

キ　`upper_limit == 0 || dist <= (double)lower_limit`

ク　`upper_limit == 0 || dist <= (double)upper_limit`

c，e に関する解答群

ア　`((double)upper_limit - dist)`

イ　`(dist - (double)lower_limit)`

ウ　`(double)(upper_limit - lower_limit)`

エ　`(double)lower_limit`

オ　`(double)upper_limit`

カ　`dist`

d に関する解答群

ア　`++idx`　　イ　`--idx`　　ウ　`idx`　　エ　`idx++`

オ　`idx--`　　カ　`idx + 1`　　キ　`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。

鉄道路線の運営会社が，始発駅及び終着駅以外の途中駅を境に2社に分割され，運賃の計算方法が次のように変更された。

(1) どちらか1社の鉄道路線だけを利用する場合は，従来どおりの方法で運賃を計算する。

(2) 2社の鉄道路線を乗り継いで利用する場合は，各社の利用部分の運賃を個別に計算し，それを合計する。

これに対応するためには，関数 make_fare_table の引数に，境となる駅の番号 term_no（1 ≦ term_no ≦ num−2）を追加し，プログラム中の **α** を次のように変更すればよい。

```
if (idx0 < term_no && idx1 > term_no)
    fare_table[idx0][idx1] = fare_table[idx1][idx0]
                           = calc_fare(dist, cost_list)
                             + fare_table[idx0][term_no];
else
    fare_table[idx0][idx1] = fare_table[idx1][idx0]
                           = calc_fare(dist, cost_list);
if ([          ])
    dist = 0.0;
```

解答群

ア	idx0 == term_no - 1	イ	idx0 == term_no
ウ	idx0 == term_no + 1	エ	idx1 == term_no - 1
オ	idx1 == term_no	カ	idx1 == term_no + 1

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〕

ローン条件情報を受け取り，“返済回数指定の月払元金均等方式”で，ローン返済計画書を出力するサブプログラムである。

返済回数指定の月払元金均等方式とは，元金部分は返済回数による均等額，利息部分は元金残高に利率を乗じて算出し，それらの合計を毎月の返済額とする返済方式である。

(1) ローン条件情報（COND-R）のレコード様式は，次のとおりである。

ローン借入日 8けた	年利 5けた	毎月返済日 2けた	初回返済日 8けた	ローン元金 11けた	返済回数 3けた

① ローン借入日及び初回返済日は，4けたの西暦年，2けたの月，2けたの日で構成される8けたの日付とする（例：2005年7月1日の場合は，20050701）。

② 毎月返済日は，1～31の値とする。返済日に対応する日が存在しない月は，月末の日を返済日とする。

③ 年利は，％を単位とし，整数部2けた小数部3けたで構成される5けたの数値とする（例：年利5.0％の場合は05000）。返済回数は1～420とする。

④ ローン条件情報で与えられたデータには，誤りがないものとする。

(2) ローン返済計画書の出力例を次に示す。

借入日=2005/06/01　毎月返済日=31　年利= 5.000%　ローン元金=　5,000,000円

回数	返済日	日数	元金部分	利息部分	返済額	残高
1	2005/06/30	30	555,556	20,547	576,103	4,444,444
2	2005/07/31	31	555,556	18,873	574,429	3,888,888
3	2005/08/31	31	555,556	16,514	572,070	3,333,332
4	2005/09/30	30	555,556	13,698	569,254	2,777,776
5	2005/10/31	31	555,556	11,796	567,352	2,222,220
6	2005/11/30	30	555,556	9,132	564,688	1,666,664
7	2005/12/31	31	555,556	7,077	562,633	1,111,108
8	2006/01/31	31	555,556	4,718	560,274	555,552
9	2006/02/28	28	555,552	2,130	557,682	0
		273	5,000,000	104,485	5,104,485	

ローン返済計画書の見出し部分（出力例の1，2行目）は，あらかじめ印字され

ているものとする。

(3) 毎月返済額の元金部分及び利息部分の計算は，次のように行う。

① 毎月の元金返済に充てる金額は，“ローン元金÷返済回数”で求める。この元金返済部分を，毎月の返済ごとにローン元金から減算し，ローン元金残高を更新する。

② “ローン元金÷返済回数”が割り切れない場合，毎月の元金返済額は小数点以下を切上げとし，最終回の元金返済額は次の端数の金額となる。

端数＝ローン元金－毎月元金返済額×（返済回数－1）

③ 毎月の利息は，“ローン元金残高×（年利÷100）×日数÷365”で求め，割り切れない場合は切り捨てる。ここで日数は，最初の返済の場合はローン借入日から初回返済日までの日数，2回目以降は前回返済日の翌日から当月返済日までの日数とする。

(4) 関数 INTEGER-OF-DATE は，8けたの日付をグレゴリオ暦で1601年1月1日を1とする通算日に変換した整数値を返す。

(5) 日付が正しいかどうかを調べるために，副プログラム DATECHK を使用する。DATECHK は，与えられた日付がカレンダーにある正しい日付かどうかを調べ，正しければ0，正しくなければ1を結果に返す。DATECHK の呼出し方法は，次のとおりである。

```
CALL "DATECHK" USING 日付 結果
```

〔プログラム〕

(行番号)

```
 1    DATA DIVISION.
 2    FILE SECTION.
 3    FD  PRINT-F                    EXTERNAL.
 4    01  PRINT-R                    PIC  X(128).
 5    WORKING-STORAGE SECTION.
 6    01  W-KAISU                    PIC  9(04).
 7    01  W-AMARI                    PIC  9(05).
 8    01  W-GANKIN-KINTOU            PIC  9(15).
 9    01  W-GANKIN-HASUU             PIC  9(15).
10    01  W-REC.
11        05  W-NISSU                PIC  9(04).
12        05  W-GANKIN               PIC  9(15).
13        05  W-RISOKU               PIC  9(15).
14        05  W-HENSAI-GAKU          PIC  9(15).
```

```
15          05  W-ZANDAKA              PIC  9(15).
16      01  MEI-R.
17          05  M-KAISU                PIC  ZZZ9.
18          05  FILLER                 PIC  X(03).
19          05  M-HENSAI-YMD           PIC  9999/99/99.
20          05  FILLER                 PIC  X.
21          05  M-NISSU                PIC  ZZZZ9.
22          05  FILLER                 PIC  X.
23          05  M-GANKIN               PIC  ZZZ,ZZZ,ZZ9.
24          05  FILLER                 PIC  X.
25          05  M-RISOKU               PIC  ZZZ,ZZZ,ZZ9.
26          05  FILLER                 PIC  X.
27          05  M-HENSAI-GAKU          PIC  ZZZ,ZZZ,ZZ9.
28          05  FILLER                 PIC  X.
29          05  M-ZANDAKA              PIC  ZZZ,ZZZ,ZZ9.
30      01  GOUKEI.
31          05  G-NISSU                PIC  9(05).
32          05  G-GANKIN               PIC  9(15).
33          05  G-RISOKU               PIC  9(15).
34          05  G-HENSAI-GAKU          PIC  9(15).
35      01  W-HENSAI-YMD               PIC  9(08).
36      01  W-HENSAI-YMD-R  REDEFINES W-HENSAI-YMD.
37          05  W-HENSAI-YY            PIC  9(04).
38          05  W-HENSAI-MM            PIC  9(02).
39          05  W-HENSAI-DD            PIC  9(02).
40      01  JDATE1                     PIC  9(15).
41      01  JDATE2                     PIC  9(15).
42      01  W-RETCD                    PIC  9.
43      LINKAGE SECTION.
44      01  COND-R.
45          03  CD-KARIIRE-YMD         PIC  9(08).
46          03  CD-NENRI               PIC  9(02)V9(03).
47          03  CD-MAITSUKI-DD         PIC  9(02).
48          03  CD-SHOKAI-YMD          PIC  9(08).
49          03  CD-GANKIN              PIC  9(11).
50          03  CD-KAISU               PIC  9(03).
51      PROCEDURE DIVISION USING COND-R.
52      MAIN-RTN.
53          PERFORM INIT-RTN.
54          PERFORM LOAN-RTN [          a          ].
55          PERFORM GOUKEI-PRINT.
56          EXIT PROGRAM.
57      INIT-RTN.
58          INITIALIZE GOUKEI.
59          DIVIDE CD-GANKIN BY CD-KAISU GIVING W-GANKIN-KINTOU
60                 REMAINDER W-AMARI.
61          IF W-AMARI > 0 THEN
62             ADD 1 TO W-GANKIN-KINTOU
63          END-IF.
64          COMPUTE W-GANKIN-HASUU = CD-GANKIN -
65                  W-GANKIN-KINTOU * (CD-KAISU - 1).
66      LOAN-RTN.
```

COBOL

```
67          IF W-KAISU = 1 THEN
68      *      最初の返済時
69             MOVE    CD-GANKIN      TO W-ZANDAKA
70             MOVE    CD-SHOKAI-YMD TO W-HENSAI-YMD
71             COMPUTE JDATE1 =
72                     FUNCTION INTEGER-OF-DATE(CD-KARIIRE-YMD)
73             COMPUTE JDATE2 =
74                     FUNCTION INTEGER-OF-DATE(CD-SHOKAI-YMD)
75             COMPUTE W-NISSU = [          b          ]
76          ELSE
77      *      2 回目以降の返済時
78             COMPUTE JDATE1 =
79                     FUNCTION INTEGER-OF-DATE(W-HENSAI-YMD)
80             PERFORM NEXT-HENSAIBI-RTN
81             COMPUTE JDATE2 =
82                     FUNCTION INTEGER-OF-DATE(W-HENSAI-YMD)
83             COMPUTE W-NISSU = [          c          ]
84          END-IF.
85          COMPUTE W-RISOKU = [          d          ].
86          IF W-KAISU = CD-KAISU
87             MOVE W-GANKIN-HASUU  TO W-GANKIN
88          ELSE
89             MOVE W-GANKIN-KINTOU TO W-GANKIN
90          END-IF.
91          COMPUTE W-HENSAI-GAKU = W-RISOKU + W-GANKIN.
92          SUBTRACT W-GANKIN FROM W-ZANDAKA.
93          MOVE SPACE           TO MEI-R.
94          MOVE W-KAISU         TO M-KAISU.
95          MOVE W-HENSAI-YMD    TO M-HENSAI-YMD.
96          MOVE W-NISSU         TO M-NISSU.
97          MOVE W-GANKIN        TO M-GANKIN.
98          MOVE W-RISOKU        TO M-RISOKU.
99          MOVE W-HENSAI-GAKU   TO M-HENSAI-GAKU.
100         MOVE W-ZANDAKA       TO M-ZANDAKA.
101         WRITE PRINT-R FROM   MEI-R.
102         ADD  W-NISSU         TO G-NISSU.
103         ADD  W-GANKIN        TO G-GANKIN.
104         ADD  W-RISOKU        TO G-RISOKU.
105         ADD  W-HENSAI-GAKU   TO G-HENSAI-GAKU.
106      NEXT-HENSAIBI-RTN.
107     *     翌月の返済日を求める
108         MOVE CD-MAITSUKI-DD TO W-HENSAI-DD.
109         ADD  1              TO W-HENSAI-MM.
110         IF W-HENSAI-MM > 12 THEN
111            ADD   1          TO W-HENSAI-YY
112            MOVE  1          TO W-HENSAI-MM
113         END-IF.
114     *     月末の存在しない日付の調整
115         MOVE  1 TO    W-RETCD.
116         PERFORM UNTIL W-RETCD = 0
117            CALL "DATECHK" USING W-HENSAI-YMD W-RETCD
```

COBOL

```
118            IF W-RETCD NOT = 0 THEN
119              [          e          ]
120            END-IF
121          END-PERFORM.
122      GOUKEI-PRINT.
123          MOVE SPACE          TO MEI-R.
124          MOVE G-NISSU        TO M-NISSU.
125          MOVE G-GANKIN       TO M-GANKIN.
126          MOVE G-RISOKU       TO M-RISOKU.
127          MOVE G-HENSAI-GAKU TO M-HENSAI-GAKU.
128          WRITE PRINT-R FROM MEI-R.
```

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　CD-KAISU TIMES

イ　VARYING W-KAISU FROM 1 BY 1 UNTIL W-KAISU < CD-KAISU

ウ　VARYING W-KAISU FROM 1 BY 1 UNTIL W-KAISU = CD-KAISU

エ　VARYING W-KAISU FROM 1 BY 1 UNTIL W-KAISU > CD-KAISU

b, c に関する解答群

ア　JDATE2 + JDATE1

イ　JDATE2 - CD-SHOKAI-YMD

ウ　JDATE2 - JDATE1

エ　JDATE2 - JDATE1 + 1

オ　JDATE2 - W-HENSAI-YMD

d に関する解答群

ア　CD-GANKIN * JDATE2 * CD-NENRI / 100 / 365

イ　CD-GANKIN * W-NISSU * CD-NENRI / 100 / 365

ウ　W-ZANDAKA * JDATE2 * CD-NENRI / 100 / 365

エ　W-ZANDAKA * W-NISSU * CD-NENRI / 100 / 365

e に関する解答群

ア　ADD　　1 TO　W-HENSAI-DD

イ　ADD　　1 TO　W-HENSAI-YY

ウ　SUBTRACT 1 FROM W-HENSAI-DD

エ　SUBTRACT 1 FROM W-HENSAI-MM

設問2　このプログラムでは，ローン元金が返済回数で割り切れない場合の端数調整を最終回の返済で行っている。パラメタ指定によって初回の返済でも調整できるようにプログラムを変更したい。行番号 50 と 51 の間に次の行を挿入し，ローン条件情報に端数調整方法を追加する。この値が 1 のときは初回調整，0 のときは最終回調整とする。

```
03  CD-HASUU              PIC  9.
```

あわせて，プログラム中の行番号 86 を変更すべき内容として正しい答えを，解答群の中から選べ。

初回調整の場合のローン返済計画書の出力例を，次に示す。

借入日=2005/06/01　毎月返済日=31 年利= 5.000% ローン元金=　5,000,000円

回数	返済日	日数	元金部分	利息部分	返済額	残高
1	2005/06/30	30	555,552	20,547	576,099	4,444,448
2	2005/07/31	31	555,556	18,873	574,429	3,888,892
3	2005/08/31	31	555,556	16,514	572,070	3,333,336
4	2005/09/30	30	555,556	13,698	569,254	2,777,780
5	2005/10/31	31	555,556	11,796	567,352	2,222,224
6	2005/11/30	30	555,556	9,132	564,688	1,666,668
7	2005/12/31	31	555,556	7,077	562,633	1,111,112
8	2006/01/31	31	555,556	4,718	560,274	555,556
9	2006/02/28	28	555,556	2,130	557,686	0
		273	5,000,000	104,485	5,104,485	

解答群

ア
```
IF CD-HASUU = 0 AND W-KAISU = CD-KAISU THEN
```

イ
```
IF CD-HASUU = 1 AND W-KAISU = 1 THEN
```

ウ
```
IF CD-HASUU = 0 AND W-KAISU = CD-KAISU AND
   CD-HASUU = 1 AND W-KAISU = 1 THEN
```

エ
```
IF CD-HASUU = 0 AND W-KAISU = CD-KAISU OR
   CD-HASUU = 1 AND W-KAISU = 1 THEN
```

オ
```
IF W-AMARI NOT = 0 OR
   CD-HASUU = 0 AND W-KAISU = CD-KAISU OR
   CD-HASUU = 1 AND W-KAISU = 1 THEN
```

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〕

整数値の加減乗除の演算をする電卓プログラムである。入出力部分はテスト用のプログラムによって用意され，電卓本体部分のプログラムのテストができるようになっている。

(1) クラス CalculatorEvent は電卓のキーが押されたときに発生するイベントである。イベントにはタイプがありフィールド type の値で表される。タイプは DIGIT，OPERATOR，CLEAR のいずれかであり，それぞれ電卓の数字キー（0～9），演算キー（+ など）及びイコールキー（=），クリアキー（C）を表す。タイプが DIGIT のときは数字キーに対応する数値を，OPERATOR のときは演算の種類を表す文字又は '=' をフィールド value に保持する。タイプが CLEAR のときは，value は使用しない。

(2) インタフェース CalculatorOutput は，電卓上に数値やエラーを表示するメソッド display を宣言する。

(3) クラス Calculator は電卓本体である。メソッド eventDispatched はイベントを受け取り，イベントのタイプに応じて演算処理などを行う。

なお，二つの数値に対する加減乗除の演算結果はJavaのint型の演算結果に一致するものとする。

(4) クラス CalculatorTest は Calculator をテストするプログラムである。CalculatorOutput を匿名クラスとして実装する。この実装では，メソッド display は指定された数値又は文字列を System.out に出力する。メソッド main は，引数 args[0] で与えられた文字列から CalculatorEvent を生成して，Calculator のメソッド eventDispatched を呼び出す。文字と電卓のキーの対応は，次の表のとおりである。

文字	電卓キー
'0' ～ '9'	数字キー（0 ～ 9）
'+'	加算キー（+）
'-'	減算キー（−）
'*'	乗算キー（×）
'/'	除算キー（÷）
'='	イコールキー（=）
'C'	クリアキー（C）

例えば，文字列“2+7=”は，電卓キー 2，+，7，= が順に押されたことを表し，この文字列が引数 args[0] としてメソッド main に渡されたとき，プログラムは次のとおり出力する。

```
2
2
7
9
```

〔プログラム 1〕

```
public class CalculatorEvent {
   public static final int DIGIT = 1;
   public static final int OPERATOR = 2;
   public static final int CLEAR = 3;

   private int type, value;

   public CalculatorEvent(int type) {
      [          a          ];
   }
   public CalculatorEvent(int type, int value) {
      if (type < DIGIT || type > CLEAR)
         throw new IllegalArgumentException();
      this.type = type; this.value = value;
   }
   public int getType() { return type; }
   public int getValue() { return value; }
}
```

〔プログラム2〕

```
public interface CalculatorOutput {
   public void display(int value);
   public void display(String value);
}
```

〔プログラム3〕

```
public class Calculator {
   private int accumulator = 0, register = 0;
   private int operator = 0;
   private CalculatorOutput output;

   public Calculator(CalculatorOutput output) {
      this.output = output;
   }

   public void eventDispatched(CalculatorEvent event) {
      switch (event.getType()) {
      case CalculatorEvent.DIGIT:
         if (operator == '=') {
            register = 0; operator = 0;
         }
         register = register * 10 + event.getValue();
         output.display(register);
         break;
      case CalculatorEvent.OPERATOR:
         try {
            register = calculate();
            output.display(register);
            accumulator = register;
            operator = event.getValue();
         } catch (ArithmeticException e) {
            output.display("Error");
            accumulator = 0; operator = 0;
         }
         if (operator != '=')
            register = 0;
         break;
      case CalculatorEvent.CLEAR:
         register = 0;
         accumulator = 0;
         operator = 0;
         output.display(register);
         break;
      }
   }
```

```
    private int calculate() {
        switch (operator) {
        case '+':
            return accumulator + register;
        case '-':
            return accumulator - register;
        case '*':
            return accumulator * register;
        case '/':
            return accumulator / register;
        }
        return register;
    }
}
```

〔プログラム4〕

```
public class CalculatorTest {
    public static void main(String[] args) {
        Calculator calc = new Calculator(
            [ b ] {
                public void display(int value) {
                    System.out.println(value);
                }
                public void display(String value) {
                    System.out.println(value);
                }
            });
        String keys = args[0];
        for (int i = 0; i < keys.length(); i++) {
            char c = keys.charAt(i);
            CalculatorEvent event = null;
            if (c >= '0' && c <= '9') {
                event = new CalculatorEvent(
                                    [ c ]);
            } else if (c == '=' || c == '+' || c == '-'
                       || c == '*' || c == '/') {
                event = new CalculatorEvent(
                                    CalculatorEvent.OPERATOR, c);
            } else if (c == 'C') {
                event = new CalculatorEvent(
                                    CalculatorEvent.CLEAR);
            }
            if (event != null)
                calc.eventDispatched(event);
        }
    }
}
```

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　`CalculatorEvent(type, 0)`

イ　`new CalculatorEvent(type, 0)`

ウ　`return new CalculatorEvent(type, 0)`

エ　`super(type, 0)`

オ　`this(type, 0)`

bに関する解答群

ア　`implements CalculatorOutput()`

イ　`interface CalculatorOutput()`

ウ　`new CalculatorOutput()`

エ　`new Temp() implements CalculatorOutput`

オ　`public class Temp implements CalculatorOutput`

cに関する解答群

ア　`c - '0', CalculatorEvent.DIGIT`

イ　`c, CalculatorEvent.DIGIT`

ウ　`CalculatorEvent.DIGIT`

エ　`CalculatorEvent.DIGIT, c`

オ　`CalculatorEvent.DIGIT, c - '0'`

設問2　次の表は文字列を引数としてメソッド main を実行したときに，最後に出力された結果を表している。表中の［　　］に入れる正しい答えを，解答群の中から選べ。ただし，プログラム中の［　　］にはすべて正しい答えが入っているものとする。

文字列	出力
3+4*5=	35
3*4***=	d
3*4=+5	e
3+4/0=	f

解答群

ア　0	イ　3	ウ　4	エ　5
オ　7	カ　12	キ　17	ク　53
ケ　/ by zero	コ　Error		

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〕

10進数の読取りを行う副プログラム NREAD と，NREAD から呼ばれる副プログラム DTOB である。

(1) NREAD は，入力装置から1レコードを読み取り，読み込んだ文字列を2進数に変換し，結果を GR0 に格納して，主プログラムに戻る。

(2) NREAD は，入力された文字列中の間隔文字を読み飛ばし，間隔文字以外について，次の条件を検査する。

① 最初の文字は数字又は負符号（－）

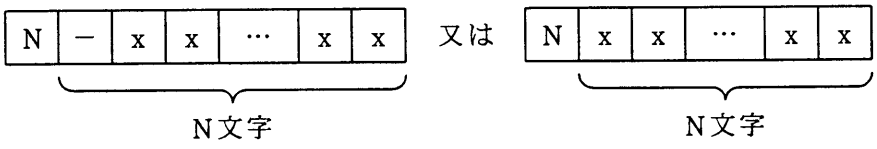
② 2番目以降の文字はすべて数字

③ 一つ以上の数字を含む。

条件を満たしていないとき，GR0 に －32768 を設定して主プログラムに戻る。条件を満たしているとき，DTOB を呼び出し，DTOB の戻り値 GR0 をそのまま NREAD の戻り値とする。

(3) DTOB は，先頭アドレスを GR1 に設定して渡された数字列を2進数に変換し，GR0 に格納して呼出し元に戻る。このとき，変換結果が －32767 ～ 32767 に収まらない場合はエラーとし，GR0 に －32768 を設定して呼出し元に戻る。

〔数字列の形式〕



N ： 文字数（N＞0）

－ ： 負符号

x ： 数字

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

アセンブラ

〔プログラム 1〕

```
(行番号)
 1  NREAD    START
 2           RPUSH
 3           IN      INBUF,LENG      ; 入力装置から 1 レコード読込み
 4           LD      GR2,LENG
 5           JZE     ERR
 6           JMI     ERR
 7           LAD     GR1,INBUF
 8           ADDL    GR2,GR1         ; 入力データ終端アドレスの設定
 9           LAD     GR3,0           ; 先頭文字確認フラグの初期化
10           LAD     GR5,1           ; NUMSTR 領域カウンタの初期化
11           LAD     GR6,0           ; 数字カウンタの初期化
12  LOOP     CPL     GR1,GR2         ; 入力データ終端？
13           [       a               ]
14           LD      GR4,0,GR1       ; 1 文字の取出し
15           CPL     GR4,=' '        ; 間隔文字ならばスキップ
16           JZE     NEXT
17           LD      GR3,GR3         ; （間隔文字以外の）先頭文字？
18           JNZ     NUMCHK
19           LAD     GR3,1
20           CPL     GR4,='-'
21           JNZ     NUMCHK
22           ST      GR4,NUMSTR,GR5  ; 文字'-'の格納
23           LAD     GR5,1,GR5
24           JUMP    NEXT
25  NUMCHK   CPL     GR4,='9'        ; 数字？
26           JPL     ERR
27           CPL     GR4,='0'
28           JMI     ERR
29           ST      GR4,NUMSTR,GR5
30           LAD     GR5,1,GR5
31           LAD     GR6,1,GR6
32  NEXT     LAD     GR1,1,GR1
33           JUMP    LOOP
34  ERR      LAD     GR0,-32768      ; エラー時の処理
35           JUMP    EXIT
36  CNV      LD      GR6,GR6
37           JZE     ERR
38           LAD     GR5,-1,GR5
39           ST      GR5,NUMSTR      ; 文字数の格納
40           LAD     GR1,NUMSTR
41           CALL    DTOB            ; DTOB 呼出し
42  EXIT     RPOP
43           RET
44  INBUF    DS      256             ; 入力データ領域
45  LENG     DS      1
46  NUMSTR   DS      257             ; DTOB 呼出しパラメタ領域
47           END
```

〔プログラム 2〕

```
(行番号)
 1  DTOB    START
 2          RPUSH
 3          LD     GR2,0,GR1   ; 文字数の取出し
 4          ADDL   GR2,GR1     ; 文字列終端アドレスの設定
 5          LAD    GR0,0       ; 戻り値の初期化
 6          LAD    GR3,0       ; 負符号フラグの初期設定
 7          LD     GR4,1,GR1   ; 最初の文字の取出し
 8          CPL    GR4,='-'
 9          JNZ    LOOP
10          LAD    GR3,1       ; 負数であることを記憶
11          LAD    GR1,1,GR1
12  LOOP    LAD    GR1,1,GR1
13          CPL    GR1,GR2     ; 変換終了？
14          JPL    FIN
15          LD     GR4,0,GR1   ; 数字 1 文字の取出し
16          SUBL   GR4,='0'    ; 1 文字を数値に変換
17          SLL    GR0,1       ;
18          JMI    ERR         ;
19          LD     GR5,GR0     ;
20          SLL    GR0,1       ;
21          JMI    ERR         ; } GR0（上位けたまでの変換値）を 10 倍する
22          SLL    GR0,1       ;   〔行番号 20 ～ 22 は設問 3 参照〕
23          JMI    ERR         ;
24          [      b       ]   ;
25          JMI    ERR         ;
26          ADDL   GR0,GR4
27          JMI    ERR
28          JUMP   LOOP
29  ERR     LAD    GR0,-32768  ; −32767 ～ 32767の範囲外ならエラー
30          JUMP   EXIT
31  FIN     LD     GR3,GR3     ; 負符号フラグチェック
32          JZE    EXIT
33          [      c       ]
34          ADDA   GR0,=1
35  EXIT    RPOP
36          RET
37          END
```

アセンブラ

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　JNZ　NEXT　　イ　JNZ　NUMCHK　　ウ　JPL　CNV

エ　JPL　EXIT　　オ　JZE　CNV　　カ　JZE　EXIT

b に関する解答群

ア　ADDL　GR0,GR4　　イ　ADDL　GR0,GR5　　ウ　ADDL　GR5,GR0

エ　ADDL　GR5,GR4　　オ　SLL　GR0,1　　カ　SLL　GR4,1

キ　SLL　GR4,2　　ク　SLL　GR5,1

c に関する解答群

ア　AND　GR0,=#7FFF　　イ　AND　GR0,GR3

ウ　OR　GR0,=#8000　　エ　OR　GR0,GR3

オ　XOR　GR0,=#8000　　カ　XOR　GR0,=#FFFF

キ　XOR　GR0,GR3

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。

IN 命令によって，文字列“△-5△24-△9”（△は間隔文字）が入力されたとき，〔プログラム1〕の ERR へは，行番号［　d　］の命令から制御が移る。ERR に制御が移ったときの GR5 の値は［　e　］である。

d に関する解答群

ア　5　　イ　6　　ウ　26　　エ　28　　オ　37

e に関する解答群

ア　1　　イ　2　　ウ　4　　エ　5　　オ　6

カ　7　　キ　8

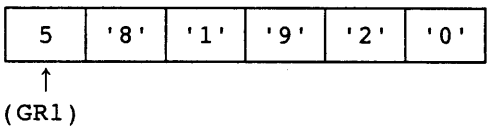
アセンブラ

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。

次の数字列がNREADから渡されたとき，DTOBの戻り値は16進表記で［　f　］である。ここで，DTOBの一部である行番号20～22の3行を“SLL　GR0,2”の1行で置き換える。同じ文字列がNREADから渡されたとき，DTOBの戻り値は16進表記で［　g　］となる。

なお，いずれの場合も，〔プログラム2〕の行番号15で最後の文字'0'を取り出したとき，GR0の値は10進表記で8192（16進表記で2000）となっている。

NREADから渡された数字列



解答群

ア　0　　イ　2000　　ウ　4000　　エ　8000　　オ　F000

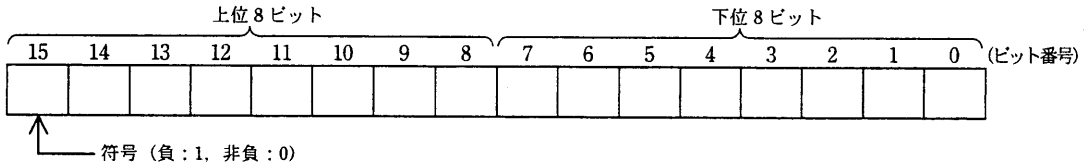
アセンブラ

■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，それ以外のとき 0 になる。論理演算命令の場合は，演算結果が 0 ～ 65535 に収まらなくなったとき 1 になり，それ以外のとき 0 になる。
SF	演算結果の符号が負（ビット番号 15 が 1）のとき 1，それ以外のとき 0 になる。
ZF	演算結果が零（全部のビットが 0）のとき 1，それ以外のとき 0 になる。

(6) 論理加算又は論理減算は，被演算データを符号のない数値とみなして，加算又は減算する。

1.2 命令

命令の形式及びその機能を示す。ここで，一つの命令コードに対し 2 種類のオペランドがある場合，上段はレジスタ間の命令，下段はレジスタと主記憶間の命令を表す。

命　令	書き方		命　令　の　説　明	FRの設定
	命令コード	オペランド		

(1) ロード，ストア，ロードアドレス命令

ロード LoaD	LD	r1,r2	r1 ← (r2)	○*1
		r,adr [,x]	r ← (実効アドレス)	
ストア STore	ST	r,adr [,x]	実効アドレス ← (r)	—
ロードアドレス Load ADdress	LAD	r,adr [,x]	r ← 実効アドレス	

(2) 算術，論理演算命令

算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	
論理積 AND	AND	r1,r2	r1 ← (r1) AND (r2)	○*1
		r,adr [,x]	r ← (r) AND (実効アドレス)	
論理和 OR	OR	r1,r2	r1 ← (r1) OR (r2)	
		r,adr [,x]	r ← (r) OR (実効アドレス)	
排他的論理和 eXclusive OR	XOR	r1,r2	r1 ← (r1) XOR (r2)	
		r,adr [,x]	r ← (r) XOR (実効アドレス)	

(3) 比較演算命令

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

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

(4) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	r,adr [,x]	符号を除き (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には，左シフトのときは 0，右シフトのときは符号と同じものが入る。	○*2
算術右シフト Shift Right Arithmetic	SRA	r,adr [,x]		
論理左シフト Shift Left Logical	SLL	r,adr [,x]	符号を含み (r) を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には 0 が入る。	
論理右シフト Shift Right Logical	SRL	r,adr [,x]		

(5) 分岐命令

正分岐 Jump on PLus	JPL	adr [,x]	FR の値によって，実効アドレスに分岐する。分岐しないときは，次の命令に進む。	—
負分岐 Jump on MInus	JMI	adr [,x]		
非零分岐 Jump on Non Zero	JNZ	adr [,x]		
零分岐 Jump on ZEro	JZE	adr [,x]		
オーバフロー分岐 Jump on OVerflow	JOV	adr [,x]		
無条件分岐 unconditional JUMP	JUMP	adr [,x]	無条件に実効アドレスに分岐する。	

命令	分岐するときの FR の値 OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

(6) スタック操作命令

プッシュ PUSH	PUSH adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← 実効アドレス	—
ポップ POP	POP r	r ← ((SP)), SP ← (SP) $+_L$ 1	

(7) コール，リターン命令

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

(8) その他

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

(注) r, r1, r2　いずれも GR を示す。指定できる GR は GR0 ～ GR7
adr　アドレスを示す。指定できる値の範囲は 0 ～ 65535
x　指標レジスタとして用いる GR を示す。指定できる GR は GR1 ～ GR7
[　]　[　] 内の指定は省略できることを示す。
(　)　(　) 内のレジスタ又はアドレスに格納されている内容を示す。
実効アドレス　adr と x の内容との論理加算値又はその値が示す番地
←　演算結果を，左辺のレジスタ又はアドレスに格納することを示す。
$+_L$, $-_L$　論理加算，論理減算を示す。
FR の設定　○　：設定されることを示す。
○*1：設定されることを示す。ただし，OF には 0 が設定される。
○*2：設定されることを示す。ただし，OF にはレジスタから最後に送り出されたビットの値が設定される。
－　：実行前の値が保持されることを示す。

1.3 文字の符号表

(1) JIS X 0201 ラテン文字・片仮名用 8 ビット符号で規定する文字の符号表を使用する。
(2) 右に符号表の一部を示す。1 文字は 8 ビットからなり，上位 4 ビットを列で，下位 4 ビットを行で示す。例えば，間隔，4，H，¥ のビット構成は，16 進表示で，それぞれ 20，34，48，5C である。16 進表示で，ビット構成が 21 ～ 7E（及び表では省略している A1 ～ DF）に対応する文字を図形文字という。図形文字は，表示（印刷）装置で，文字として表示（印字）できる。
(3) この表にない文字とそのビット構成が必要な場合は，問題中で与える。

行＼列	02	03	04	05	06	07
0	間隔	0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[	k	{
12	,	<	L	¥	l	|
13	-	=	M	]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

2. アセンブラ言語 CASLⅡの仕様

2.1 言語の仕様

(1) CASLⅡは，COMETⅡのためのアセンブラ言語である。
(2) プログラムは，命令行及び注釈行からなる。
(3) 1命令は1命令行で記述し，次の行へ継続できない。
(4) 命令行及び注釈行は，次に示す記述の形式で，行の1文字目から記述する。

行の種類		記述の形式
命令行	オペランドあり	[ラベル] {空白} {命令コード} {空白} {オペランド} [{空白} [コメント]]
	オペランドなし	[ラベル] {空白} {命令コード} [{空白} [{;} [コメント]]]
注釈行		[空白] {;} [コメント]

(注) [ ] [ ] 内の指定が省略できることを示す。
{ } { } 内の指定が必須であることを示す。
ラベル その命令の（先頭の語の）アドレスを他の命令やプログラムから参照するための名前である。長さは 1～8 文字で，先頭の文字は英大文字でなければならない。以降の文字は，英大文字又は数字のいずれでもよい。なお，予約語である GR0～GR7 は，使用できない。
空白 1文字以上の間隔文字の列である。
命令コード 命令ごとに記述の形式が定義されている。
オペランド 命令ごとに記述の形式が定義されている。
コメント 覚え書きなどの任意の情報であり，処理系で許す任意の文字を書くことができる。

2.2 命令の種類

命令は，4種類のアセンブラ命令（START，END，DS，DC），4種類のマクロ命令（IN，OUT，RPUSH，RPOP）及び機械語命令（COMETⅡの命令）からなる。その仕様を次に示す。

命令の種類	ラベル	命令コード	オペランド	機能
アセンブラ命令	ラベル	START	[実行開始番地]	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
		END		プログラムの終わりを明示
	[ラベル]	DS	語数	領域を確保
	[ラベル]	DC	定数 [,定数] …	定数を定義
マクロ命令	[ラベル]	IN	入力領域,入力文字長領域	入力装置から文字データを入力
	[ラベル]	OUT	出力領域,出力文字長領域	出力装置へ文字データを出力
	[ラベル]	RPUSH		GR の内容をスタックに格納
	[ラベル]	RPOP		スタックの内容を GR に格納
機械語命令	[ラベル]	（「1.2 命令」を参照）		

2.3 アセンブラ命令

アセンブラ命令は，アセンブラの制御などを行う。

(1) START [実行開始番地]

START 命令は，プログラムの先頭を定義する。

実行開始番地は，そのプログラム内で定義されたラベルで指定する。指定がある場合はその番地から，省略した場合は START 命令の次の命令から，実行を開始する。

また，この命令につけられたラベルは，他のプログラムから入口名として参照できる。

(2) | END | |

END 命令は，プログラムの終わりを定義する。

(3) | DS | 語数 |

DS 命令は，指定した語数の領域を確保する。

語数は，10 進定数（≧0）で指定する。語数を 0 とした場合，領域は確保しないが，ラベルは有効である。

(4) | DC | 定数［,定数］… |

DC 命令は，定数で指定したデータを（連続する）語に格納する。

定数には，10 進定数，16 進定数，文字定数，アドレス定数の 4 種類がある。

定数の種類	書き方	命　令　の　説　明
10 進定数	n	n で指定した 10 進数値を，1 語の 2 進数データとして格納する。ただし，n が−32768 ～ 32767 の範囲にないときは，その下位 16 ビットを格納する。
16 進定数	#h	h は 4 けたの 16 進数（16 進数字は 0 ～ 9，A ～ F）とする。h で指定した 16 進数値を 1 語の 2 進数データとして格納する（0000 ≦ h ≦ FFFF）。
文字定数	'文字列'	文字列の文字数（＞0）分の連続する領域を確保し，最初の文字は第 1 語の下位 8 ビットに，2 番目の文字は第 2 語の下位 8 ビットに，…と順次文字データとして格納する。各語の上位 8 ビットには 0 のビットが入る。 文字列には，間隔及び任意の図形文字を書くことができる。ただし，アポストロフィ（'）は 2 個続けて書く。
アドレス定数	ラベル	ラベルに対応するアドレスを 1 語の 2 進数データとして格納する。

2.4 マクロ命令

マクロ命令は，あらかじめ定義された命令群とオペランドの情報によって，目的の機能を果たす命令群を生成する（語数は不定）。

(1) | IN | 入力領域,入力文字長領域 |

IN 命令は，あらかじめ割り当てた入力装置から，1 レコードの文字データを読み込む。

入力領域は，256 語長の作業域のラベルであり，この領域の先頭から，1 文字を 1 語に対応させて順次入力される。レコードの区切り符号（キーボード入力の復帰符号など）は，格納しない。格納の形式は，DC 命令の文字定数と同じである。入力データが 256 文字に満たない場合，入力領域の残りの部分は実行前のデータを保持する。入力データが 256 文字を超える場合，以降の文字は無視される。

入力文字長領域は，1 語長の領域のラベルであり，入力された文字の長さ（≧0）が 2 進数で格納される。ファイルの終わり（end of file）を検出した場合は，−1 が格納される。

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

(2) | OUT | 出力領域,出力文字長領域 |

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

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

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

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

(3) | RPUSH |

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

(4) | RPOP |

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

2.5 機械語命令

機械語命令のオペランドは，次の形式で記述する。

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

2.6 その他

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

(2) 生成された命令語，領域は，主記憶上で連続した領域を占める。

3. プログラム実行の手引

3.1 OS

プログラムの実行に関して，次の取決めがある。

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

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

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

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

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

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

3.2 未定義事項

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

〔 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。

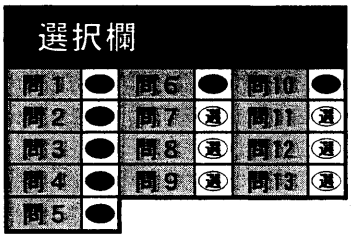
(2) 答案用紙は光学式読取り装置で処理しますので，答案用紙の**マークの記入方法**のとおりマークしてください。

(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，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〕



(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	ア イ ● 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。