平成20年度　春期
基本情報技術者
午後　問題

試験時間	13:00 ～ 15:30（2時間30分）

注意事項

1. 試験開始及び終了は，監督員の時計が基準です。監督員の指示に従ってください。
2. 試験開始の合図があるまで，問題冊子を開いて中を見てはいけません。
3. この注意事項は，問題冊子の**裏表紙**に続きます。必ず読んでください。
4. **答案用紙への受験番号などの記入は，試験開始の合図があってから始めてください。**
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. 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) B又はHBの黒鉛筆又はシャープペンシルを使用してください。訂正の場合は，あとが残らないように消しゴムできれいに消し，消しくずを残さないでください。
   (2) 答案用紙は光学式読取り装置で処理しますので，答案用紙の**マークの記入方法**のとおりマークしてください。
   (3) **受験番号欄**に，**受験番号**を記入及びマークしてください。正しくマークされていない場合，答案用紙の**マークの記入方法**のとおりマークされていない場合は，採点されません。
   (4) **生年月日欄**に，受験票に印字されているとおりの**生年月日**を記入及びマークしてください。正しくマークされていない場合は，採点されないことがあります。
   (5) **選択した問題**については，次の例に従って，**選択欄**の**問題番号**の(選)をマークしてください。マークがない場合は，採点の対象になりません。

〔問6と問10を選択した場合の例〕

選択欄					
問1	●	問6	●	問10	●
問2	●	問7	(選)	問11	(選)
問3	●	問8	(選)	問12	(選)
問4	●	問9	(選)	問13	(選)
問5	●				

   (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)

裏表紙の注意事項も，必ず読んでください。

共通に使用される擬似言語の記述形式

擬似言語を使用した問題では，各問題文中に注記がない限り，次の記述形式が適用されているものとする。

〔宣言，注釈及び処理〕

記述形式		説明
○		手続，変数などの名前，型などを宣言する。
/* 文 */		**文**に注釈を記述する。
処理	・**変数** ← **式**	**変数**に**式**の値を代入する。
	・**手続**(**引数**, …)	**手続**を呼び出し，**引数**を受け渡す。
	▲ **条件式** ▼ **処理**	単岐選択処理を示す。 **条件式**が真のときは**処理**を実行する。
	▲ **条件式** **処理 1** ▼ **処理 2**	双岐選択処理を示す。 **条件式**が真のときは**処理 1** を実行し，偽のときは**処理 2** を実行する。
	■ **条件式** **処理** ■	前判定繰返し処理を示す。 **条件式**が真の間，**処理**を繰り返し実行する。
	■ **処理** ■ **条件式**	後判定繰返し処理を示す。 **処理**を実行し，**条件式**が真の間，**処理**を繰り返し実行する。
	■ **変数：初期値，条件式，増分** **処理** ■	繰返し処理を示す。 開始時点で**変数**に**初期値**（定数又は式で与えられる）が格納され，**条件式**が真の間，**処理**を繰り返す。また，繰り返すごとに，**変数**に**増分**（定数又は式で与えられる）を加える。

〔演算子と優先順位〕

演算の種類	演算子	優先順位
単項演算	+，−，not	高 ↑
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　ハードディスク装置に関する次の記述を読んで，設問1，2に答えよ。

X社では，ハードディスク装置である製品Aと製品Bを生産している。製品Aと製品Bの仕様を表に示す。

表　製品Aと製品Bの仕様

仕様	製品A	製品B
記憶容量	60Gバイト	80Gバイト
回転数	4,200回／分	5,400回／分
平均位置決め時間	14.0ミリ秒	12.0ミリ秒
データ転送速度	20Mバイト／秒	30Mバイト／秒

注　1Gバイト＝1,000Mバイト，1Mバイト＝1,000kバイト

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。解答は小数第2位以下を切り捨てるものとする。

製品Aでは，平均回転待ち時間は［ a ］ミリ秒であり，100kバイトのデータを転送する時間は［ b ］ミリ秒である。

一方，製品Bでは，平均回転待ち時間が5.5ミリ秒であり，100kバイトのデータを転送する時間が3.3ミリ秒であることから，100kバイトの平均アクセス時間（平均位置決め時間＋平均回転待ち時間＋データ転送時間）は，製品Aに比べて［ c ］ミリ秒短い。

解答群

ア　3.3　　イ　5.0　　ウ　5.3　　エ　5.5

オ　7.1　　カ　11.1　　キ　14.2

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。解答は小数第2位以下を切り捨てるものとする。

読取り時のアクセス時間を短縮するために，ディスクキャッシュを搭載した製品Cを開発した。ディスクキャッシュへの書込みと読取りは，2kバイトのブロック単位で行う。あるブロックに対するデータの読取り要求があった場合，ディスクキャッシュを検索し，ディスクキャッシュ上にそのブロックのデータがあればディスクキャッシュ上のデータを返す。ディスクキャッシュ上にそのブロックのデータがなければ，磁気ディスクからデータを読み込み，データを返すと同時にディスクキャッシュ上の最も古いブロックのデータと交換する。

ディスクキャッシュを検索する平均時間は1.0ミリ秒，ディスクキャッシュの平均ヒット率は0.4，ディスクキャッシュからの1ブロック当たりのデータの平均読取り時間は0.4ミリ秒とする。ディスクキャッシュがヒットしなかったとき，磁気ディスクからの1ブロック当たりのデータの平均読取り時間は，ディスクキャッシュ上の最も古いブロックのデータと交換する時間も含めて17.0ミリ秒とする。このとき，製品Cの1ブロック当たりの読取り時の平均アクセス時間は［　d　］ミリ秒となる。

解答群

ア　7.0	イ　8.0	ウ　10.3
エ　11.3	オ　13.2	カ　14.2

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1，2に答えよ。

〔プログラムの説明〕

長さが Textlen の文字列 SourceText の中で，長さが Patlen の文字列 Pattern と一致する部分の文字列（以下，部分文字列という）の出現回数を数える関数 MatchCounter である。ここで，0＜Patlen≦Textlen である。

(1) MatchCounter の処理手順は，次のとおりである。

① 一致する部分文字列の出現回数を数える変数 Counter の値を0に初期化する。

② SourceText の比較開始位置を先頭から順に1文字ずつ後ろにずらしながら，その比較開始位置から始まる長さ Patlen の文字列と Pattern が一致するかどうかを調べ，一致したら出現回数 Counter の値に1を加算する。

③ Counter の値を返す。

(2) MatchCounter の引数と返却値の仕様を表に示す。文字列は文字型配列の各要素に1文字ずつ格納されている。また，各配列の添字は0から始まる。

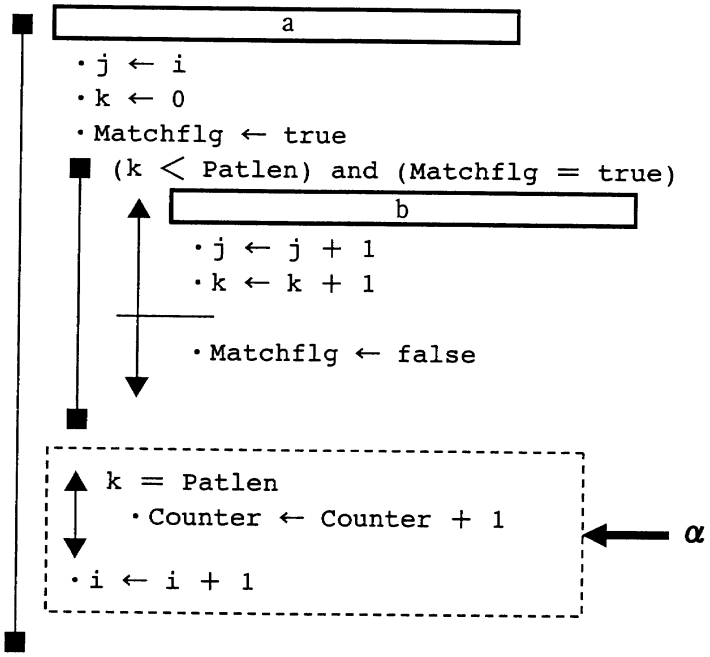
表　MatchCounter の引数と返却値の仕様

引数／返却値	データ型	入力／出力	意味
SourceText[]	文字型	入力	検索される文字列が格納されている1次元配列
Textlen	整数型	入力	検索される文字列の長さ
Pattern[]	文字型	入力	検索する文字列が格納されている1次元配列
Patlen	整数型	入力	検索する文字列の長さ
返却値	整数型	出力	SourceText の中で Pattern と一致した部分文字列の出現回数

〔プログラム〕

```
○整数型: MatchCounter(文字型: SourceText[], 整数型: Textlen,
                       文字型: Pattern[], 整数型: Patlen)
○整数型: Counter, i, j, k
○論理型: Matchflg

・Counter ← 0
・i ← 0
```




```
・return Counter
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　i + Patlen ≦ Textlen　　イ　i + Patlen < Textlen

ウ　i + Textlen ≦ Patlen　　エ　i + Textlen < Patlen

bに関する解答群

ア　SourceText[i] = Pattern[k]

イ　SourceText[j] = Pattern[k]

ウ　SourceText[k] = Pattern[i]

エ　SourceText[k] = Pattern[j]

設問2　Pattern と一致した部分文字列は以降の検索対象から外すように，このプログラムを変更する。

例えば，SourceText と Pattern が図に示す文字列であるとき，プログラムでは，比較開始位置を①，②，③，…と移動して，一致する部分文字列の出現回数を求めるので，下線の部分文字列と二重下線の部分文字列がともに数えられる。これに対して，一致した部分文字列を検索対象から外す場合は，比較開始位置を①，②，⑤，…と移動するので，二重下線の部分文字列は数えられない。

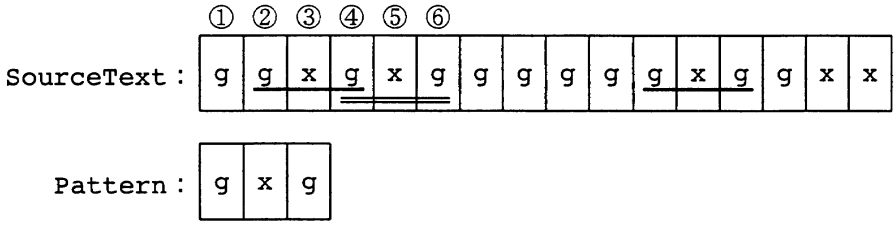


図　比較開始位置を変更する場合の例

このとき，プログラム中の **α** 部分の変更内容として正しい答えを，解答群の中から選べ。

解答群

ア
```
k = Patlen
  ・Counter ← Counter + 1
・i ← i + Patlen
```

イ
```
k = Patlen
  ・Counter ← Counter + 1
  ・i ← i + Patlen
```

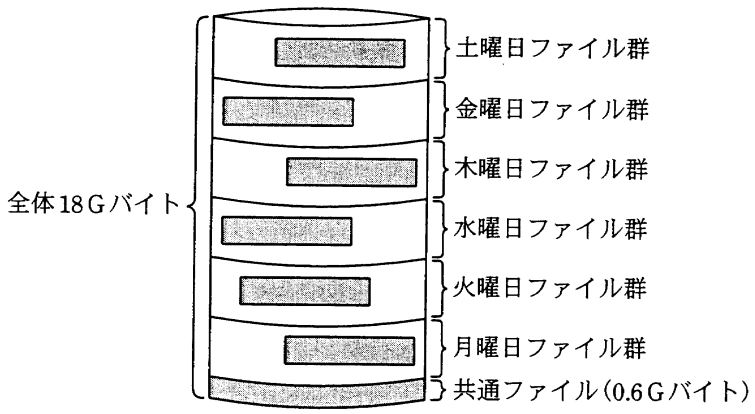
ウ
```
k = Patlen
  ・Counter ← Counter + 1
  ・i ← i + Patlen
・i ← i + 1
```

エ
```
k = Patlen
  ・Counter ← Counter + 1
  ・i ← i + 1
・i ← i + Patlen
```

問3　データファイルのバックアップに関する次の記述を読んで，設問1，2に答えよ。

Y社では，業務系システムが使用する業務データを磁気テープにバックアップすることを検討している。

この業務系システムのサービス運用時間は，営業日（月曜日～土曜日）の9:00～19:00である。全体で18Gバイトからなる業務データは，共通ファイルと，月曜日～土曜日の各曜日に利用する六つの曜日ファイル群からなる。各営業日には，共通ファイル全体（0.6Gバイト）及びその曜日の曜日ファイル群の一部（場所は不定で0.6Gバイト）の計1.2Gバイトを更新する。データの更新状況を図に示す。ここで，1Gバイト＝1,000Mバイトとする。



注　網掛けの部分は，更新領域（0.6Gバイト）を表す。

図　データの更新状況

このシステムにおいて，バックアップ方式を次のとおり検討している。

(1) 週1回，日曜日にすべての業務データのバックアップ（フルバックアップ）を実施する。

(2) 月曜日～土曜日に実施する部分バックアップは，次のいずれかの方式とする。

① 方式A

日曜日のフルバックアップの時点以後に更新された業務データをすべてバックアップする。

② 方式B

前日のバックアップの時点以後に更新された業務データだけをバックアップする。

(3) その他の条件は，次のとおりである。

① バックアップは，その日の業務終了後に実施し，その日のうちに終了する。

② 磁気ディスクから磁気テープへのバックアップの速度は2Mバイト／秒，磁気テープから磁気ディスクへのリストアの速度も同じく2Mバイト／秒である。

③ フルバックアップと各営業日の部分バックアップはそれぞれ別の磁気テープに保存することとし，それぞれのバックアップデータは1本に収まるものとする。

設問1 バックアップ時間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。

日曜日のフルバックアップに要する時間は150分である。

月曜日～土曜日の部分バックアップは，方式Aの場合で最も時間がかかる曜日には［ a ］分，方式Bの場合で毎日［ b ］分かかる。

なお，磁気テープはバックアップ開始までにセットされているものとする。

解答群

ア　5	イ　10	ウ　30
エ　35	オ　145	カ　150

設問2　障害発生時のデータ復旧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。

障害発生時のデータ復旧に使用する磁気テープの最大本数は，フルバックアップされた磁気テープを含めると，方式Aでは［　c　］本，方式Bでは［　d　］本である。

方式Aを採用した場合，復旧には最長で［　e　］分かかる。

なお，磁気テープの交換時間は3分とする。また，復旧の時間には，最初の磁気テープをセットする時間と最後の磁気テープを取り出す時間は含まない。

c，dに関する解答群

ア　1　　イ　2　　ウ　6　　エ　7　　オ　8

eに関する解答群

ア　38　　イ　183　　ウ　185　　エ　188　　オ　193

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) リストを作成するのに用いる副プログラム MakeList と，作成したリストを逐次探索する関数 OrganizingSearch である。OrganizingSearch は，見つかった要素がリストの先頭にくるように要素を並べ替える。

(2) リストは構造体の配列で表現する。構造体の定義は，次のとおりである。

```
構造体型: List[100] {
    整数型: Next = −1,   /* 次の要素の添字，初期値は−1 */
  文字列型: Value        /* 要素の値，可変長文字列 */
}
```

ここで，List は大域変数であり，配列の添字は0から始まる。添字が n の要素 List[n] の Next と Value は，それぞれ List[n].Next，List[n].Value で表す。

(3) 副プログラム MakeList は，1回の呼出しで，引数の文字列を値とする要素を作成し，リストの最後に追加する。MakeList は探索対象のリストが完成するまで続けて呼び出される。整数型の大域変数 Listsize には，作成したリストの要素数を格納する。リストの要素は，List[0] から，List[Listsize − 1] に格納する。

リストに登録する要素の個数は，配列の大きさ以下とする。また，リストの先頭の要素が格納されている配列 List の添字は，整数型の大域変数 First に格納する。First の初期値は−1とする。リストが空の状態から，MakeList の引数として 'cpu'，'ram'，'dos'，'vpn'，'dvd' を順に与えて作成したリストの例を図1に示す。

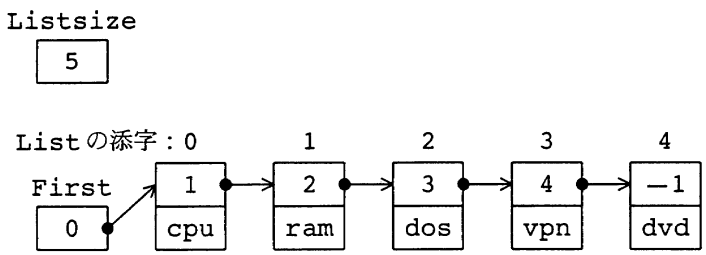


図1　リストの例

(4) 副プログラム MakeList の引数の仕様を表1に示す。

表1　MakeList の引数の仕様

引数	データ型	入力／出力	意味
Mvalue	文字列型	入力	リストに追加する文字列

(5) 関数 OrganizingSearch は，リストの先頭から引数で与えられた文字列を探索し，文字列が見つかれば，見つかった要素がリストの先頭になるようにリストの要素を並べ替え，見つかった要素が格納されている配列 List の添字を返却する。見つからなければ－1を返却する。図1のリストの例で，文字列 'dos' を見つけた直後のリストの状態を図2に示す。この例では，OrganizingSearch の返却値は2となる。

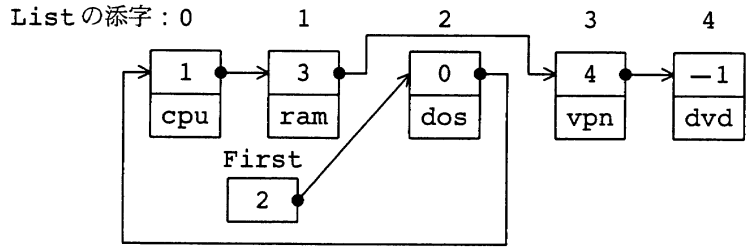


図2　'dos' を見つけた直後のリストの状態

(6) 関数 OrganizingSearch の引数と返却値の仕様を表2に示す。

表2　OrganizingSearch の引数と返却値の仕様

引数／返却値	データ型	入力／出力	意味
Svalue	文字列型	入力	探索する文字列
返却値	整数型	出力	Svalue と一致した文字列を Value としてもつ要素の添字 一致する要素がなければ－1

〔プログラム〕

```
○大域: 整数型: Listsize = 0      /* リストの要素数 */
○大域: 整数型: First = −1        /* リストの先頭要素の添字 */
○大域: 構造体型: List[100] {
           整数型: Next = −1,    /* 次の要素の添字，初期値は−1 */
         文字列型: Value          /* 要素の値，可変長文字列 */
        }

/* 副プログラム MakeList */
○MakeList(文字列型: Mvalue)

▲ Listsize ≠ 0
│ ・List[Listsize − 1].Next ← [ a ]
├──
│ ・First ← 0
▼
・List[Listsize].Value ← Mvalue
・Listsize ← Listsize + 1
/* 副プログラム MakeList の終わり */


/* 関数 OrganizingSearch */
○整数型: OrganizingSearch(文字列型: Svalue)
○整数型: Current
○整数型: Temp

・Current ← First

■ Current ≠ −1
│ ▲ List[Current].Value = Svalue
│ │ ▲ Current ≠ First
│ │ │ ・[ b ] ← List[Current].Next
│ │ │ ・List[Current].Next ← [ c ]
│ │ │ ・First ← [ d ]
│ │ ▼
│ │ ・break                  /* 繰返し処理から抜け出す。 */
│ ├──
│ │ ・Temp ← Current  ◀━━━━━━━ α
│ │ ・Current ← List[Current].Next
│ ▼
■
・return Current
/* 関数 OrganizingSearch の終わり */
```

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　First　　イ　First + 1　　ウ　First − 1

エ　Listsize　　オ　Listsize + 1　　カ　Listsize − 1

b～dに関する解答群

ア　Current　　イ　First

ウ　List[Current].Next　　エ　List[First].Next

オ　List[Temp].Next　　カ　Temp

設問2　図3に示すリストに対して，文字列 'cgi'，'cpu'，'dos' の順に探索した場合，プログラム中の **α** の部分が実行された直後の List[Temp].Value の値を順に並べたものとして正しい答えを，解答群の中から選べ。

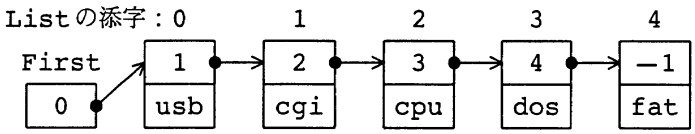


図3　リスト

解答群

ア　cgi, cpu, usb

イ　usb, cgi, cpu

ウ　usb, cgi, usb, cgi, cpu, usb, cgi, cpu, dos

エ　usb, cgi, usb, cpu, cgi

オ　usb, cgi, usb, cpu, cgi, usb

カ　usb, usb, cgi, usb, cgi, cpu

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。

小売企業のA社には，本社と20か所の店舗がある。店舗では商品を販売しており，販売結果は販売データとして本社にある販売システムで管理されている。今回，A社では，販売データを基に販売構成一覧を作成することにした。

〔販売構成一覧作成プログラムの概要〕

(1) 販売データは，販売番号と商品コードをキーとして販売ファイルに記録される。販売番号は，1回の販売ごとに割り当てる全社で一意となる番号である。販売ファイルのレコード様式を図1に示す。

販売番号	商品コード	店舗コード	販売日時	販売数	販売金額

注　下線はキー項目を表す。

図1　販売ファイルのレコード様式

(2) 商品名，商品カテゴリ名及びメーカコードは，商品コードをキーとして商品ファイルに記録されている。商品ファイルのレコード様式を図2に示す。

商品コード	商品名	商品カテゴリ名	メーカコード

注　下線はキー項目を表す。

図2　商品ファイルのレコード様式

(3) 販売構成一覧には，商品カテゴリを集計種別として商品カテゴリ名（項目）ごとに集計したものと，メーカを集計種別としてメーカコード（項目）ごとに集計したものとの2種類がある。集計種別を商品カテゴリ，すなわち集計する項目（以下，集計項目という）を商品カテゴリ名として集計したときの販売構成一覧の例を，図3に示す。

2008-04-15

販売構成一覧

集計期間　2008-01-21～2008-01-31
集計種別　商品カテゴリ

順位	商品カテゴリ名	販売金額	販売数	構成比率
1	子供衣料品	159,457千円	33,333個	33.3％
2	生鮮食料品	122,343千円	443,210個	25.5％
3	乳児衣料品	96,879千円	22,222個	20.2％
⋮	⋮	⋮	⋮	⋮

図3　販売構成一覧の例

(4) 利用者は，販売データの集計期間を指定する。指定した期間の販売データは必ず1件以上あるものとする。また，利用者は，集計種別として商品カテゴリ又はメーカのどちらかを指定する。

(5) 販売構成一覧の作成手順は，次のとおりである。

① 集計期間の販売データを販売ファイルから抽出し，商品コードの昇順に並べ替える。

② ①の結果の販売データ及び商品ファイルのレコードを，商品コードをキーとして突き合わせる。キーが一致した場合，商品ファイルのレコードと販売データのレコードから必要な項目の値を取り出して，それらを中間ファイルAに出力する。

③ 中間ファイルAのレコードから，販売数と販売金額の値を取り出す。さらに，集計項目の値を取り出し，それら三つの値からなるレコードを中間ファイルBに出力する。

④ 中間ファイルBのレコードを並べ替えて，集計項目ごとに販売数と販売金額を合計し，中間ファイルCに出力する。中間ファイルBの全レコードの販売金額を合計し，中間ファイルDに出力する。

⑤ 集計項目ごとに合計した販売金額の高い順に順位を付ける。

⑥ 集計項目ごとに合計した販売金額の，すべての販売金額の合計に対する百分率を算出する。これを，販売構成一覧で表示する集計項目ごとの構成比率とする。

⑦ 販売構成一覧の形式に合わせて作表する。

(6) 中間ファイルAと中間ファイルBのレコード様式を図4に示す。

中間ファイルA

商品カテゴリ名	メーカコード	販売数	販売金額

中間ファイルB

集計項目	販売数	販売金額

図4 中間ファイルAと中間ファイルBのレコード様式

(7) 販売構成一覧の作成の流れを図5に示す。

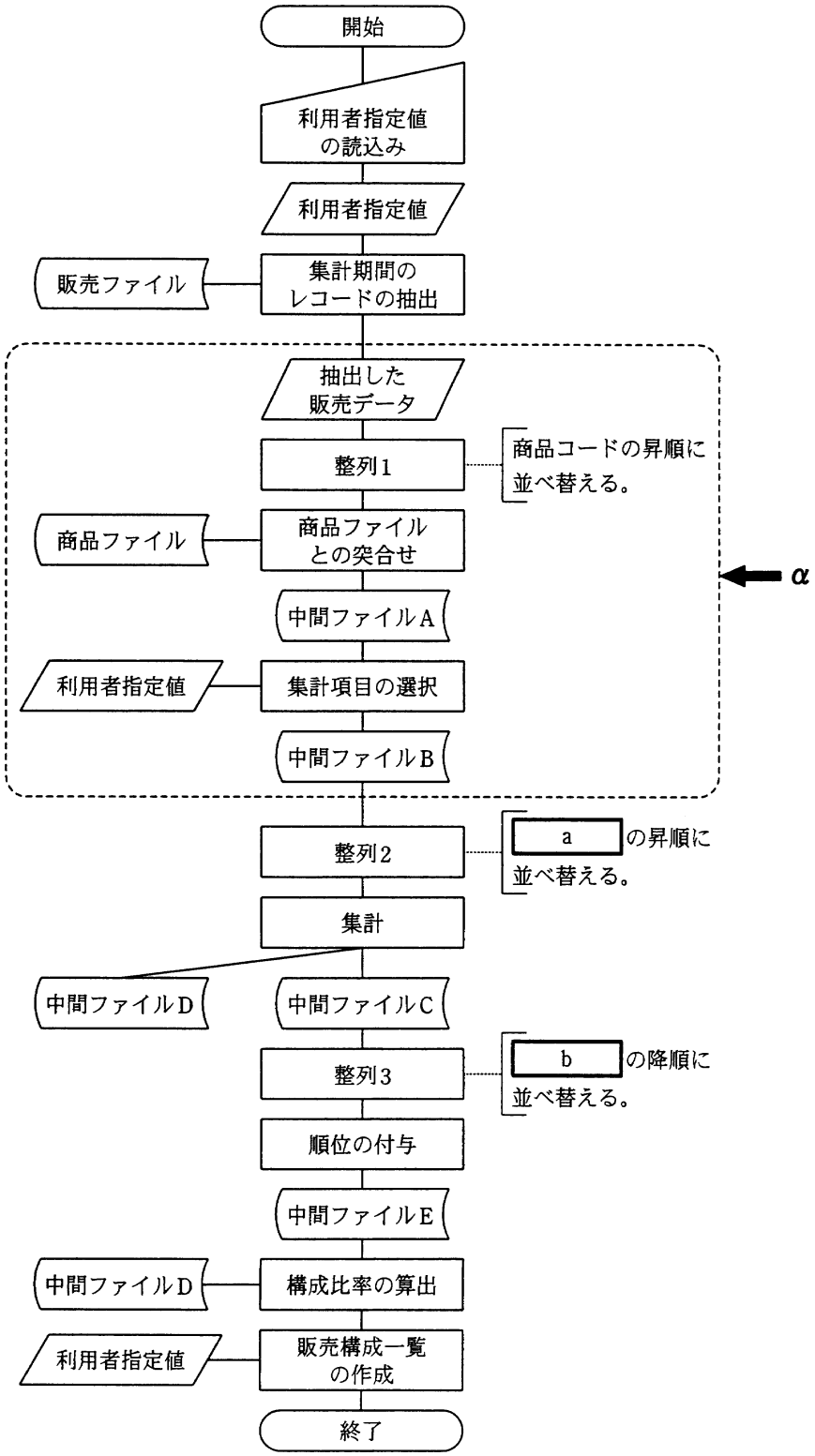


図５　販売構成一覧の作成の流れ

設問1　図5の“利用者指定値の読込み”で入力する項目として正しい答えを，解答群の中から選べ。

解答群

ア　集計開始日付，集計終了日付

イ　集計開始日付，集計終了日付，構成比率

ウ　集計開始日付，集計終了日付，集計種別

エ　集計開始日付，集計終了日付，順位

設問2　図5の“整列2”と“整列3”で行う並替えのキー項目として，図5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　合計した販売金額　　イ　合計した販売数　　ウ　集計項目

エ　商品カテゴリ名　　オ　メーカコード

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。

利用者が指定できる集計種別として店舗を追加し，店舗コード別販売構成一覧を作成する場合を考える。図5中の**α**部分を図6に置き換えることで，既存の処理を変更せずに店舗コード別販売構成一覧が作成できる。図5の“抽出した販売データ”の項目には店舗コードはあるが，中間ファイルAには店舗コードがない。中間ファイルBの集計項目には，店舗コードの値が必要である。そこで，集計種別が店舗の場合に実行する処理として［　c　］を追加する。その実行条件は［　d　］である。

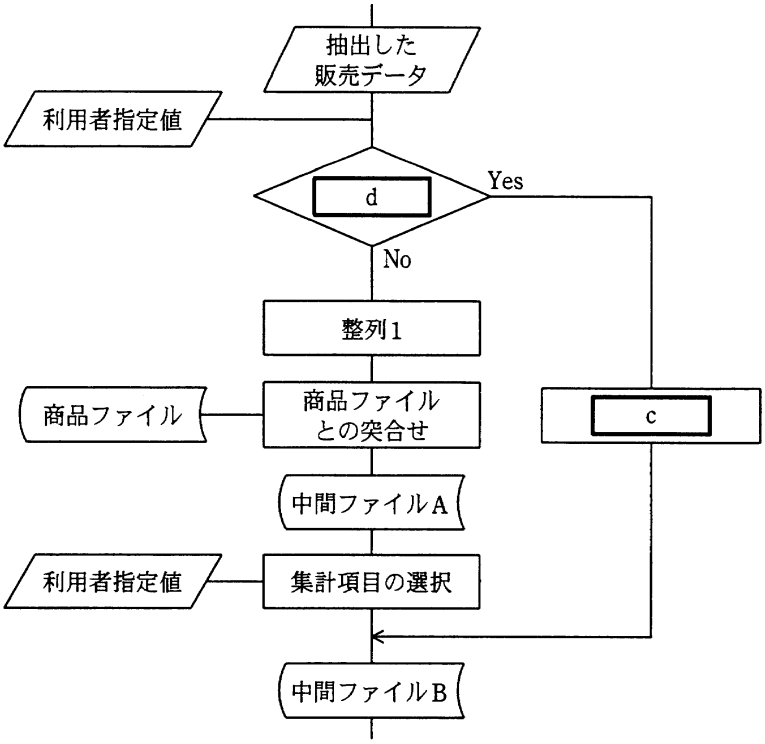


図６　図５中の α 部分と置き換える部分

cに関する解答群

ア　集計期間のレコードの再抽出

イ　順位の付与

ウ　商品ファイルとの突合せ

エ　中間ファイルBの様式への変換

オ　店舗コードの昇順に整列

dに関する解答群

ア　（集計種別＝“商品カテゴリ”）AND（集計種別＝“メーカ”）

イ　（集計種別≠“商品カテゴリ”）OR（集計種別≠“メーカ”）

ウ　集計種別＝“店舗”

エ　集計種別≠“店舗”

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。解答は，重複して選んでもよい。

販売構成一覧の表示について，利用者から次の要求①～③が発生した。

① 順位が10位までのデータだけが表示されるようにする。

② 構成比率で5％以上のデータだけが表示されるようにする。

③ 順位が同じデータの場合，販売数で降順に表示されるようにする。

要求①～③を最も効率よく満たす方法を考える。要求①に対しては，図5中の［ e ］の処理に，順位の値が10以下のレコードだけを出力する機能を追加すればよい。また，要求②に対しては，図5中の［ f ］の処理に，構成比率の値が5以上のレコードを出力する機能を追加すればよい。一方，要求③に対しては，第1キーを順位の昇順，第2キーを販売数の降順として，読み込んだレコードを並べ替えて出力する“整列4”を，新規の処理として図5中の［ g ］の直前に追加すればよい。

解答群

ア　構成比率の算出　　イ　集計　　ウ　順位の付与

エ　整列2　　オ　整列3

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 markup_reference は，文書から参考資料名を抜き出して，参照番号に置き換えるとともに，文書の末尾に参考資料名の一覧を追加して，出力するプログラムである。

(1) 元の文書中では，参考資料名は文字“\”で囲まれている。文字“\”がこれ以外の用途で使用されることはない。

(2) 参照番号は，元の文書中の異なる参考資料名に対して，出現順に1から順に割り当てる番号である。

(3) プログラムの実行例は，図のとおりである。図中の，処理後の文書の例に示すように，参考資料名の一覧は，“References”以降に，参照番号と参考資料名を対にして出力する。

(元の文書の例)

```
The program language includes C, COBOL, Java, etc\Computer Journal\.
 The feature of the Java language has been to have taken the idea of
 object-oriented\Java Report\. In other languages, the idea of objec
t-oriented is being taken\Computer Journal\.
```

(処理後の文書の例)

```
The program language includes C, COBOL, Java, etc[1]. The feature of
 the Java language has been to have taken the idea of object-oriente
d[2]. In other languages, the idea of object-oriented is being taken
[1].

References
[1] Computer Journal
[2] Java Report
```

図　プログラムの実行例

(4) 参考資料名は 255 文字以下であり，途中で改行されることはない。個数は 50 以下である。

(5) 元の文書に含まれる文字は，次のとおりである。

① 英数字

② 図形文字（ ! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~ ）

③ 空白文字

④ 改行文字

(6) 関数 markup_reference の引数は，次のとおりである。ただし，ファイルの名称に誤りはないものとする。

in_filename　元の文書が格納されているファイルの名称

out_filename　処理後の文書を格納するファイルの名称

〔プログラム〕

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

#define MRNUM 50      /* 参考資料名の最大個数 */
#define MRLNG 255     /* 参考資料名の最大文字数 */
#define MARK  '\\'    /* 参考資料名の囲み文字 */

void markup_reference(const char *, const char *);

void markup_reference(const char *in_filename,
                      const char *out_filename){
   FILE *ifp, *ofp;
   char ch, ref_name[MRLNG + 1],
        ref_name_tbl[MRNUM][MRLNG + 1];
   int i, ref_num = 0;

   ifp = fopen(in_filename, "r");
   ofp = fopen(out_filename, "w");
   while((ch = fgetc(ifp)) != EOF){
      if( [   a   ] )
         fputc(ch, ofp);
      else{
         for(i = 0; (ref_name[i] = fgetc(ifp)) != MARK; i++);
         [   b   ] = '\0';
         for(i = 0; i < ref_num; i++)
            if(strcmp(ref_name, ref_name_tbl[i]) == 0)
               break;
```

```
            if(  c  ){
               strcpy(ref_name_tbl[ref_num], ref_name);
               ref_num++;
            }
            fprintf(ofp, "[%d]",  d  );
         }
      }
      fprintf(ofp, "\n\nReferences\n");
      for(i = 0; i < ref_num; i++)
         fprintf(ofp, "[%d] %s\n", i + 1, ref_name_tbl[i]);
      fclose(ifp);
      fclose(ofp);
   }
```

C

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　ch == '\0'　　イ　ch != '\0'

ウ　ch == MARK　　エ　ch != MARK

オ　ref_num == 0　　カ　ref_num != 0

bに関する解答群

ア　ch　　イ　ref_name[i]

ウ　ref_name[i + 1]　　エ　ref_name[i - 1]

オ　*ref_name

cに関する解答群

ア　i == 0　　イ　i != 0

ウ　i < ref_num　　エ　i >= ref_num

オ　ref_num == 0

dに関する解答群

ア　i + 1　　イ　ref_num + 1

ウ　&i + 1　　エ　&ref_num + 1

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〕

ある会社の工場では，1年に1回，従業員とその家族向けに工場を開放して様々なイベントを開催している。このプログラムは，イベントに参加した従業員が提出した“楽しかったイベント”のアンケート結果を集計する。

(1) アンケート結果ファイルは，図1に示すレコード様式の順ファイルである。

従業員番号 6けた	イベント				
	工場見学 (FACTORY) 1けた	健康チェック (HEALTH) 1けた	ミニSL (MINI-SL) 1けた	動物コーナー (ANIMAL) 1けた	PK合戦 (PK) 1けた

図1　アンケート結果ファイルのレコード様式

① アンケート結果ファイルには，楽しかったイベントに1が，それ以外には0が格納されている。

② 楽しかったイベントは二つまで選択できる。三つ以上のイベントが選択されていた場合は，無効として集計対象から除外する。

③ アンケート結果ファイルのレコード数は9,999以下とする。

(2) 集計結果は，図2に示すとおり画面上に表示する。

```
FACTORY : 0265
HEALTH  : 0141
MINI-SL : 0238
ANIMAL  : 0059
PK      : 0097
```

図2　集計結果の例

〔プログラム〕

```
(行番号)
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD  RESULT-FILE.
 4  01  RESULT-REC.
 5      02  EMPLOYEE       PIC X(6).
 6      02  SURVEY         PIC X(5).
 7  WORKING-STORAGE SECTION.
 8  01  SURVEY-DATA.
 9      02  EVENT          PIC 9(1) OCCURS 5.
10  01  TOTAL.
11      02  EVENT-TOTAL    PIC 9(4) OCCURS 5 VALUE ZERO.
12  77  READ-FLAG          PIC X(1) VALUE SPACE.
13      88  DATA-EOF       VALUE "E".
14  77  WK-CNT             PIC 9(1).
15  77  CHOICE-CNT         PIC 9(1).
16  01  HEADER             PIC X(40) VALUE
17                         "FACTORY HEALTH  MINI-SL ANIMAL  PK".
18  01  EVENT-HEADER REDEFINES HEADER.
19      02  EVENT-NAME     PIC X(8) OCCURS 5.
20  PROCEDURE DIVISION.
21  MAIN-PROC.
22      OPEN INPUT RESULT-FILE.
23      PERFORM UNTIL DATA-EOF
24        READ RESULT-FILE AT END     SET DATA-EOF TO TRUE
25                         NOT AT END MOVE SURVEY TO SURVEY-DATA
26                                    PERFORM CNT-PROC
27        END-READ
28      END-PERFORM.
29      CLOSE RESULT-FILE.
30      PERFORM PRT-PROC.
31      STOP RUN.
32  CNT-PROC.
33      MOVE ZERO TO CHOICE-CNT.
34      PERFORM VARYING WK-CNT FROM 1 BY 1 UNTIL WK-CNT > 5
35        [          a          ]
36      END-PERFORM.
37      IF CHOICE-CNT <= 2 THEN
38         PERFORM VARYING WK-CNT FROM 1 BY 1 UNTIL WK-CNT > 5
39           [          b          ]
40         END-PERFORM
41      END-IF.
42  PRT-PROC.
43      PERFORM VARYING WK-CNT FROM 1 BY 1 UNTIL WK-CNT > 5
44        DISPLAY EVENT-NAME(WK-CNT) ": " EVENT-TOTAL(WK-CNT)
45      END-PERFORM.
```

COBOL

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

解答群

ア　COMPUTE CHOICE-CNT = CHOICE-CNT + EVENT(WK-CNT)

イ　COMPUTE CHOICE-CNT = CHOICE-CNT + EVENT-TOTAL(WK-CNT)

ウ　COMPUTE EVENT-TOTAL(WK-CNT)
　　= EVENT-TOTAL(WK-CNT) + CHOICE-CNT

エ　COMPUTE EVENT-TOTAL(WK-CNT)
　　= EVENT-TOTAL(WK-CNT) + EVENT(WK-CNT)

オ　MOVE EVENT(WK-CNT) TO CHOICE-CNT

カ　MOVE EVENT(WK-CNT) TO EVENT-TOTAL(WK-CNT)

設問2　参考データとして，三つ以上のイベントを選択した無効なアンケート結果の総数を，集計結果の次に表示するようにプログラムを変更する。表中の［　　］に入れる正しい答えを，解答群の中から選べ。

COBOL

表　プログラムの変更内容

処置	変更内容
行番号19と20の間に追加	77　INV-CNT　　PIC 9(4) VALUE ZERO.
行番号40と41の間に追加	ELSE ［　c　］
［　d　］に追加	DISPLAY "INVALID DATA : " INV-CNT.

cに関する解答群

ア　ADD 1 TO INV-CNT

イ　MOVE CHOICE-CNT TO INV-CNT

ウ　MOVE EVENT-TOTAL(WK-CNT) TO INV-CNT

エ　MOVE WK-CNT TO INV-CNT

dに関する解答群

ア　行番号32と33の間　　イ　行番号36と37の間

ウ　行番号41と42の間　　エ　行番号45の次

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〕

電気ポットの状態遷移を模したプログラムである。この電気ポットは，電源に接続すると休止状態になり，沸騰ボタンが押されると加熱を開始する。沸点に到達すると加熱を終了し，休止状態に戻る。ただし，電気ポットに水がない場合は，沸騰ボタンが押されても加熱を開始せず，警告を表示する。電気ポットの状態遷移図を図1に示す。

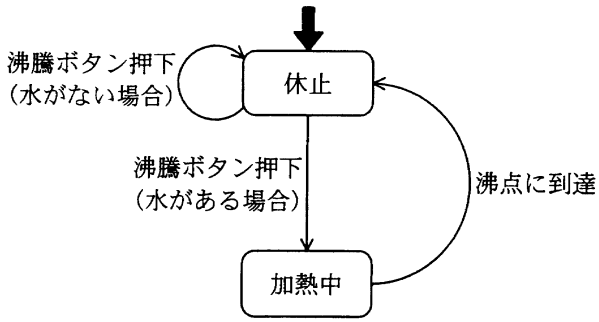


図1　電気ポットの状態遷移図

(1) 抽象クラス State は，電気ポットの状態を定義する。

(2) クラス Idle 及びクラス Heating は，それぞれ，休止状態と加熱中状態を示す State のサブクラスである。

(3) クラス ElectricPot は電気ポットを表す。フィールド currentState は，電気ポットの状態を示す。フィールド content は，水量を示す。メソッド heat は，沸騰ボタンが押されたときに呼ばれ，電気ポットに水があれば加熱を開始する（電気ポットの状態を加熱中に遷移）。水がない場合は例外を投げる。メソッド boiled は，水が沸点に到達したときに呼ばれ，加熱を終了する（電気ポットの状態を休止に遷移）。メソッド isEmpty は，水がない場合 true，それ以外の場合 false を返す。メソッド main はテスト用のメインプログラムである。実行結果を図2に示す。

```
Idle -> Heating
Heating -> Idle
No Water!
```

図2　メソッド main の実行結果

〔プログラム1〕

```
public abstract class State {
   public static final State IDLE = new Idle();
   public static final State HEATING = new Heating();

   public void heat(ElectricPot pot) throws Exception {
      if ( [ a ] )
         [ b ];
      pot.changeState(HEATING);
   }

   public void boiled(ElectricPot pot) {
      pot.changeState(IDLE);
   }

   private static class Idle [ c ] {
      public String toString() { return "Idle"; }
   }

   private static class Heating [ c ] {
      public String toString() { return "Heating"; }
   }
}
```

〔プログラム2〕

```
public class ElectricPot {
   private State currentState = State.IDLE;
   private int content;

   public ElectricPot(int content) { this.content = content; }
   public void setContent(int content) {
      this.content = content;
   }

   public boolean isEmpty() { return (content == 0); }
   public void changeState(State newState) {
      System.out.println(currentState + " -> " + newState);
      currentState = newState;
   }

   public void heat() throws Exception {
      [ d ].heat(this);
   }

   public void boiled() {
      [ d ].boiled(this);
   }
```

```
    public static void main(String[] args) {
       try {
          ElectricPot pot = new ElectricPot(10);
          pot.heat();
          pot.boiled();
          pot.setContent(0);
          pot.heat();
          pot.boiled();
       } catch (Exception e) {
          System.out.println(e.getMessage());
       }
    }
}
```

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　isEmpty()　　イ　!isEmpty()

ウ　pot.isEmpty()　　エ　!pot.isEmpty()

オ　pot == null　　カ　pot != null

bに関する解答群

ア　return "No Water!"

イ　System.out.println("No Water!")

ウ　throw "No Water!"

エ　throw new Exception("No Water!")

cに関する解答群

ア　extends Exception　　イ　extends Object

ウ　extends State　　エ　implements Exception

オ　implements Object　　カ　implements State

dに関する解答群

ア　currentState　　イ　State

ウ　super　　エ　this

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，設問に答えよ。

〔プログラムの説明〕

文字列中の単語を切り出して，単語管理テーブルを作成する副プログラム TOKEN である。

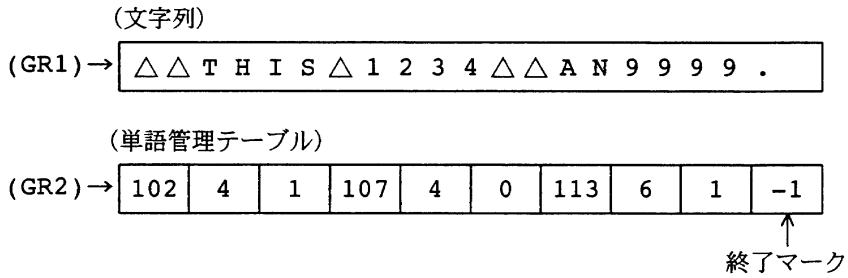
(1) 文字列は英字，数字，空白文字（間隔文字）の0文字以上の並びで，最後にピリオドが置かれる。単語は，1 文字以上の空白文字で区切られた英数字の並びとする。

(2) 単語管理テーブルには，文字列中に現れる単語ごとに，3 語からなる要素を作成し，単語の先頭アドレス，長さ，属性の順に格納する。属性は，単語が数字だけからなるときは 0，英字が含まれるときは 1 とする。単語の切出しが終了したとき，単語管理テーブルの終了を示すマークとして−1を格納する。

(3) 主プログラムは，文字列の先頭アドレスを GR1 に，単語管理テーブルの先頭アドレスを GR2 に設定して，TOKEN を呼ぶ。

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

副プログラム TOKEN の実行例を図に示す。



注　文字列の先頭アドレスを 100 番地とし，数字は 10 進表記である。
△は空白文字を示す。

図　副プログラム TOKEN の実行例

〔プログラム〕

```
TOKEN   START
        RPUSH
        LD      GR3,=-1         ; 単語の処理中を示すフラグの初期化
;                                 0: 数字だけの単語を処理中である。
;                                 1: 英字を含む単語を処理中である。
;                                -1: 単語の処理中ではない。
        LAD     GR1,-1,GR1
LP      LAD     GR1,1,GR1
        LD      GR4,0,GR1       ; 1 文字を取り出す。
        CPL     GR4,='.'        ; 終了判定
        JZE     FIN
        CPL     GR4,=' '
        [        a        ]
        CALL    SETTKN
        JUMP    LP
ALNUM   LD      GR3,GR3         ; 単語の処理中？
        JPL     LP              ;   英字を含む単語を処理中なら LP へ
        JZE     ACHK            ;   数字だけの単語を処理中なら ACHK へ
        [        b        ]
        LD      GR6,GR1         ; 単語の先頭アドレスを退避
ACHK    CPL     GR4,='A'        ; 検査対象文字は数字？
        JMI     LP              ;   数字の場合，次の文字の取出しへ
        LD      GR3,=1          ;   英字の場合
        JUMP    LP
;
FIN     CALL    SETTKN
        LD      GR5,=-1
        ST      GR5,0,GR2       ; 終了マークを格納
        RPOP
        RET
;
SETTKN  LD      GR3,GR3         ; 単語の処理中？
        JMI     FIN2            ;   処理中でなければ何もしない。
        ST      GR6,0,GR2       ; 単語の先頭アドレスを管理テーブルに格納
        LD      GR5,GR1
        SUBL    GR5,GR6         ; 単語の長さを計算
        ST      GR5,1,GR2       ; 単語の長さを格納
        ST      GR3,2,GR2       ; 単語の属性を格納
        LD      GR3,=-1         ; 単語の処理中状態を解除
        [        c        ]
FIN2    RET
        END
```

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	JNZ	ACHK	イ	JNZ	ALNUM	ウ	JNZ	LP
エ	JZE	ACHK	オ	JZE	ALNUM	カ	JZE	LP

bに関する解答群

ア	JMI	ALNUM	イ	JMI	LP
ウ	LD	GR3,=0	エ	LD	GR3,=1
オ	LD	GR3,0	カ	LD	GR3,1

cに関する解答群

ア	LAD	GR1,1,GR1	イ	LAD	GR1,3,GR1
ウ	LAD	GR2,1,GR2	エ	LAD	GR2,3,GR2
オ	LD	GR1,GR6	カ	LD	GR2,GR6

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

なお，2問以上選択した場合には，はじめの1問について採点します。

問10 次のCプログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

(1) 正の有限小数又は循環小数を既約分数（分子と分母がともに整数で共通の約数をもたない分数）に変換するプログラムである。

(2) 有限小数とは，小数点以下のけた数が有限けたである小数をいう。一方，循環小数とは，小数部の特定の位置以降は，同じ数字列が無限に繰り返される小数をいう。

例えば，$\frac{3}{14}$ は 0.2142857142857 … と表記され，142857 が繰り返されるので循環小数である。この小数部における繰返しの部分を循環節と呼ぶ。

(3) プログラムでは，循環節を除く小数部のけた数が n，循環節のけた数が k の場合，その小数を $10^{(n+k)}$ 倍したものから 10^n 倍したものを引くと，整数となることを用いて既約分数への変換を行う。

$\frac{3}{14}$ の小数表記を図に示す。

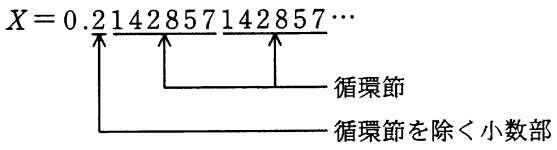


図 $\frac{3}{14}$ の小数表記

この場合，$n=1$，$k=6$ なので，$10^{(1+6)}X-10^1X$ を計算すると次のとおりになる。

$$\begin{array}{rr} & 10000000X = 2142857.142857142857\cdots \\ - & 10X = 2.142857142857\cdots \\ \hline & 9999990X = 2142855 \end{array}$$

したがって，$X=\frac{2142855}{9999990}$ となる。さらに，分子と分母の最大公約数で約分すると，既約分数である $X=\frac{3}{14}$ を得る。

(4) プログラムへの入力では，循環節は“{”と“}”で囲んで表す。例えば，0.2142857142857… は 0.2{142857} と入力する。

(5) プログラムへは，整数（小数点を含まない）を入力することもできる。

(6) プログラムの実行例を表に示す。

表　プログラムの実行例

入力	出力
32	= 32/1
5.237	= 5237/1000
0.875	= 7/8
1.{3}	= 4/3
0.2{142857}	= 3/14

(7) プログラム中で定義されている関数の仕様は，次のとおりである。

```
void tofraction(char *str);
```

引数：整数，有限小数，循環小数のいずれかを表現した文字列 str

機能：str を既約分数に変換して表示する。str に含まれる数字は 9 個以下とする。また，誤った文字列が与えられることはない。

(8) 次の関数があらかじめ定義されているものとする。

① `long gcd(long a, long b);`

引数：自然数 a，b

返却値：a と b の最大公約数

② `long power10(int num);`

引数：0～9 の整数 num

返却値：10 の num 乗

(9) 次のライブラリ関数を用いる。

```
int isdigit(int c);
```

返却値：文字 c が 10 進数字であれば 0 以外，そうでなければ 0

〔プログラム〕

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

void tofraction(char *);
long power10(int);
long gcd(long, long);

void tofraction(char *str){

   long numerator = 0;        /* 分子 */
   long denominator;          /* 分母 */
   int flag = 0;
   int n = 0;                 /* 循環節を除く小数部のけた数 */
   int k = 0;                 /* 循環節のけた数 */
   long measure;              /* 分子と分母の最大公約数 */

   while( [ a ] ){
      if(isdigit(*str)){      /* 10 進数字か？ */
         numerator [ b ];
         numerator += *str - '0';
         if (flag == 1) n++;
         if (flag == 2) k++;
      }else if(*str == '.'){
         flag = 1;
      }else if(*str == '{'){
         flag = 2;
      }
      [ c ];
   }

   if(flag [ d ]) {          /* 有限小数又は整数の場合 */
      denominator = power10(n);
   }else{                    /* 循環小数の場合 */
      denominator  = [ e ];
      numerator   -= [ f ];
   }

   measure = gcd(numerator, denominator);
   numerator /= measure;
   denominator /= measure;
   printf("= %ld/%ld\n", numerator, denominator);
}
```

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　`*str != '\0'`　　イ　`*str != '0'`

ウ　`str != '\0'`　　エ　`str != '0'`

bに関する解答群

ア　`*= 10`　　イ　`+= 10`　　ウ　`/= 10`

エ　`-= 10`　　オ　`%= 10`

cに関する解答群

ア　`(*str)++`　　イ　`(*str)--`

ウ　`str++`　　エ　`str--`

dに関する解答群

ア　`!= 0`　　イ　`!= 1`　　ウ　`!= 2`

エ　`== 0`　　オ　`== 1`　　カ　`== 2`

eに関する解答群

ア　`power10(k)`

イ　`power10(n) * power10(k)`

ウ　`(power10(n) - 1) * power10(k)`

エ　`power10(n) * (power10(k) - 1)`

fに関する解答群

ア　`denominator / power10(n)`

イ　`denominator / power10(k)`

ウ　`numerator / power10(n)`

エ　`numerator / power10(k)`

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〕

比例代表選挙において，得票ファイルを入力し，ドント式で各政党の当選者数を決定して，政党別獲得議席数リストを出力するプログラムである。ドント式とは，各政党の得票数を整数1，2，3，…で割ってゆき，その商の大きい順に議席定数に達するまで議席を各政党に配分する方式である。議席定数15名の場合の例を表1に示す。

表1　議席定数15名の場合の例

	A党	B党	C党	D党
得票数	1,000	550	350	100
÷1	1,000 ①	550 ②	350 ④	100
÷2	500 ③	275 ⑥	175 ⑩	50
÷3	333 ⑤	183 ⑨	116 ⑮	33
÷4	250 ⑦	137 ⑬	87	25
÷5	200 ⑧	110	70	
÷6	166 ⑪	91		
÷7	142 ⑫			
÷8	125 ⑭			
÷9	111			
獲得議席数	8	4	3	0

注　①～⑮は当選の順番を示す。

(1) 得票ファイルVOTE-Fは順ファイルで，レコード様式は図1のとおりである。

政党名（20けた）	得票数（10けた）

} 政党別得票数情報（得票した政党分並ぶ。）

図1　得票ファイルVOTE-Fのレコード様式

① 政党数は最大50とする。

② 得票ファイルのデータに誤りはないものとする。

(2) 議席定数は，このプログラムへのパラメタとして渡され，数字3けたで値は1以上とする。

(3) 政党別獲得議席数リストファイル PRINT-F の印字様式は，図2のとおりである。

政党名	獲得議席数
XXX … XXX	ZZ9
XXX … XXX	ZZ9
⋮	⋮

図2 政党別獲得議席数リストファイル PRINT-F の印字様式

① 政党別獲得議席数リストの見出しは印刷済とする。

② 獲得議席数の多い順に印字する。

〔プログラム〕

(行番号)

```
 1  DATA DIVISION.
 2  FILE SECTION.
 3  FD  VOTE-F.
 4  01  VOTE-R.
 5      05  V-PARTY-NAME       PIC X(20).
 6      05  V-VALUE            PIC 9(10).
 7  FD  PRINT-F.
 8  01  PRINT-R.
 9      05  P-PARTY-NAME       PIC X(20).
10      05  FILLER             PIC X(05).
11      05  P-SEATS            PIC ZZ9.
12  SD  SORT-F.
13  01  SORT-R.
14      05  S-PARTY-NAME       PIC X(20).
15      05  S-VPOINT           PIC 9(10).
16  WORKING-STORAGE SECTION.
17  01  EOF-SW                 PIC 9(01).
18  01  FOUND-SW               PIC 9(01).
19  01  SEAT-CNT               PIC 9(03).
20  01  RESULT-INFO.
21      05  R-CNT              PIC 9(03).
22      05  R-TBL              OCCURS 50.
23          10  R-PARTY-NAME   PIC X(20).
24          10  R-SEATS        PIC 9(10).
25  01  WK-I                   PIC 9(10).
26  LINKAGE SECTION.
27  01  NUM-SEATS              PIC 9(03).
```

```
28  PROCEDURE DIVISION USING NUM-SEATS.
29  MAIN-CTL.
30      MOVE 0 TO R-CNT.
31      SORT SORT-F [          a          ]
32           INPUT  PROCEDURE S-INPUT-PROC
33           OUTPUT PROCEDURE S-OUTPUT-PROC.
34      PERFORM PRINT-PROC.
35      EXIT PROGRAM.
36  S-INPUT-PROC.
37      OPEN INPUT VOTE-F.
38      MOVE 0 TO EOF-SW.
39      PERFORM UNTIL EOF-SW = 1
40        READ VOTE-F AT END MOVE 1 TO EOF-SW
41                    NOT AT END
42                    PERFORM VARYING WK-I FROM 1 BY 1
43                            UNTIL   WK-I > NUM-SEATS
44                      MOVE V-PARTY-NAME TO S-PARTY-NAME
45                      DIVIDE V-VALUE BY WK-I GIVING S-VPOINT
46                      RELEASE SORT-R
47                    END-PERFORM
48        END-READ
49      END-PERFORM.
50      CLOSE VOTE-F.
51  S-OUTPUT-PROC.
52      MOVE 0 TO EOF-SW.
53      PERFORM VARYING SEAT-CNT FROM 1 BY 1
54                [          b          ]
55        RETURN SORT-F AT END MOVE 1 TO EOF-SW
56                      NOT AT END PERFORM REGIST-RTBL
57        END-RETURN
58      END-PERFORM.
59  REGIST-RTBL.
60      MOVE 0 TO FOUND-SW.
61      PERFORM VARYING WK-I FROM 1 BY 1
62              UNTIL WK-I > R-CNT OR FOUND-SW = 1
63        IF S-PARTY-NAME = R-PARTY-NAME(WK-I) THEN
64           [          c          ]
65           MOVE 1 TO FOUND-SW
66        END-IF
67      END-PERFORM.
68      IF FOUND-SW = 0 THEN
69         ADD 1 TO R-CNT
70         MOVE S-PARTY-NAME TO R-PARTY-NAME(R-CNT)
71         [          d          ]
72      END-IF.
```

COBOL

```
 73  PRINT-PROC.
 74      OPEN OUTPUT PRINT-F.
 75      PERFORM VARYING WK-I FROM 1 BY 1 UNTIL WK-I > R-CNT
 76        MOVE SPACE TO PRINT-R
 77        MOVE R-PARTY-NAME(WK-I) TO P-PARTY-NAME
 78        MOVE R-SEATS      (WK-I) TO P-SEATS
 79        WRITE PRINT-R
 80      END-PERFORM.
 81      CLOSE PRINT-F.
```

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　ASCENDING　KEY S-PARTY-NAME

イ　ASCENDING　KEY S-VPOINT

ウ　DESCENDING KEY S-PARTY-NAME

エ　DESCENDING KEY S-VPOINT

bに関する解答群

ア　UNTIL EOF-SW = 1

イ　UNTIL SEAT-CNT >= NUM-SEATS

ウ　UNTIL SEAT-CNT > NUM-SEATS OR EOF-SW = 1

エ　UNTIL SEAT-CNT <= NUM-SEATS

c，dに関する解答群

ア　ADD　1 TO R-SEATS(R-CNT)

イ　ADD　1 TO R-SEATS(WK-I)

ウ　MOVE　1 TO R-SEATS(R-CNT)

エ　MOVE　1 TO R-SEATS(WK-I + 1)

オ　MOVE　1 TO R-SEATS(WK-I - 1)

設問2　この方式では，最終当選順位と次点以降の順位の商が等しい場合が起こりうる。また，最終当選順位の前にも等しい商での当選が並んでいる可能性がある。

このため，最終当選順位と次点の順位の商が同じ場合は，次のメッセージ“*m*-WAY TIE FOR *n*-th PLACE”（第*n*位から*m*名が同点である）を表示する

ようにプログラムを変更する。次の表2中の［　　］に入れる正しい答えを，解答群の中から選べ。

表2　プログラムの変更内容

処置	変更内容
行番号25と26の間に追加	01　W-VPOINT　　PIC 9(10). 01　WARNING-MESSAGE. 　　05　EQUAL-CNT　　PIC 9(03). 　　05　FILLER PIC X(13)　VALUE "-WAY TIE FOR ". 　　05　EQUAL-FROM　　PIC 9(03). 　　05　FILLER PIC X(09)　VALUE "-th PLACE".
［ e ］に追加	PERFORM SET-BREAK
［ f ］に追加	PERFORM CHECK-EQUAL.

行番号81の次に追加	* SET EQUAL-FROM OF BREAK POINT AND COUNT EQUAL-CNT
	SET-BREAK.
	　IF SEAT-CNT = 1 THEN
	　　MOVE 1 TO EQUAL-FROM
	　　MOVE 1 TO EQUAL-CNT
	　ELSE
	　　IF S-VPOINT = W-VPOINT THEN
	　　ADD　1 TO EQUAL-CNT
	　　ELSE
	　　MOVE SEAT-CNT TO EQUAL-FROM
	　　MOVE 1 TO EQUAL-CNT
	　　END-IF
	　END-IF.
	　MOVE S-VPOINT TO W-VPOINT.
	* CHECK EQUAL-CNT AND DISPLAY WARNING-MESSAGE
	CHECK-EQUAL.
	　IF EOF-SW NOT = 1 THEN
	　　RETURN SORT-F AT END MOVE 1 TO EOF-SW
	　　END-RETURN
	　　IF EOF-SW NOT = 1 AND S-VPOINT = W-VPOINT THEN
	　　PERFORM UNTIL EOF-SW = 1 OR
	　　S-VPOINT NOT = W-VPOINT
	　　ADD 1 TO EQUAL-CNT
	　　RETURN SORT-F AT END MOVE 1 TO EOF-SW
	　　END-RETURN
	　　END-PERFORM
	　　ELSE
	　　MOVE 1 TO EQUAL-CNT
	　　END-IF
	　　IF EQUAL-CNT > 1 THEN
	　　DISPLAY WARNING-MESSAGE
	　　END-IF
	　END-IF.

COBOL

解答群

ア　行番号 34 と 35 の間

イ　行番号 50 と 51 の間

ウ　行番号 52 と 53 の間

エ　行番号 56 と 57 の間

オ　行番号 58 と 59 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1，2に答えよ。

〔プログラムの説明〕

キーとそのキーに関連付けられた値をある形式に従って出力するプログラムである。

プログラム1-1は，与えられた二つの文字列をキー及びそのキーに関連付けられた値として“キー＝値”の形式（以下，プロパティ形式という）で出力する。クラスToPropertiesのメソッドwriteの引数keyはキー，引数valueは値である。キー及び値は英数字からなる文字列である。

プログラム1-2は，クラスToPropertiesのテストプログラムである。メソッドmainを実行すると，図1に示す結果が得られる。

```
color=red
font=serif
```

図1　プログラム1-2のメソッドmainの実行結果

このプロパティ形式に加えて，“<”と“>”で囲まれたタグを使用する形式（以下，タグ形式という）で出力できるようにする。タグ形式では，キーと値の組を図2に示す形式で表し，その組をエントリという。

```
<entry key="キー">値</entry>
```

図2　タグ形式におけるエントリ

エントリの並びを，タグ<data>及び</data>で囲んで出力する。

これを実現するために，次のプログラムを新たに作成し，プログラム1-1に変更を加えた。

(1) プログラム2-1は，プロパティ形式とタグ形式の出力を統一して扱うインタフェースKeyValueWriterを定義する。メソッドinitializeは，必要な前処理を行い，メソッドwriteは，引数で与えられたキーと値を出力し，メソッドcompleteは，後処理を行う。出力形式は，これらのメソッドの実装で決まる。

(2) プログラム2-2は，インタフェースKeyValueWriterを実装するクラスToTagsである。

(3) プログラム2-3は，インタフェースKeyValueWriterで定義されている全メソッドの省略時の実装を与えるクラスKeyValueWriterAdapterである。各メソッドの実装は，必ず空である。

(4) プログラム2-4は，プログラム1-1をクラスKeyValueWriterAdapterに適用できるように変更したものである。

(5) プログラム2-5は，変更されたクラスToPropertiesおよびクラスToTagsの出力を検査するテストプログラムである。

全プログラムの変更後，プログラム2-5のメソッドmainを実行すると，図3に示す実行結果が得られる。

```
width=200
height=120
color=blue
---------------
<data>
<entry key="width">200</entry>
<entry key="height">120</entry>
<entry key="color">blue</entry>
</data>
```

図3　プログラム2-5のメソッドmainの実行結果（全プログラムの変更後）

〔プログラム1-1〕

```
public class ToProperties {
   public void write(String key, String value) {
      System.out.println(key + "=" + value);
   }
}
```

〔プログラム1-2〕

```
public class Test1 {
   public static void main(String[] args) {
      ToProperties writer = new ToProperties();
      writer.write("color", "red");
      writer.write("font", "serif");
   }
}
```

〔プログラム 2-1〕

```
public interface KeyValueWriter {
   public void initialize();
   public void write(          a          );
   public void complete();
}
```

〔プログラム 2-2〕

```
public class ToTags implements KeyValueWriter {
   public void initialize() {
      System.out.println("<data>");
   }
   public void write(String key, String value) {
      System.out.println("<entry key=\"" + key + "\">"
                         + value + "</entry>");
   }
   public void complete() {
      System.out.println("</data>");
   }
}
```

〔プログラム 2-3〕

```
public class KeyValueWriterAdapter
                              implements KeyValueWriter {
   public void initialize() { }
   public void write(          a          ) { }
   public void complete() { }
}
```

〔プログラム 2-4〕

```
public          b          {
   public void write(String key, String value) {
      System.out.println(key + "=" + value);
   }
}
```

〔プログラム 2-5〕

```
public class Test2 {
   static String[][] pairs = { {"width", "200"},
                               {"height", "120"},
                               {"color", "blue"} };
```

```
    public static void main(String[] args) {
        write(new ToProperties());
        System.out.println("---------------");
        write(new ToTags());
    }
    private static void write( [ c ] out) {
        out.initialize();
        for (String[] pair : pairs) {
            out.write( [ d ] );
        }
        out.complete();
    }
}
```

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　K key, V value　　イ　Object key, Object value

ウ　String key, String value　　エ　String[] key, String[] value

bに関する解答群

ア　abstract class ToProperties extends KeyValueWriterAdapter

イ　class ToProperties extends KeyValueWriterAdapter

ウ　class ToProperties extends ToTags

エ　class ToProperties implements KeyValueWriter

オ　class ToProperties implements ToTags

cに関する解答群

ア　KeyValueWriter　　イ　KeyValueWriterAdapter

ウ　Test2　　エ　ToProperties

オ　ToTags

dに関する解答群

ア　pair[0], pair[1]　　イ　pair[1], pair[0]

ウ　pair[i], pair[i + 1]　　エ　pairs[][0], pairs[][1]

オ　pairs[][1], pairs[][0]

設問2　プログラムを更に拡張して，与えられた文字列をコメントとして出力できるようにする。すなわち，プロパティ形式では与えられた文字列の先頭に“# ”を付加して出力し，タグ形式では与えられた文字列を“<!-- ”及び“ -->”で囲んで出力する。例えば，与えられた文字列が“This is a comment.”のとき，プロパティ形式では図4のとおりに，タグ形式では図5のとおりに出力する。ただし，コメントとして与えられた文字列は，改行文字及び文字列“--”を含まないものとする。

```
# This is a comment.
```

図4　プロパティ形式でのコメントの出力例

```
<!-- This is a comment. -->
```

図5　タグ形式でのコメントの出力例

この拡張のために，次の表に示す各コードをプログラムに追加した。各コードの追加先のプログラムとして，表中の［　　］に入れる正しい答えを，解答群の中から選べ。

表　コードの追加先と追加内容

追加先	追加内容

e	public void writeComment(String comment) {
	　System.out.println("# " + comment);
	}

f	public void writeComment(String comment) {
	　System.out.println("<!-- " + comment + " -->");
	}

g	public void writeComment(String comment);
h	public void writeComment(String comment) { }

解答群

ア　プログラム2-1　　イ　プログラム2-2

ウ　プログラム2-3　　エ　プログラム2-4

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，設問に答えよ。

〔プログラムの説明〕

(1) プログラム MAIN は，入力ファイル中の1～3けたの数字列からなるレコードを入力し，入力した数字列を10進数値とみなしたときの最大値，最小値及び平均値を10進数の数字列で出力する。入力ファイルの例を図1に，このときの出力の例を図2に示す。

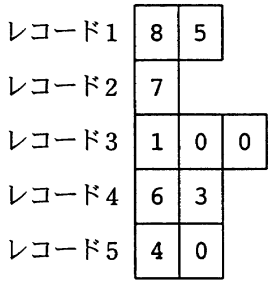


図1　入力ファイルの例

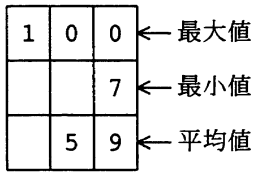


図2　出力の例

① 0≦入力レコードの数値≦100とする。

② 1≦レコード数≦100とする。

③ 平均値は小数点以下を切り捨てる。

(2) プログラムMAINは，三つの副プログラムINPUT，DIVIDE及びPRINTを使用する。

(3) 副プログラムINPUTは，入力ファイルからレコードを入力する。

① 入力したレコードの数字列を2進数に変換し，GR0に設定して，主プログラムに戻る。

② ファイルの終わりを検出した場合は，GR0に−1を設定して，主プログラムに戻る。

③ 副プログラムから戻るとき，汎用レジスタGR1～GR7の内容は元に戻す。

(4) 副プログラムDIVIDEは，除算を行う。

① 被除数はGR2に，除数はGR1に設定されて，主プログラムから渡される。GR1＞0，GR2≧0とする。

② 商をGR3に，剰余をGR2に設定して，主プログラムに戻る。

③ 副プログラムは，GR2，GR3以外の汎用レジスタの内容は変更しない。

アセンブラ

(5) 副プログラムPRINTは，2進数nを3けたの10進数の数字列に変換して出力する。

① nはGR1に設定されて，主プログラムから渡される。

② 出力する数字列は右詰めにし，全体で3文字になるように左端から空白文字を補う。

③ 副プログラムから戻るとき，汎用レジスタGR1～GR7の内容は元に戻す。

〔プログラム1〕

```
MAIN    START
        CALL  INPUT
        ST    GR0,MAX
        ST    GR0,MIN
        LD    GR2,GR0          ; 合計値を初期化
        LD    GR1,=1           ; カウンタを初期化
LOOP1   CALL  INPUT
        LD    GR0,GR0          ; 入力値
        JMI   FIN1             ; ファイルの終わり
        ADDA  GR2,GR0
        ADDA  GR1,=1
        CPA   GR0,MAX
        JPL   HIGH
        CPA   GR0,MIN
        JMI   LOW
        JUMP  LOOP1
HIGH    ST    GR0,MAX
        JUMP  LOOP1
LOW     ST    GR0,MIN
        JUMP  LOOP1
FIN1    CALL  DIVIDE
        LD    GR1,MAX
        CALL  PRINT            ; 最大値を出力
        LD    GR1,MIN
        CALL  PRINT            ; 最小値を出力
              a
        CALL  PRINT            ; 平均値を出力
        RET
MAX     DS    1
MIN     DS    1
        END
```

〔プログラム2〕

```
INPUT   START
        RPUSH
        IN      IBUF,ILEN       ; レコードを入力
        LD      GR0,ILEN
        JMI     FIN2            ; ファイルの終わり
        LD      GR1,=0          ; けた数のカウンタ
        LD      GR4,=0          ; 数値の初期化
LOOP2   LD      GR3,IBUF,GR1
        AND     GR3,=#000F      ; 数字を数値に変換
        LD      GR5,GR4         ;
        [        b        ]     ;
        SLL     GR4,1           ;  } GR4の内容を10倍する。
        ADDA    GR4,GR5         ;
        ADDA    GR4,GR3
        ADDA    GR1,=1
        CPA     GR1,GR0
        [        c        ]
        LD      GR0,GR4
FIN2    RPOP
        RET
ILEN    DS      1
IBUF    DS      256
        END
```

〔プログラム3〕

```
DIVIDE  START
        LD      GR3,=0          ; 商 ← 0
LOOP3   CPA     GR2,GR1
        [        d        ]
        ADDA    GR3,=1
        SUBA    GR2,GR1
        JUMP    LOOP3
FIN3    RET
        END
```

〔プログラム4〕

```
PRINT   START
        RPUSH
        LD      GR4,=3          ; 出力けた数
        LD      GR2,GR1         ; 被除数 ← 数値
        LD      GR1,=10         ; 除数 ← 10
LOOP4   CALL    DIVIDE          ; 数値 / 10
        OR      GR2,=#0030      ; 剰余を数字に変換
        SUBA    GR4,=1
        ST      GR2,OBUF,GR4
        LD      GR2,GR3         ; 被除数 ← 商
        [   e   ]               ; 商 = 0 ?
        LD      GR2,=#0020      ; GR2 ← 空白文字
LOOP5   SUBA    GR4,=1          ;
        [   f   ]               ;
        ST      GR2,OBUF,GR4    ;  } 残りのけた位置に空白文字を入れる。
        JUMP    LOOP5           ;
FIN4    OUT     OBUF,OLEN
        RPOP
        RET
OLEN    DC      3
OBUF    DS      3
        END
```

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	LD	GR1,GR0	イ	LD	GR1,GR2	ウ	LD	GR1,GR3
エ	LD	GR2,GR0	オ	LD	GR2,GR1	カ	LD	GR2,GR3

bに関する解答群

ア	SLL	GR5,1	イ	SLL	GR5,2	ウ	SLL	GR5,3
エ	SRL	GR5,1	オ	SRL	GR5,2	カ	SRL	GR5,3

cに関する解答群

ア	JMI	FIN2	イ	JMI	LOOP2
ウ	JPL	FIN2	エ	JPL	LOOP2
オ	JZE	FIN2	カ	JZE	LOOP2

dに関する解答群

ア　JMI　FIN3　　イ　JNZ　FIN3

ウ　JPL　FIN3　　エ　JZE　FIN3

e，fに関する解答群

ア　JMI　FIN4　　イ　JMI　LOOP4

ウ　JNZ　FIN4　　エ　JNZ　LOOP4

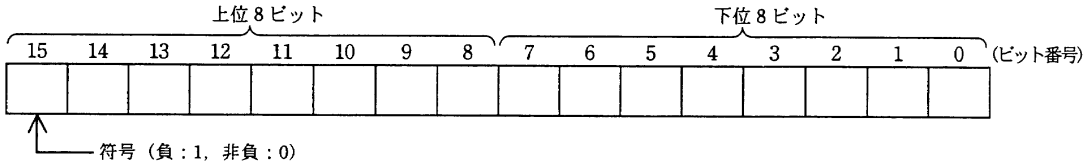
オ　JPL　FIN4　　カ　JZE　LOOP4

■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)	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,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 〕

〔 メ モ 用 紙 〕

〔 メ モ 用 紙 〕

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

退室可能時間	13:40 ～ 15:20

8. **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。
9. 問題冊子の余白などは，適宜利用して構いません。
10. 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。
11. 試験中，机上に置けるもの及び使用できるものは，次のものに限ります。
    なお，会場での貸出しは行っていません。
    受験票，黒鉛筆又はシャープペンシル，鉛筆削り，消しゴム，定規，時計（アラームなど時計以外の機能が付いているものは不可），ハンカチ，ティッシュ
    これら以外は机上に置けません。使用もできません。
12. 試験終了後，この問題冊子は持ち帰ることができます。
13. 答案用紙は，いかなる場合でも，すべて提出してください。回収時に提出しない場合は，採点されません。
14. 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。