文部科学省後援

情報システム試験

平成20年度後期 情報検定

＜実施　平成21年2月8日（日）＞

プログラミングスキル

（説明時間　10：00～10：10）

（試験時間　10：10～11：40）

・試験問題は試験開始の合図があるまで開かないでください。

・解答用紙（マークシート）への必要事項の記入は，試験開始の合図と同時に行いますので，それまで伏せておいてください。

・試験開始の合図の後，次のページを開いてください。**＜受験上の注意＞**が記載されています。必ず目を通してから解答を始めてください。

・試験問題は，すべてマークシート方式です。正解と思われるものを１つ選び，解答欄の()をＨＢの黒鉛筆でぬりつぶしてください。２つ以上ぬりつぶすと，不正解になります。

・辞書，参考書類の使用および筆記用具の貸し借りは一切禁止です。

・電卓の使用が認められます。ただし，下記の機種については使用が認められません。

＜使用を認めない電卓＞

１．電池式（太陽電池を含む）以外の電卓

２．文字表示領域が複数行ある電卓（計算状態表示の一行は含まない）

３．プログラムを組み込む機能がある電卓

４．電卓が主たる機能ではないもの

＊パソコン（電子メール専用機等を含む），携帯電話（ＰＨＳ），ポケットベル，電子手帳，電子メモ，電子辞書，翻訳機能付き電卓，音声応答のある電卓，電卓付腕時計等

５．その他試験監督者が不適切と認めるもの

財団法人　専修学校教育振興会

＜受験上の注意＞

１．この試験問題は44ページあります。ページ数を確認してください。
乱丁等がある場合は，手をあげて試験監督者に合図してください。
※問題を読みやすくするために空白ページを設けている場合があります。

２．解答用紙（マークシート）に，受験者氏名・受験番号を記入し，受験番号下欄の数字をぬりつぶしてください。正しく記入されていない場合は，採点されませんので十分注意してください。

３．試験問題についての質問には，一切答えられません。自分で判断して解答してください。

４．試験中の筆記用具の貸し借りは一切禁止します。筆記用具が破損等により使用不能となった場合は，手をあげて試験監督者に合図してください。

５．試験を開始してから30分以内は途中退出できません。30分経過後退出する場合は，もう一度，受験番号・マーク・氏名が記載されているか確認して退出してください。なお，試験終了５分前の合図以降は退出できません。試験問題は各自お持ち帰りください。

６．合否通知の発送は平成21年3月中旬の予定です。
①団体受験された方は，団体経由で合否の通知をいたします。
②個人受験の方は，受験票に記載されている住所に郵送で合否の通知をいたします。
③合否等の結果についての電話・手紙等でのお問い合わせには，一切応じられませんので，ご了承ください。

＜問題の構成＞

必須問題　全員解答

問題１～問題３	2 ページ～14 ページ

選択問題　Ａ　次の問題から１問選択し解答せよ。

（選択した問題は解答用紙「選択欄」に必ずマークすること）

※選択欄にマークがなく，解答のみマークした場合は採点を行いません。

・Ｃ言語の問題	16 ページ～19 ページ
・Ｊａｖａの問題	20 ページ～24 ページ
・アセンブラの問題	25 ページ～26 ページ

選択問題　Ｂ　次の問題から１問選択し解答せよ。

（選択した問題は解答用紙「選択欄」に必ずマークすること）

※選択欄にマークがなく，解答のみマークした場合は採点を行いません。

・Ｃ言語の問題	27 ページ～32 ページ
・Ｊａｖａの問題	33 ページ～41 ページ
・アセンブラの問題	42 ページ～44 ページ

必須問題

問題１　次の単純挿入法によるデータの整列（ソート）に関する説明を読み，設問に答えよ。

［流れ図の説明］

単純挿入法を使って，n 個の要素を持つ一次元配列 DATA[i](i=0～n-1)を昇順に整列(ソート)する流れ図である。単純挿入法の手順は以下のとおりである。

（単純挿入法の手順）

配列DATAが(i-1)番目まで昇順に整列されているとき，DATA(i)の要素を挿入する適切な位置を見つけて，DATA[i]を挿入する（図1参照)。挿入位置を見つける処理は以下のとおりである。

① DATA[i]をWに退避する。
② 挿入する位置が見つかるまで，DATA[i-1]～DATA[0]までの各要素と退避した値と比較して挿入位置を探索する。
③ 挿入位置にWを挿入する。
④ ①～③の処理をiの値が1～n-1までくり返す

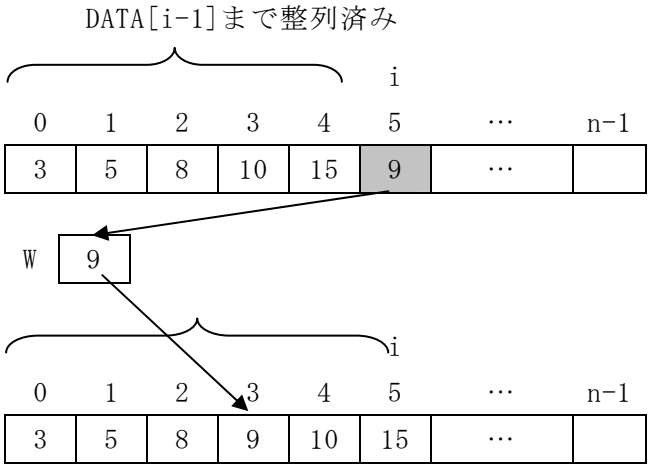


図１　挿入法

＜設問＞　流れ図中(図２)の［　　］に入れるべき適切な字句を解答群から選べ。

(1) ～ (3) の解答群

ア．DATA[i] ← W　　イ．W ← DATA[i]　　ウ．W ← DATA[i＋1]
エ．DATA[j] ← W　　オ．DATA[j＋1] ← W　　カ．i ← i ＋ 1
キ．i ← i － 1　　ク．j ← j ＋ 1　　ケ．j ← j － 1

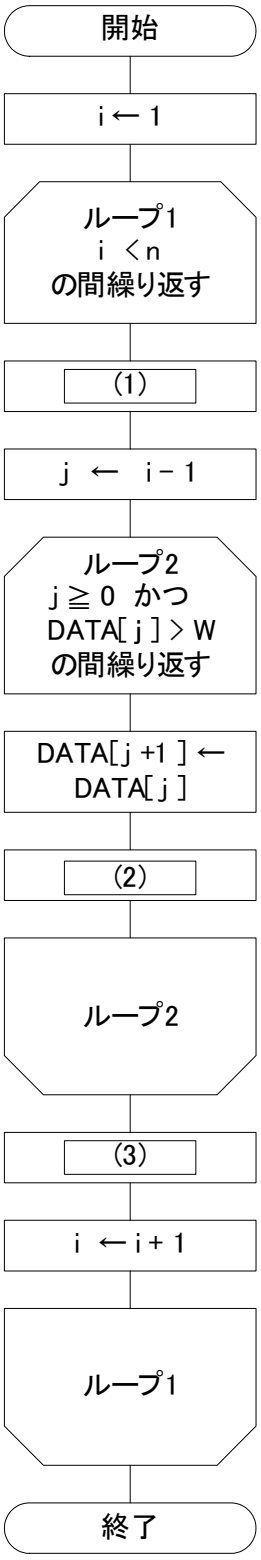

開始
i ← 1
ループ1
i ＜n
の間繰り返す
(1)
j ← i－1
ループ2
j ≧ 0 かつ
DATA[j] ＞ W
の間繰り返す
DATA[j +1] ←
DATA[j]
(2)
ループ2
(3)
i ← i + 1
ループ1
終了

図２　挿入法の流れ図

問題２　次のヒープ構造に関する説明を読み，設問に答えよ。

［完全２分木とヒープ構造の説明］

完全 2 分木は，2 分木（バイナリツリー）の一種であり，根（ルート）を先頭の節（ノード）として，最終世代以外のノードがすべて埋まっており，最終世代のノードは左から順に埋まっているものをいう。（図１）

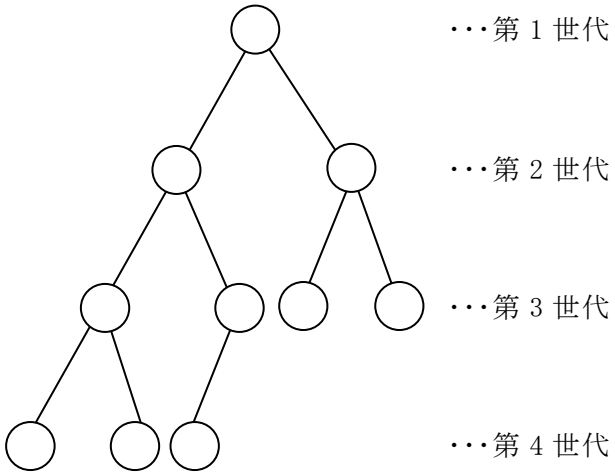


図１　完全２分木の例

ヒープ構造は，親子関係になっているどのノード間においても，（親ノードの値）≧（子ノードの値）が成り立っている完全 2 分木，あるいは，（親ノードの値）≦（子ノードの値）が成り立っている完全２分木をいう。（図２）

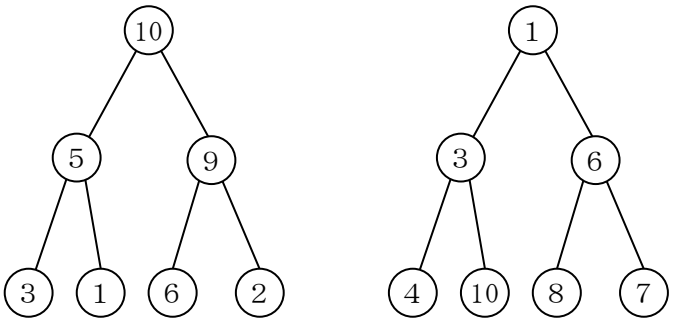


図２　ヒープ構造の例

木構造は，内部的にはリスト構造で表現する。したがって，2 分木もルートを起点とし，通常 2 つのポインタを使った連結リストで表現するが，ポインタを使用しない配列でも表現できる。

完全 2 分木を配列で表現する場合には，図 3 に示すように各ノードと各配列要素を対応付ける。したがって，配列の添字を 1 とした場合，ルートは添字 1 の配列要素に対応し，以下，世代順に，同一世代では左のノードから順番に対応付けられる。

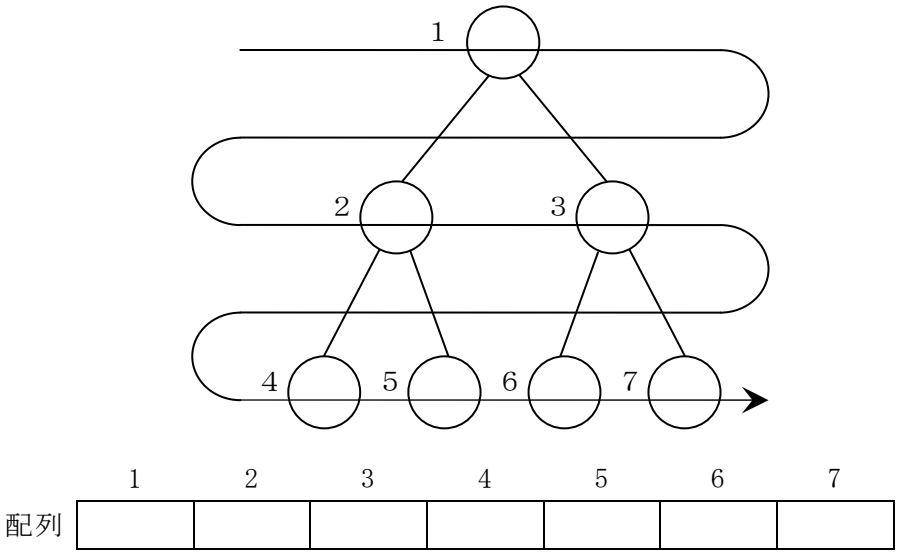


図 3　完全 2 分木と配列との対応関係

［ヒープ構造の作成に関する説明］

配列を使ってヒープ構造を作成するための考え方は以下のようになる。なお，配列の添字の先頭は 1 とし，ここでのヒープ構造は，（親の値）≧（子の値）とする。

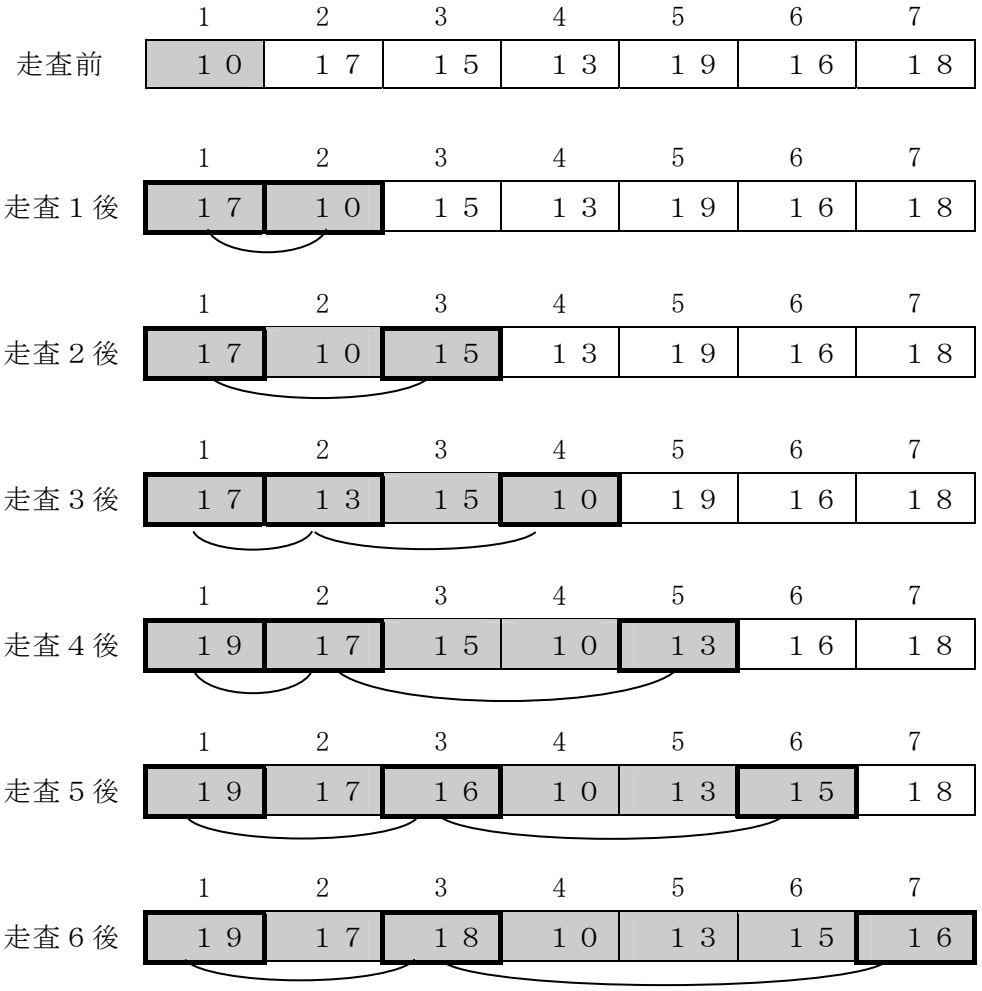
1．配列に N 個のデータを入力する。この時点において，全体としてはヒープ構造になっていない。ルートに該当する配列の先頭データだけを切り離して考えると，子がない状態であるため先頭の要素はヒープ構造であると考えられる。残りの（N－1）個のデータはヒープ構造ではない。

2．配列要素 2 から N までについて，以下の処理を繰り返す。

　2－1　各要素位置を最初の子として親の要素位置を計算する。

　2－2　比較する親がある，かつ，親の値よりも子の値が大きい間，親の値と子の値を入れ換え続ける。この時，親の要素位置は次の処理のために新しい子の要素位置になる。

実際に，N=7 の場合に配列データがヒープ構造になっていく様子を以下に示す。線で結んだ位置は，1 回の走査で比較または移動の対象となった要素位置である。



＜設問＞　次のヒープ構造を作成する流れ図中の［　　］に入るべき適切な字句を解答群から選べ。なお，データは配列 A[i]（i=1, 2, ･･･, N）に格納する。また，流れ図中の除算は小数点以下切捨てとする。

(1) の解答群

ア．A[j] ＜ A[k]　　イ．A[j] = A[k]

ウ．A[j] ＞ A[k]　　エ．A[j] ≠ A[k]

(2) の解答群

ア．j ← 1　　イ．j ← N

ウ．j ← i　　エ．j ← k

(3) の解答群

ア．i ← i ＋ 1　　イ．j ← j ＋ 1

ウ．k ← k ＋ 1　　エ．N ← N － 1

［流れ図］

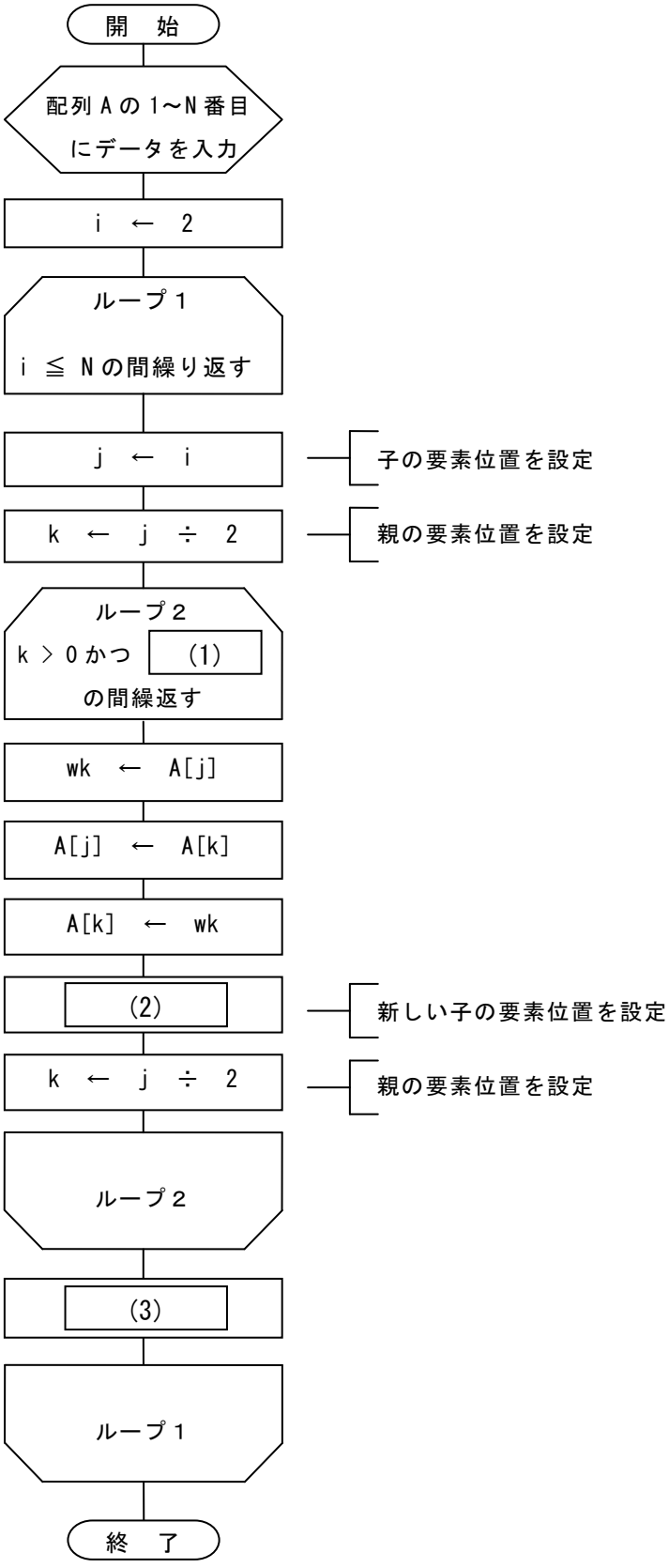


図４　ヒープ構造を作成する流れ図

問題３　次の売上一覧表作成に関する説明を読んで，設問に答えよ。

Ｊレストランは，日本全国に 100 店舗を構える 24 時間営業のチェーン店で，毎日午前 0 時～午後 23 時 59 分までの売上データを翌日午前 0 時に営業本部へ自動的に伝送する仕組みになっている（図 1 参照）。

営業本部では，各店舗から届いた売上データをバッチ処理で集計し，毎朝 8 時までに売上一覧表（図 2）を作成している。売上一覧表は，商品コードの昇順に出力され，同一商品コード内においては，店舗コードの昇順に出力される。

なお，各店舗から送られてきたデータから売上一覧表を作成するまでの流れは図 3 のとおりである。

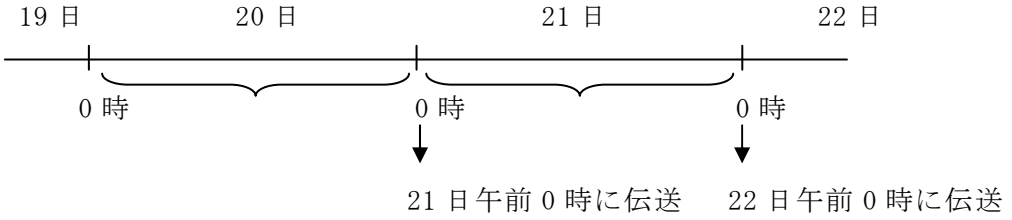


図１　営業本部へ伝送するタイミング

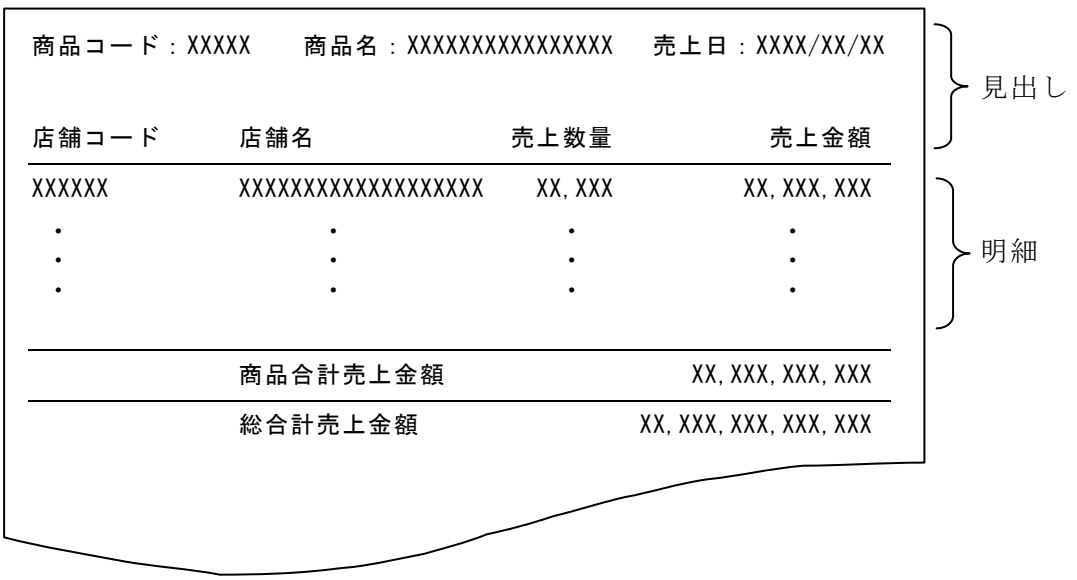


図２　売上一覧表の形式

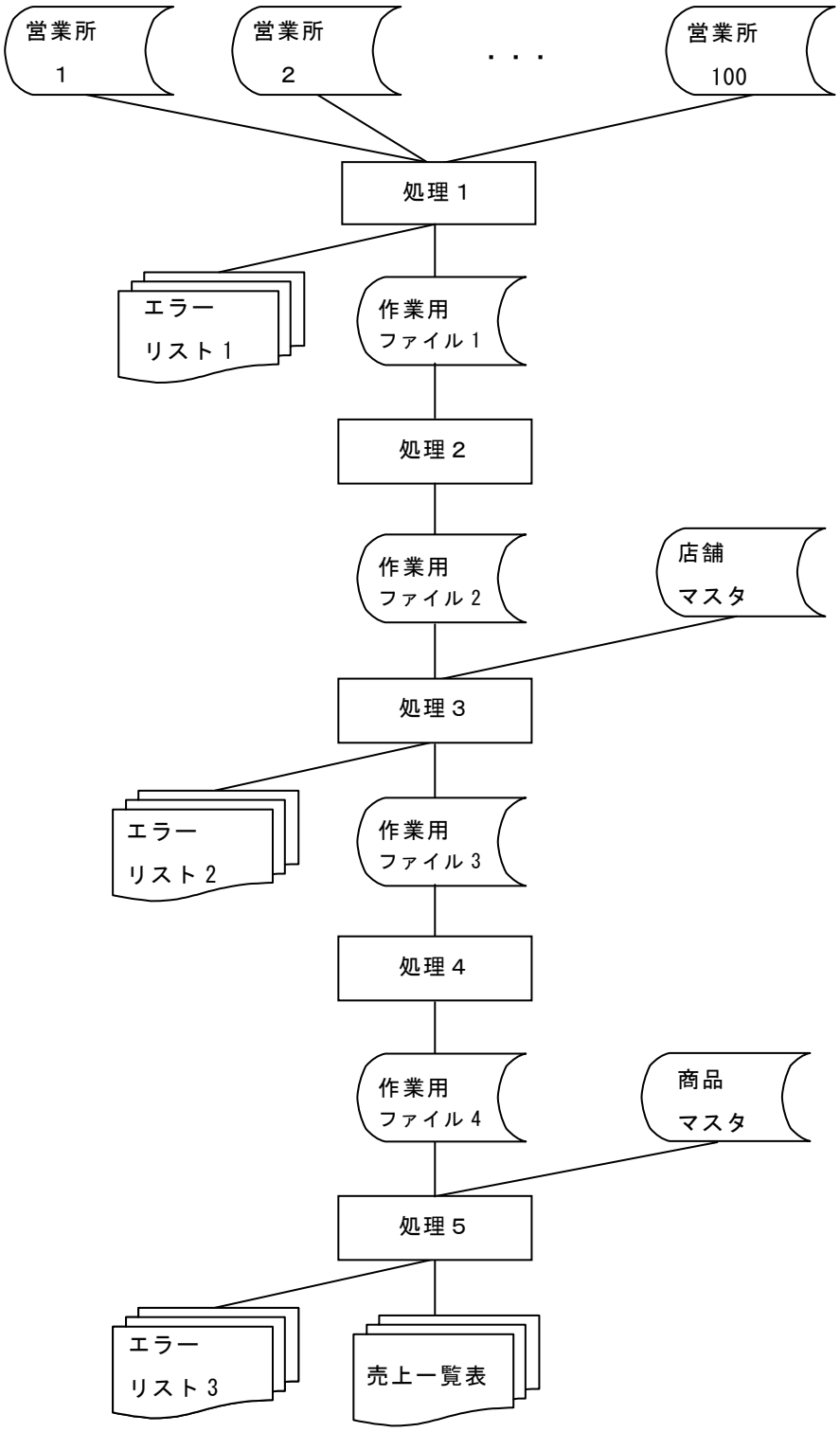


図３　売上一覧表を作成するまでの流れ

［レコードの形式］

・各店舗から送られてくる売上情報

店舗コード	商品コード	日付	売上数量

店舗ごとに商品コードの昇順に並んでいる。売上数量は，商品ごとにその日の売上数量を集計した値が入る。ただし，売上がなかった商品はデータを生成しない。なお，このデータは各店舗必ず 1 件以上存在する。

・店舗マスタ

店舗コード	店舗名

店舗コードは都道府県区分 2 桁と連番 3 桁，計 5 桁の数値で表現しており，99999 より小さい値である。

このファイルは順編成ファイルで店舗コードの昇順に整列済みであり，同一の店舗コードはない。

・商品マスタ

商品コード	商品名	単価

商品コードは分類区分 3 桁と連番 3 桁，計 6 桁の数値で表現しており，999999 より小さい値である。

このファイルは順編成ファイルで商品コードの昇順に整列済みであり，同一の商品コードはない。

［処理の概要］

・処理１　各営業所から送られてきた売上情報をレコードの形式を変更せず作業用ファイル 1 へ出力する。各店舗から送られたデータをチェックし，異常があればエラーリスト 1 へ出力する。

・処理２　作業用ファイル 1 を整列し作業用ファイル 2 を作成する。

・処理３　作業用ファイル 2 と店舗マスタを突き合わせ，作業用ファイル 3 を作成する。エラーがあれば，エラーリスト 2 へ出力する。

・処理４　作業用ファイル 3 を整列し作業用ファイル 4 を作成する。

・処理５　作業用ファイル 4 と商品マスタを突き合わせ，売上一覧表を作成する。エラーがあればエラーリスト 3 へ出力する。

＜設問１＞　処理 1 で出力するエラーリスト 1 の内容として適切でないものを解答群から選べ。

(1) の解答群

ア．店舗コードに数字以外の文字がある。
イ．商品コードがマスタ上に存在しない。
ウ．月が 12 を超えている。
エ．売上数量が 0 になっている。

＜設問２＞　処理 2 および処理 4 の整列処理で使用する最低限必要な項目を解答群から選べ。なお，項目名が 2 つある時は，左から順に第 1 キー，第 2 キーとする。
ただし，処理 2 の項目は (2) に，処理 4 の項目は (3) にマークせよ。

(2)，(3) の解答群

ア．店舗コード　　イ．商品コード
ウ．日付，店舗コード　　エ．商品コード，店舗コード
オ．売上数量，商品コード　　カ．商品コード，売上数量

＜設問３＞　次の処理 5 の流れ図中の［　　］に入るべき適切な字句を解答群から選べ。なお，商品マスタと作業用ファイル 4 のレコード形式は，図 4 のとおりである。また，印刷の条件は下記のとおりである。

［印刷の条件］

・1 ページあたり明細行を 50 行まで印字する。
・商品が変わった場合，50 行に満たなくとも改ページする。
・商品売上合計金額は，商品が変わった時だけ印字する。
・総合計売上金額は，最後のページだけ印字する。

商品マスタ

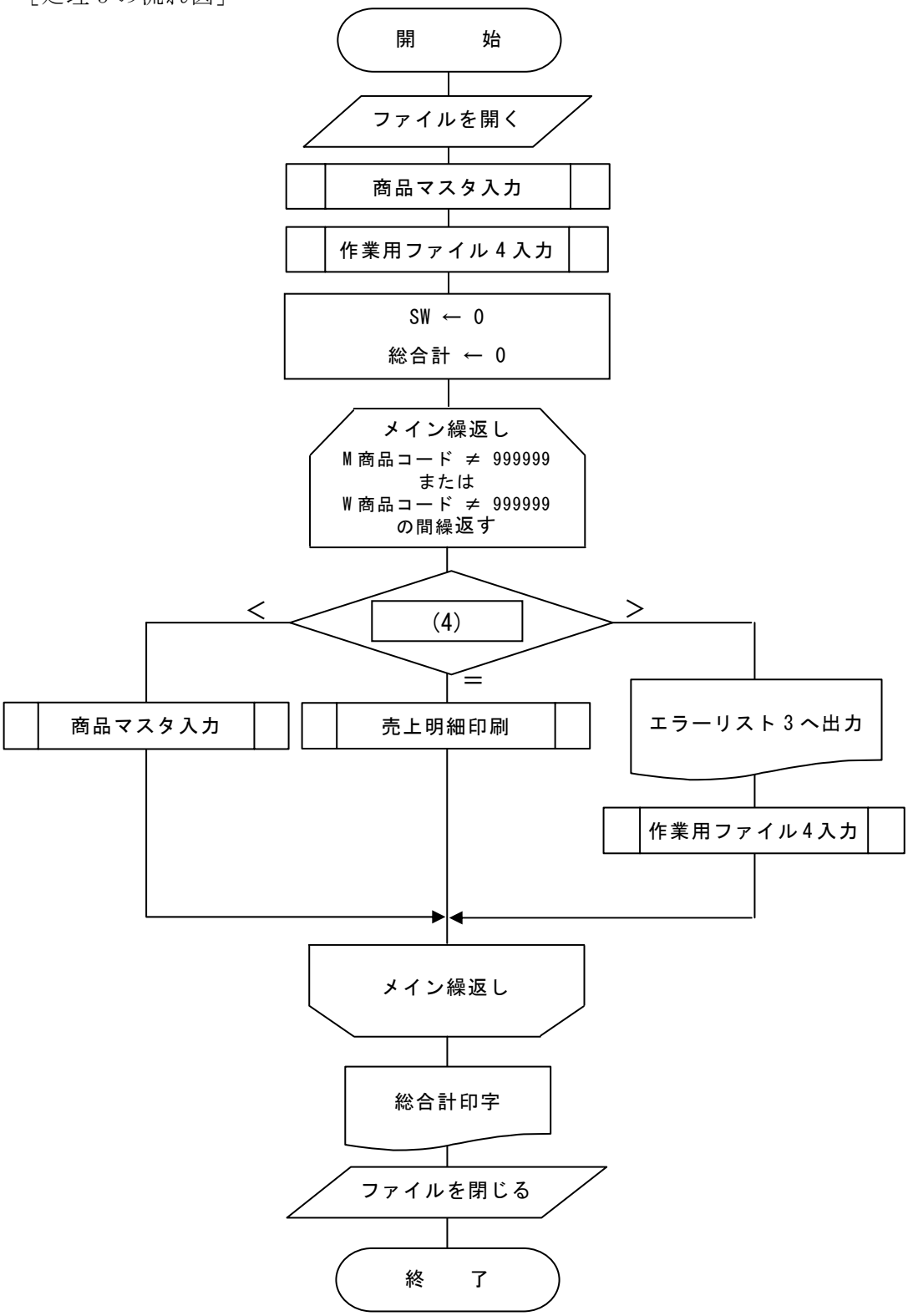
商品コード	商品名	単価
（変数名） M 商品コード	M 商品名	M 単価

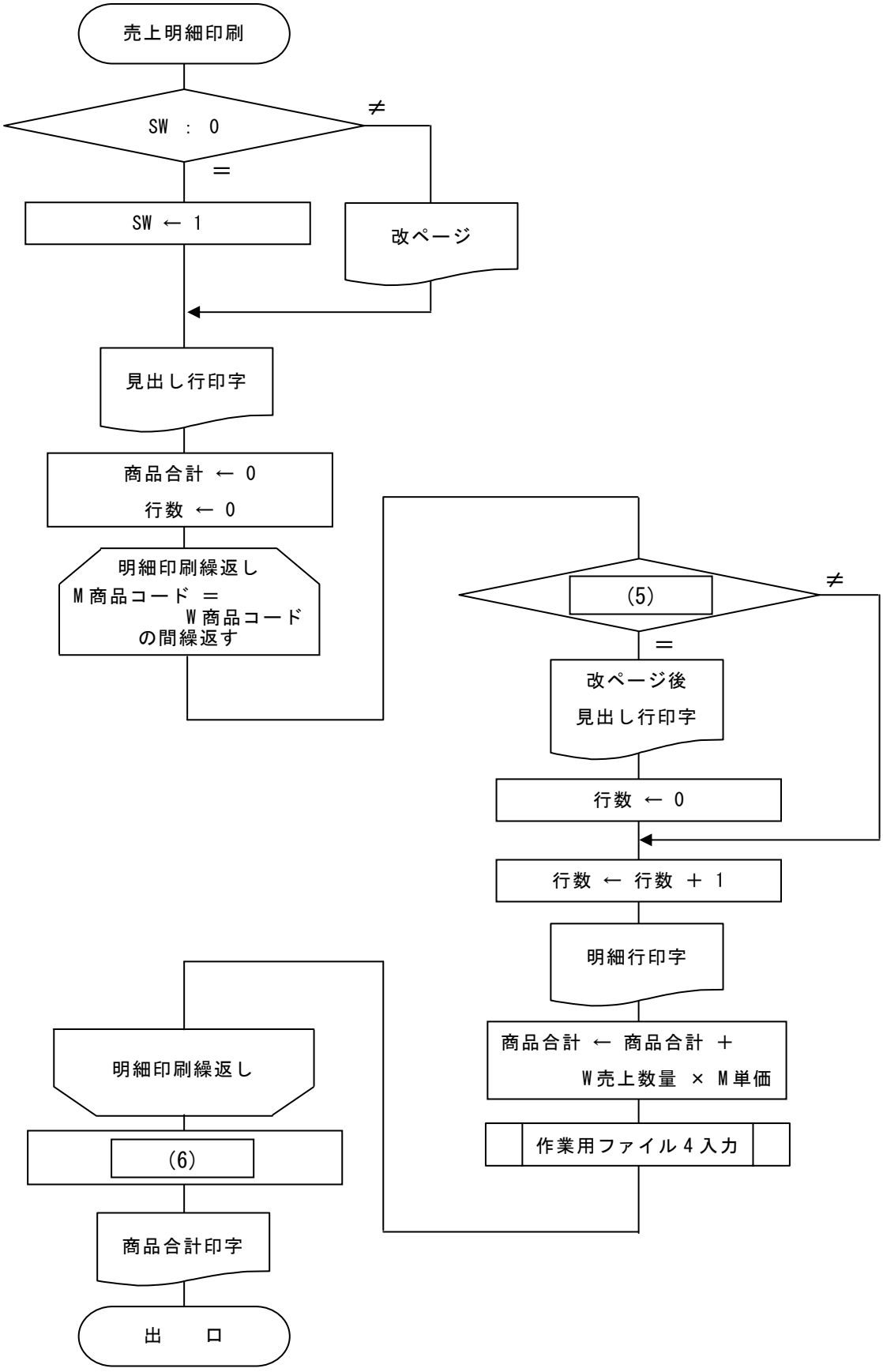
作業用ファイル 4

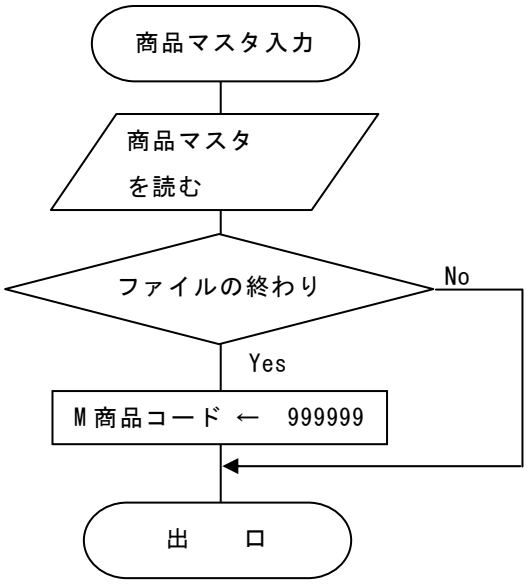
店舗コード	店舗名	商品コード	日付	売上数量
（変数名） W 店舗コード	W 店舗名	W 商品コード	W 日付	W 売上数量

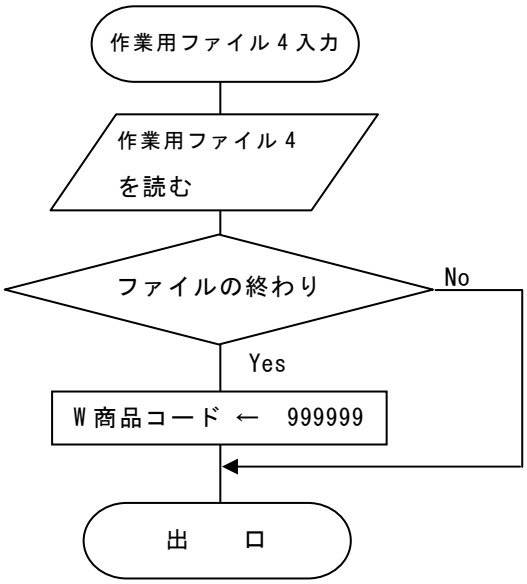
図４　商品マスタと作業用ファイル 4 の形式および変数名

［処理５の流れ図］


開　　始
ファイルを開く
商品マスタ入力
作業用ファイル４入力
SW ← 0
総合計 ← 0
メイン繰返し
M 商品コード ≠ 999999
または
W 商品コード ≠ 999999
の間繰返す
(4)
<
=
>
商品マスタ入力
売上明細印刷
エラーリスト３へ出力
作業用ファイル４入力
メイン繰返し
総合計印字
ファイルを閉じる
終　　了


売上明細印刷
SW ： 0
≠
＝
SW ← 1
改ページ
見出し行印字
商品合計 ← 0
行数 ← 0
明細印刷繰返し
M商品コード ＝
W商品コード
の間繰返す
(5)
≠
＝
改ページ後
見出し行印字
行数 ← 0
行数 ← 行数 ＋ 1
明細行印字
商品合計 ← 商品合計 ＋
W売上数量 × M単価
作業用ファイル４入力
明細印刷繰返し
(6)
商品合計印字
出　　口





(4) の解答群

ア．M 商品コード ： W 商品コード　　イ．W 商品コード ： M 商品コード

ウ．M 商品コード ： 999999　　エ．W 商品コード ： 999999

(5) の解答群

ア．行数 ： 50　　イ．行数 ： 51

ウ．SW ： 0　　エ．SW ： 1

(6) の解答群

ア．商品合計 ← W 売上数量 × M 単価

イ．商品合計 ← 商品合計 ＋ W 売上数量 × M 単価

ウ．総合計 ← 総合計 ＋ 商品合計

エ．総合計 ← 総合計 ＋ W 売上数量 × M 単価

＜設問４＞　エラーリスト３に出力されるデータについて,適切な記述を解答群から選べ。

(7) の解答群

ア．作業用ファイル４のレコードの中で，店舗マスタファイルに同じ店舗コードが存在しないレコードが出力される。

イ．作業用ファイル４のレコードの中で，商品マスタファイルに同じ商品コードが存在しないレコードが出力される。

ウ．商品マスタファイルに存在する最大の商品コードより大きな商品コードを持つ作業用ファイル４のレコードは出力されない。

エ．店舗マスタファイルに存在する最大の店舗コードより大きな店舗コードを持つ作業用ファイル４のレコードが出力される。

これより

＜ 選 択 問 題 ＞

選択問題はＡ，Ｂから，それぞれ１問選択し解答せよ。
選択した問題は必ず，解答用紙「選択欄」にマークすること。
※選択欄にマークがなく，解答のみの場合は採点を行いません。

各構成は以下のとおり。

選択問題　Ａ

選択問題　Ｂ

選択問題Ａ　C言語の問題

次のプログラムは，単語の出現数を数えるものである。プログラムの説明を読み設問に答えよ。

［プログラムの説明］

標準入力から入力された単語をポインタでつないだリスト構造でメモリに記憶し，出現した数を数えて出力する。

単語（word）
出現数（cnt）
次ポインタ（next）

図１　メモリに記憶する構造

１．処理の概要

単語は入力された順番に記憶領域中にリスト構造で格納し次ポインタで連結する。すでに格納されている単語が入力された場合はその数をカウントアップし，そうでなければ新しく要素を追加する。

なお，リストの先頭は変数 Top に格納する。Top の初期値は NULL である。また，リストの最後の次ポインタは NULL を格納する。

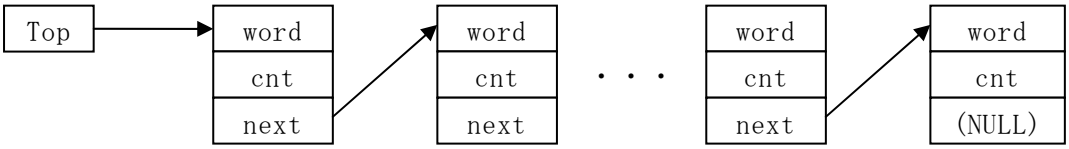


図２　リストの構造

２．標準入力からの入力について

標準入力から入力した単語の 1 文字目に‘#’が入力されるまで単語の数を数える処理を繰り返す。なお，入力する単語のバイト数は 80 バイト以内とする。

３．入出力例

```
単語を入れる:Ｃ言語
単語を入れる:Java
単語を入れる:Java
単語を入れる:インターネット
単語を入れる:ブラウザ
単語を入れる:Java
単語を入れる:#
```

図３　入力例

```
Ｃ言語 (1)
Java (3)
インターネット (1)
ブラウザ (1)
```

図４　出力例

［関数の仕様］

Entry 関数

引　数：単語

機　能：単語をリスト中から検索し，同じものがあれば出現数をカウントアップし，同じものが無ければ新しくリストに追加する。

戻り値：なし

newEntry 関数

引　数：単語

機　能：単語を格納する領域をメモリ中に確保する

戻り値：メモリ中に確保した場所を示すポインタ

Print 関数

引　数：なし

機　能：リストの先頭から単語と出現数を標準出力へ表示する

戻り値：なし

［プログラム］

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define LENGTH  80  /* 1つの単語の最大文字数 */
typedef struct dt {
    char word[LENGTH], int  cnt, struct dt *next;
} DATA;
DATA *newEntry(char *);
void Entry(char *);
void Print(void);
DATA *Top;  /* リストの先頭を格納する変数 */
int main(void) {
    char inbuff[LENGTH];
    Top = NULL;
    while (1) {
        printf("単語を入れる:");
        scanf("%s", inbuff);
        if (inbuff[0] == '#') break;
        Entry(inbuff);
    }
    Print();
    return (0);
```

```
}
/* 単語をリストから検索し，見つかればカウントアップし
     見つからなければリストの最後に追加する            */
void Entry(char *text) {
    DATA *newp, *p, *oldp;
    int sw;
    if (Top == NULL) { /* 最初のデータの場合 */
        newp = newEntry(text);
        (1) ;       /* リストの先頭を設定 */
    } else {           /* 同じ文字列を探す */
        p = Top;
        sw = 0;
        while ( (2) ) {
            if (strcmp(p->word, text)==0) {/* 同じ単語が見つかった時 */
                (3) ;
                sw = 1;
            }
            oldp = p;      /* 現在のポインタを退避 */
            p = p->next; /* 次のデータを調べるためにポインタを進める */
        }
        if (sw == 0) {  /* 初めて出現した単語の場合 */
            newp = newEntry(text);
            (4) ;
        }
    }
}

/* 新しく単語をリストに追加する */
DATA *newEntry(char *text) {
    DATA *p;
    p = (DATA *)malloc( sizeof(DATA) );
    strcpy(p->word, text);
    (5) ;
    p->next = NULL;
    return(p);
}
```

```
/* リストの先頭から単語と出現数を出力 */
void Print(void) {
    DATA *p;
    p = Top;
    while( p != NULL) {
        printf("%s (%d)¥n", p->word, p->cnt);
        p = p->next;
    }
}
```

＜設問＞　プログラム中の［　　］に入るべき適切な命令を解答群から選べ。

(1)，(4) の解答群

ア．`Top = newp`　　イ．`p->next = newp`

ウ．`newp->next = p`　　エ．`oldp->next = newp`

オ．`newp->next = NULL`　　カ．`p->next = NULL`

(2) の解答群

ア．`p == NULL && sw == 0`　　イ．`p != NULL && sw == 0`

ウ．`p == NULL || sw == 0`　　エ．`p != NULL || sw == 0`

(3)，(5) の解答群

ア．`p->cnt++`　　イ．`p->cnt = 0`

ウ．`p->cnt = 1`　　エ．`newp->cnt++`

オ．`newp->cnt = 0`　　カ．`newp->cnt = 1`

選択問題Ａ　Javaの問題

次のプログラムは，単語の出現数を数えるものである。プログラムの説明を読み設問に答えよ。

［プログラムの説明］

標準入力から入力された単語をリスト構造でメモリに記憶し，出現した数を数えて出力する。

単語（word）
出現数（cnt）
次インスタンス（next）

図１　記憶するインスタンスの構造

１．処理の概要

単語は入力された順番にインスタンスを生成して次インスタンスに格納することによってリスト構造を作成する。すでに格納されている単語が入力された場合はその数をカウントアップし，そうでなければ新しく要素を追加する。

なお，リストの先頭は top に格納する。top の初期値は null である。また，リストの最後の次インスタンスには null を格納する。

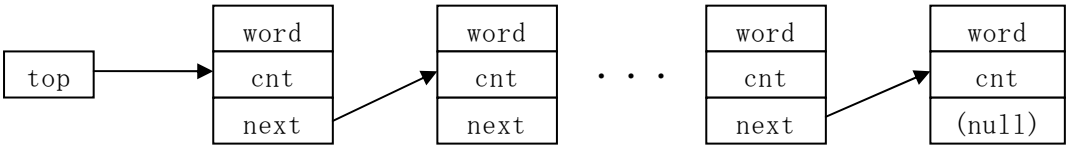


図２　リストの構造

２．標準入力からの入力について

標準入力から入力した単語の 1 文字目に‘#’が入力されるまで単語の数を数える処理を繰り返す。

３．入出力例

```
単語を入れる:Ｃ言語
単語を入れる:Java
単語を入れる:Java
単語を入れる:インターネット
単語を入れる:ブラウザ
単語を入れる:Java
単語を入れる:#
```

図３　入力例

```
Ｃ言語 (1)
Java (3)
インターネット (1)
ブラウザ (1)
```

図４　出力例

［クラスの仕様］

Data クラス

説明	入力された単語と出現数及びリスト構造を扱うクラス
フィールド	**word** ：入力された単語
	cnt ：単語の出現数
	next ：このインスタンスの次の要素
コンストラクタ	入力された単語と出現数をフィールドに設定し **next** に **null** を設定する
メソッド	**getCnt**：単語の出現数を返す
	putCnt：単語の出現数を設定する
	getWord：入力された単語を返す
	putWord：入力された単語を設定する
	getNext：このインスタンスの次の要素を返す
	putNext：このインスタンスに次の要素を設定する

EntryWord クラス

説明	入力された単語と出現数の集まりであるリスト構造を扱うクラス
フィールド	**top** ：リストの先頭のインスタンス
メソッド	**entry** ：単語をリスト内から検索し，同じものがあれば出現数をカウントアップし，同じものがなければ新しくリストに追加する。
	print ：リストの先頭から単語と出現数を標準出力へ表示する。
	newEntry：リストのインスタンスを作成する。

［プログラム１］

```
public class Data{
    private String word;
    private int cnt;
    private Data next;
    Data(String word,int cnt){
        this.word = word;
        this.cnt  = cnt;
        next = null;            /*次インスタンスを null に設定*/
    }
    public int getCnt(){
        return cnt;
    }
    public void putCnt(int cnt){
        this.cnt = cnt;
```

```
    }
    public String getWord(){
        return word;
    }
    public void putWord(String word){
        this.word = word;
    }
    public Data getNext(){
        return next;
    }
    public void putNext(Data next){
        this.next = next;
    }
}
```

［プログラム２］

```
public class EntryWord{
    Data top = null; /*リストの先頭を格納する変数（初期値 null）*/
    /*単語をリストから検索し，見つかればカウントアップし
        見つからなければリストの最後に追加する      */
    public void entry(String text){
        Data newp,p,oldp;
        int sw;
        if(top == null){                /*最初のデータの場合*/
            newp = newEntry(text);
            (1) ;                       /*リストの先頭を設定*/
        }else{                          /*同じ単語を探す*/
            p = top;
            oldp = top;
            sw = 0;
            while( (2) ){
                if(p.getWord().equals(text)){ /*同じ単語が見つかった時*/
                    (3) ;
                    sw = 1;
                }
                oldp = p;       /*単語を検索したインスタンスを退避*/
                p = p.getNext();        /*リストの次のインスタンスを呼び出す*/
            }
            if(sw == 0){                /*初めて出現した単語の場合*/
                newp = newEntry(text);
                (4) ;
```

```
            }
        }
    }
    /*新しく単語をリストに追加する*/
    public Data newEntry(String text){
        return  (5)  ;
    }
    /*リストの先頭から単語と出現数を出力*/
    public void print(){
        Data p;
        p = top;
        while(p != null){
            System.out.printf(
                        "%s (%d)¥n",p.getWord(),p.getCnt());
            p = p.getNext();
        }
    }
}
```

［プログラム３］

```
import java.io.*;
class TestEntry{
    public static void main(String[] args) throws IOException{
        /*標準入力から入力を受け取る br を作成する*/
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        EntryWord ew = new EntryWord();
        String inbuff;
        while(true){
            System.out.println("単語を入れる");
            /* 標準入力から１行読み込む*/
            inbuff = br.readLine();
            if(inbuff.charAt(0) == '#') break;
            ew.entry(inbuff);
        }
        ew.print();
    }
}
```

＜設問＞　プログラム中の［　　］に入るべき適切な命令を解答群から選べ。

(1)，(4) の解答群

ア．`top = newp`　　イ．`p.putNext(newp)`

ウ．`newp.putNext(p)`　　エ．`oldp.putnNext(newp)`

オ．`newp.putNext(null)`　　カ．`p.putNext(null)`

(2) の解答群

ア．`p == null && sw == 0`　　イ．`p != null && sw == 0`

ウ．`p == null || sw == 0`　　エ．`p != null || sw == 0`

(3) の解答群

ア．`p.putCnt(cnt+1)`　　イ．`p.putCnt(p.getCnt()+1)`

ウ．`p.putCnt(1)`　　エ．`p.putCnt(p.getCnt())`

(5) の解答群

ア．`new Data()`　　イ．`new Data(text,1)`

ウ．`new Data(text,top.getCnt())`　　エ．`new Data(text,1,null)`

選択問題Ａ　アセンブラの問題

次のＣＡＳＬⅡプログラムの説明を読んで，設問に答えよ。

［プログラムの説明］

文字列の中から指定された部分文字列を探索する副プログラムである。主プログラムでは図１のように文字列および部分文字列が格納されている。**GR1～GR4** の各レジスタは表１のように設定され，副プログラムが呼び出される。副プログラムでは部分文字列が見つかった場合は部分文字列の開始位置を，見つからない場合は-1 を **GR0** に設定し，主プログラムに戻る。

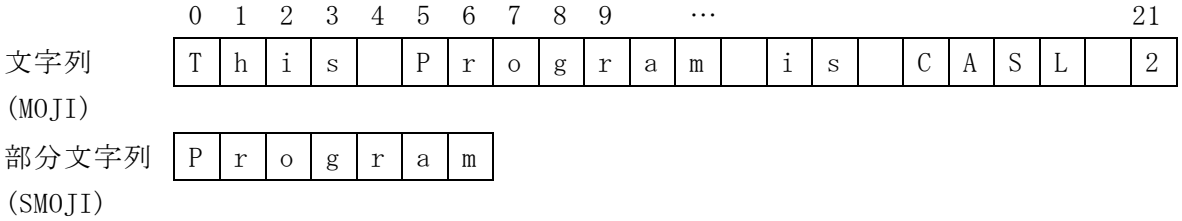


図１　文字列および部分文字列の格納例

この状態で副プログラムが呼び出された場合，**GR0** には５が格納されて主プログラムに戻る。

表１　各レジスタの内容

GR1	文字列の先頭(MOJI)の絶対番地
GR2	部分文字列(SMOJI)の絶対番地
GR3	文字列の文字数
GR4	部分文字列の文字数

＜設問１＞　プログラム中の［　　］に入れるべき適切な命令を解答群から選べ。

(1) の解答群

ア．**CPA GR0,GR2**　　イ．**CPA GR0,0,GR2**

ウ．**CPA GR0,GR6**　　エ．**CPA GR0,0,GR6**

(2) ～ (4) の解答群

ア．**JUMP LOOP0**　　イ．**JUMP LOOP1**　　ウ．**JUMP FOUND**

エ．**JZE FOUND**　　オ．**JMI LOOP0**　　カ．**JNZ LOOP1**

［プログラム］

```
100  SUBRTN   START
110           SUBL     GR3,GR4     ;（文字列の文字数-部分文字列の文字数）
120           ADDL     GR3,GR1     ;  を GR3 に設定する
130  ;                             （空白行）
140           ADDL     GR4,GR2     ;検索文字の最終位置
150           SUBL     GR4,=1      ;（絶対番地）
160           PUSH     0,GR1       ;GR1 の退避
170  LOOP0    LD       GR5,GR1
180           LD       GR6,GR2
190  LOOP1    LD       GR0,0,GR5
200           |      (1)      |    ;文字の比較
210           JNZ      L1
220           CPL      GR6,GR4     ;部分文字列がすべて一致したか？
230           |      (2)      |
240           LAD      GR5,1,GR5
250           LAD      GR6,1,GR6
260           |      (3)      |    ;部分文字列の次の文字を比較へ分岐
270  L1       LAD      GR1,1,GR1
280           CPL      GR1,GR3
290           JPL      NFOUND
300           |      (4)      |    ;部分文字列の先頭から比較開始する
310  FOUND    LD       GR0,GR1
320           POP      GR1
330           SUBL     GR0,GR1     ;開始位置を GR0 に求める
340           JUMP     EXIT
350  NFOUND   LD       GR0,=-1
360  EXIT     RET
370           END
```

＜設問２＞　主プログラムで文字列および部分文字列が図１のように格納されて，副プログラムが呼び出されたとき，プログラム中のラベル **L1** の命令は何回実行されるか解答群から選べ。

(5) の解答群

ア．０回　　イ．２回　　ウ．５回　　エ．６回

選択問題Ｂ　C言語の問題

次のプログラムは，日数計算をするものである。プログラムの説明を読み設問に答えよ。

［プログラムの説明］

標準入力から機能タイプ（0 または 1）を選択し，以下の 2 種類の計算をする。

機能タイプ 0：自年月日から指定日数後の日付を求めて出力する。

機能タイプ 1：自年月日から至年月日までの期間日数を求めて出力する。

＜実行例＞

```
機能タイプ(0:指定日数 1:期間日数):0
自年月日:2008/2/1
指定日数:29
2008/3/1
```

図１　機能タイプ 0 の例

```
機能タイプ(0:指定日数 1:期間日数):1
自年月日:2008/2/1
至年月日:2008/3/1
30 日間
```

図２　機能タイプ 1 の例

１．入力値のチェック

以下のチェックを行い，入力値に誤りがある場合はエラーメッセージを表示して終了する。

チェック項目	チェック内容
日付チェック	・基準年月日（2000 年 1 月 1 日）以降の日付であること。 ・月日はうるう年を考慮し，実在する日付であること。
指定日数チェック	・1 以上の整数であること。

２．計算機能

２．１　機能タイプ 0 の計算

指定された自年月日に指定日数を加算し，日付を求める。

２．２　機能タイプ 1 の計算

基準年月日から指定された自年月日までの日数と基準年月日から指定された至年月日までの日数をもとに期間日数を求める。

なお，基準年月日は 2000 年 1 月 1 日とする。

３．うるう年の計算

次の 2 つの条件のうち，いずれかを満たす年がうるう年である。

①　4 で割り切れ，かつ，100 で割り切れない年。

②　400 で割り切れる年。

［関数の仕様］

DayC 関数

引　数：自年月日，指定日数

機　能：自年月日から指定日数後の日付を計算する

戻り値：指定日数後の日付

DaySyear 関数

引　数：年月日

機　能：2000 年 1 月 1 日から指定した年月日間の日数を計算する

戻り値：日数

isLeap 関数

引　数：年

機　能：年がうるう年かどうかを判定する

戻り値：うるう年のとき 1，平年のとき 0

Check 関数

引　数：機能タイプ，自年月日，至年月日，指定日数

機　能：指定した機能タイプによって年月日や指定日数を検査する

戻り値：エラーの種類により，**int** 型の下位 3 ビットを以下のようにビット単位で値を設定して生成したエラーコード

未使用（0 にする）	①	②	③

① 至年月日に誤りがあるとき 1，そうでなければ 0

② 指定日数に誤りがあるとき 1，そうでなければ 0

③ 自年月日に誤りがあるとき 1，そうでなければ 0

ErrMsg 関数

引　数：エラーコード

機　能：エラーコードの値によりメッセージを標準出力に出力する

戻り値：なし

［プログラム］

```
#include <stdio.h>
#include <stdlib.h>
#define SYEAR 2000          /* 2000 年を定義*/
typedef struct date {
    int yy, mm, dd;
} DATE;
int daytab[2][13] = { {0,31,28,31,30,31,30,31,31,30,31,30,31},
                      {0,31,29,31,30,31,30,31,31,30,31,30,31} };
```

```
DATE DayC(DATE, int);
int Check(int, DATE, DATE, int);
int isLeap(int);
int DaySyear(DATE);
void ErrMsg(int);

int main(void) {
    DATE dt1, dt2={SYEAR, 1, 1};
    int type, code, day=0;

    printf("機能タイプ(0:指定日数 1:期間日数):");
    scanf("%d", &type);
    if (type < 0 || type > 1)
        printf("機能タイプが違います¥n");
    else {
        printf("自年月日:");
        scanf("%d/%d/%d", &dt1.yy, &dt1.mm, &dt1.dd);
        if (type == 0) {  /* 機能タイプ0の時の処理 */
            printf("指定日数:");
            scanf("%d", &day);
            code = Check(type, dt1, dt2, day);
            if (code == 0) {
                dt2 = DayC(dt1, day);
                printf("%d/%d/%d¥n", dt2.yy, dt2.mm, dt2.dd );
            } else
                ErrMsg(code);
        } else {  /* 機能タイプ1の時の処理 */
            printf("至年月日:");
            scanf("%d/%d/%d", &dt2.yy, &dt2.mm, &dt2.dd);
            code = Check(type, dt1, dt2, day);
            if (code == 0)
                printf("%d日間¥n",
                        abs(DaySyear(dt1) - DaySyear(dt2)) + 1 );
            else
                ErrMsg(code);
        }
    }
    return (0);
}
```

```
DATE DayC(DATE dt1, int day) {
    int k;
    DATE dtw = dt1;
    dtw.dd += day;
    k = isLeap(dtw.yy);
    /* 日がある月の日数を下回るまで月をカウントアップ */
    while(dtw.dd > daytab[k][dtw.mm]) {
        dtw.dd -= daytab[k][dtw.mm];
        dtw.mm++;
        if(dtw.mm > 12) { /* 12 月を超えた場合 */
            dtw.yy++;
            (1) ;
            k = isLeap(dtw.yy);
        }
    }
    return(dtw);
}

int DaySyear(DATE dtm2) {
    int days=0, k, m;
    DATE dtm1={SYEAR, 1, 1}; /* 基準年月日を設定 */
    /* 基準年月日から自年月日の年までの日数を加算 */
    while(dtm1.yy < dtm2.yy) {
        (2) ;
        dtm1.yy++;
    }
    k = isLeap(dtm2.yy);
    /* 自年月日の月までの日数を加算 */
    for(m = 1; m < dtm2.mm; m++) days += daytab[k][m];
    /* 自年月日の日を加算 */
    days += dtm2.dd;
    return(days);
}

int isLeap(int yy) {
    if( (3) )
        return(1);
    else
        return(0);
}
```

```
int Check(int type, DATE dt1, DATE dt2, int day) {
    int k, code;
    k = isLeap(dt1.yy);
    code = 0;
    if( dt1.yy<SYEAR || dt1.mm<1 || dt1.mm>12 ||
        dt1.dd<1 || dt1.dd>daytab[k][dt1.mm]) code = 0x01;
    if( [ (4) ] ) {
        k = isLeap(dt2.yy);
        if( dt2.yy<SYEAR || dt2.mm<1 || dt2.mm>12 ||
            dt2.dd<1 || dt2.dd>daytab[k][dt2.mm])
            code = code | 0x04;
    } else
        if(day <= 0) code = code | 0x02;
    return(code);
}

void ErrMsg(int code) {
    if( [ (a) ] ) printf("自年月日が無効です。¥n");
    if( [ (b) ] ) printf("指定日数が無効です。¥n");
    if( [ (c) ] ) printf("至年月日が無効です。¥n");
}
```

＜設問１＞　プログラム中の［　　］に入るべき適切な字句を解答群から選べ。

(1) の解答群

ア. `dtw.mm = 1`　　イ. `dtw.mm = 12`

ウ. `dtw.mm++`　　エ. `dtw.mm--`

(2) の解答群

ア. `days += 365`

イ. `days += 366`

ウ. `days += 365 + isLeap(dtm1.yy)`

エ. `days += 365 + isLeap(dtm2.yy)`

(3) の解答群

ア. `yy%4 == 0 || yy%100 != 0 || yy%400 == 0`

イ. `yy%4 == 0 && yy%100 != 0 && yy%400 == 0`

ウ. `yy%4 == 0 || yy%100 != 0 && yy%400 == 0`

エ. `yy%4 == 0 && yy%100 != 0 || yy%400 == 0`

(4) の解答群

ア．**type == 0**　　イ．**type == 1**

ウ．**type < 0**　　エ．**type > 1**

＜設問２＞　プログラム中の（a）～（c）に入るべき字句の組み合わせとして適切なものを（5）の解答群から選べ。

(5) の解答群

	(a)	(b)	(c)
ア．	**code & 0x01**	**code & 0x02**	**code & 0x04**
イ．	**code & 0x01**	**code & 0x04**	**code & 0x02**
ウ．	**code & 0x02**	**code & 0x01**	**code & 0x04**
エ．	**code & 0x04**	**code & 0x02**	**code & 0x01**

＜設問３＞　次に示すデータを標準入力から入力したとき，Check 関数の戻り値として適切なものを解答群から選べ。

機能タイプ	自年月日	日数	至年月日	戻り値
0	**2008/2/5**	**31**		(6)
0	**2008/12/1**	**-1**		(7)
1	**2008/11/31**		**2009/2/30**	(8)

(6) ～ (8) の解答群

ア．**0x00**　イ．**0x01**　ウ．**0x02**　エ．**0x03**

オ．**0x04**　カ．**0x05**　キ．**0x06**　ク．**0x07**

選択問題Ｂ　Javaの問題

次のプログラムは，図書の貸出と返却処理を行うものである。プログラムの説明を読み設問に答えよ。

[プラグラムの説明]

現在図書館にある蔵書一覧を作成し，貸出時にはその要素を削除し，返却時に要素を追加することで図書を管理している。また，貸出時に利用者と貸し出す図書を貸出情報として作成し，返却時に返却済みと登録をすることで貸出情報の管理をしている。

TestLibrary 実行時の出力結果を以下に示す。

```
貸出　名前：田中一郎(ID001)は書籍名：プログラミングスキル参考書(B001)を借りました
貸出　名前：田中一郎(ID001)は書籍名：システムデザインスキル参考書(B002)を借りました
貸出　名前：佐藤次郎(ID002)は書籍名：基本スキル参考書(B003)を借りました
返却　名前：田中一郎(ID001)は書籍名：プログラミングスキル参考書(B001)を返却した
貸出不可　名前：佐藤次郎(ID002)は書籍名：システムデザインスキル参考書(B002)を借りようとしましたが貸し出し中です
返却　名前：佐藤次郎(ID002)は書籍名：基本スキル参考書(B003)を返却した
貸出　名前：田中一郎(ID001)は書籍名：基本スキル参考書(B003)を借りました
貸出　名前：田中一郎(ID001)は書籍名：プログラミングスキル参考書(B001)を借りました
返却不可　名前：佐藤次郎(ID002)は書籍名：プログラミングスキル参考書(B001)を借りようとしましたが貸し出し中です
```

図１　出力結果１

プログラムで使用している **ArrayList** クラスは通常の配列と異なり，要素を追加するたびにサイズが自動的に長くなる配列のように扱うことができるものである。以下に，**ArrayList** クラスの使用例を示す。

・インポート指定（**ArrayList** クラスは **java.util** パッケージにある）

例：**import java.util.ArrayList;**

・インスタンスの確保

例：**ArrayList** 型の変数 **list** を初期化する。

但し，**list** には **String** 型のインスタンスしか入れることができない。

```
ArrayList<String> list = new ArrayList<String>();
```

上記の形式で初期化すると，**list** の要素は **String** 型であると保証されるため，使用するときに明示的なキャストをする必要が無い。

・要素の追加（**add** メソッド）

例：変数 **list** が示す **ArrayList** に文字列**"sample"**を追加する。

```
list.add("sample");
```

（同じ値を持つ要素が存在していても，常に最後尾に追加される。）

・要素の削除（**remove** メソッド）

例：変数 **list** が示す **ArrayList** の先頭から **2** 番目の要素を削除する。

```
list.remobe(1);
```

（先頭の要素は **0** 番目として扱う）

・要素の比較（**contains** メソッド）

例：変数 **list** が示す **ArrayList** 内の全ての要素と比較をして **obj** と同じ要素があるかを確認する

```
boolean x = list.contains(obj);
```

（**list** 内にあれば **true**，無ければ **false** を返す）

・要素の取り出し（**indexOf** メソッド）

例：変数 **list** が示す **ArrayList** の先頭から**"sample"**と同じ内容の要素を検索し，その要素の場所を **x** に代入する

```
x = list.indexOf("sample");
```

（先頭から検索し最初に見つかった要素の場所（先頭の要素は **0** 番目として扱う）を返す。見つからなかった場合は**-1** を返す）

プログラムでは，拡張された **for** 文を使用している。以下に，その使用例を示す。

使用例：**ArrayList** 型の **item** の先頭から順に値を取り出し，**Integer** 型の **i** に代入して表示することを **item** の要素分繰り返す。

```
ArrayList<Integer> item = new ArrayList<Integer>();
item.add(1);
item.add(2);
for(Integer i:item)
        System.out.println(i);
```

出力結果

```
1
2
```

[クラスの説明]

User クラス

説明　　利用者を表すクラス

フィールド	**name**	利用者名
	id	利用者番号
メソッド	**getId**	利用者番号を返す
	toString	フィールドの内容を加工して文字列で返す

Book クラス

説明		書籍を表すクラス
フィールド	**name**	書籍名
	id	書籍番号
メソッド	**getId**	書籍番号を返す
	toString	フィールドの内容を加工して文字列で返す

Rental クラス

説明		どの利用者がどの書籍を借りているかを表すクラス
フィールド	**user**	利用者情報
	book	書籍情報
	returnBook	真なら返却済み，偽なら未返却
メソッド	**getUser**	利用者情報を返す
	getBook	書籍情報を返す
	setTrue	返却済みに設定する

Library クラス

説明		貸出処理と返却処理を行うクラス
フィールド	**libraryTable**	現在図書館に存在する図書の一覧
	rentalTable	貸出情報の一覧
コンストラクタ	**libraryTable**	を作成する
メソッド	**checkout**	貸出処理
	returnBook	返却処理

なお，このクラスのインスタンスを複数のスレッドが同時に使用することはないものとする。

［プログラム１］

```
public class User{
    private String name;
    private String id;
    User(String name,String id){
        this.name = name;
        this.id = id;
    }
    public String getId(){
        return id;
    }
```

```
    public String toString(){
        return "名前：" + name + "(" + id + ")";
    }
}
```

[プログラム２]

```
public class Book{
    private String name;
    private String id;
    Book(String name,String id){
        this.name = name;
        this.id = id;
    }
    public String getId(){
        return id;
    }
    public String toString(){
        return "書籍名：" + name + "(" + id + ")";
    }
}
```

[プログラム３]

```
public class Rental{
    private User user;
    private Book book;
    private boolean returnBook;
    Rental(User user,Book book,boolean returnBook){
        this.user = user;
        this.book = book;
        this.returnBook = returnBook;
    }
    public User getUser(){
        return user;
    }
    public Book getBook(){
        return book;
    }
    public void setTrue(){
        returnBook = true;
    }
```

```
    public boolean getReturnBook(){
        return returnBook;
    }
}
```

[プログラム４]

```
import java.util.ArrayList;
public class Library{
    /*貸し出されていない図書一覧 libraryTable の定義*/
    ArrayList<Book> libraryTable = new ArrayList<Book>();
    /*貸出状況の一覧 rentalTable の定義*/
    ArrayList<Rental> rentalTable = new ArrayList<Rental>();
    /*貸し出されていない図書一覧の初期状態の設定*/
    Library(Book[] books){
        for(Book book : books)
            (1) ;
    }
    public void checkout(User user,Book book){
        if( (2) ){
            System.out.println("貸出不可  " + user + "は"
                + book + "を借りようとしましたが貸し出し中です");
        }else{
            /*貸出状況への追加*/
            rentalTable.add( (3) );
            /*貸し出されていない図書一覧から削除*/
            libraryTable.remove( (4) );
            System.out.println(
                    "貸出  " + user + "は" + book + "を借りました");
        }
    }

    public void returnBook(User user,Book book){
        int sw = 0;
        for(Rental re:rentalTable){
            if( (5) ){
                re.setTrue();
                sw = 1;
            }
        }
```

```
        if(sw == 0)
            System.out.println("返却不可 "+ user + "は"
                + book + "を返却できません");
        else{
            System.out.println("返却  " + user + "は"
                + book +"を返却した");
            /*貸し出されていない図書一覧への追加*/
            [   (1)   ];
        }
    }
}
```

［プログラム５］

```
class TestLibrary{
    public static void main(String[] args){
        Book proskill = new Book("プログラミングスキル参考書","B001"),
            sysskill = new Book("システムデザインスキル参考書","B002"),
            basskill = new Book("基本スキル参考書","B003");
        Book[] books = {proskill,sysskill,basskill};
        Library library = new Library(books);
        User ichiro = new User("田中一郎","ID001"),
            jiro = new User("佐藤次郎","ID002");
        library.checkout(ichiro,proskill);
        library.checkout(ichiro,sysskill);
        library.checkout(jiro,basskill);
        library.returnBook(ichiro,proskill);
        library.checkout(jiro,sysskill);
        library.returnBook(jiro,basskill);
        library.checkout(ichiro,basskill);
        library.checkout(ichiro,proskill);
        library.checkout(jiro,proskill);
    }
}
```

＜設問１＞　プログラム中の［　　］に入るべき適切な命令を解答群から選べ

(1)，(4) の解答群

ア．**libralyTable.add(book)**　　イ．**libraryTable.indexOf(book)**

ウ．**libralyTable.remove(book)**　　エ．**rentalTable.add(book)**

オ．**rentalTable.indexOf(book)**　　カ．**rentalTable.remove(book)**

(2) の解答群

ア．**libraryTable.contains(book)**

イ．**!libraryTable.contains(book)**

ウ．**rentalTable.contains(book)**

エ．**!rentalTable.contains(book)**

(3) の解答群

ア．**user,book,true**　　イ．**user,book,false**

ウ．**new Rental(user,book,true)**　　エ．**new Rental(user,book,false)**

(5) の解答群

ア．
```
re.getUser().getId() == user.getId() &&
    re.getBook().getId() == book.getId()
```

イ．
```
re.getUser().getId() == user.getId() &&
    re.getBook().getId() == book.getId() &&
    re.getReturnBook() == false
```

ウ．
```
re.getUser().getId().equals(user.getId()) &&
    re.getBook().getId().equals(book.getId())
```

エ．
```
re.getUser().getId().equals(user.getId()) &&
    re.getBook().getId().equals(book.getId()) &&
    re.getReturnBook() == false
```

＜設問２＞　設問１のプログラムに貸出処理後の貸出情報一覧と図書館に存在する図書一覧を表示させるためにプログラム４に **checklibrary** メソッドを追加した。また，プログラム５を次のように変更した場合，実行結果はどのようになるか図２の［　　］に入れるべき適切な字句を解答群から選べ。

[プログラム４に以下のメソッドを追加]

```
public void checkLibrary(){
    System.out.println("貸出情報一覧");
    for(Rental re:rentalTable)
        System.out.println(re.getUser()+" "+re.getBook()+
            " "+(re.getReturnBook()?"返却済":"貸出中"));
    System.out.println("現在図書館に存在する図書一覧");
    for(Book book:libraryTable)
        System.out.println(book);
}
```

[変更後プログラム５]

```
class TestLibrary{
```

```
    public static void main(String[] args){
        Book proskill = new Book("プログラミングスキル参考書","B001"),
             sysskill = new Book("システムデザインスキル参考書","B002"),
             basskill = new Book("基本スキル参考書","B003");
        Book[] books = {proskill,sysskill,basskill};
        Library library = new Library(books);
        User ichiro = new User("田中一郎","ID001"),
             jiro  = new User("佐藤次郎","ID002");
        library.checkout(ichiro,sysskill);
        library.checkout(jiro,basskill);
        library.checkout(jiro,sysskill);
        library.checkout(jiro,proskill);
        library.returnBook(ichiro,sysskill);
        library.returnBook(ichiro,basskill);
        library.returnBook(jiro,proskill);
        library.checkLibrary();
    }
}
```

```
貸出　名前：田中一郎(ID001)は書籍名：システムデザインスキル参考書(B002)を借りました
貸出　名前：佐藤次郎(ID002)は書籍名：基本スキル参考書(B003)を借りました
貸出不可　名前：佐藤次郎(ID002)は書籍名：システムデザインスキル参考書(B002)を借り
ようとしましたが貸し出し中です
貸出　名前：佐藤次郎(ID002)は書籍名：プログラミングスキル参考書(B001)を借りました
返却　名前：田中一郎(ID001)は書籍名：システムデザインスキル参考書(B002)を返却した
返却不可 名前：田中一郎(ID001)は書籍名：基本スキル参考書(B003)を返却できません
返却　名前：佐藤次郎(ID002)は書籍名：プログラミングスキル参考書(B001)を返却した
貸出情報一覧
```

(6)

現在図書館に存在する図書一覧

(7)

図２　実行結果２

(6) の解答群

ア.　名前：佐藤次郎(ID002) 書籍名：基本スキル参考書(B003) 貸出中

イ.　名前：田中一郎(ID001) 書籍名：システムデザインスキル参考書(B002) 返却済
名前：佐藤次郎(ID002) 書籍名：基本スキル参考書(B003) 貸出中
名前：佐藤次郎(ID002) 書籍名：プログラミングスキル参考書(B001) 返却済

ウ.　名前：田中一郎(ID001) 書籍名：システムデザインスキル参考書(B002) 返却済
名前：佐藤次郎(ID002) 書籍名：基本スキル参考書(B003) 貸出中
名前：佐藤次郎(ID002) 書籍名：システムデザインスキル参考書(B002) 貸出中
名前：佐藤次郎(ID002) 書籍名：プログラミングスキル参考書(B001) 返却済

エ.　名前：田中一郎(ID001) 書籍名：システムデザインスキル参考書(B002) 貸出中
名前：佐藤次郎(ID002) 書籍名：基本スキル参考書(B003) 貸出中
名前：佐藤次郎(ID002) 書籍名：システムデザインスキル参考書(B002) 貸出中
名前：佐藤次郎(ID002) 書籍名：プログラミングスキル参考書(B001) 貸出中
名前：田中一郎(ID001) 書籍名：システムデザインスキル参考書(B002) 返却済
名前：田中一郎(ID001) 書籍名：基本スキル参考書(B003) 返却済
名前：佐藤次郎(ID002) 書籍名：プログラミングスキル参考書(B001) 返却済

(7) の解答群

ア.　書籍名：基本スキル参考書(B003)

イ.　書籍名：基本スキル参考書(B003)
書籍名：システムデザインスキル参考書(B002)

ウ.　書籍名：システムデザインスキル参考書(B002)
書籍名：プログラミングスキル参考書(B001)

エ.　書籍名：プログラミングスキル参考書(B001)
書籍名：システムデザインスキル参考書(B002)
書籍名：基本スキル参考書(B003)

選択問題Ｂ　アセンブラの問題

次のＣＡＳＬⅡプログラムの説明を読んで，設問に答えよ。

［プログラムの説明］

プログラム 1 はリスト構造を持つデータの中から指定したデータを探索する副プログラムである。主プログラムの LIST 番地から始まる連続した領域にデータとポインタをセットにして，図 1 のように格納されている。ポインタは次のデータの格納位置を表す。先頭のデータのポインタは(LIST+1)番地に格納されており,最後のデータのポインタには 10 進数の 9999 が格納されている。副プログラム SRC は GR0 と GR 1 の各レジスタを表 1 のように設定し，呼び出される。なお，LIST 内のデータ数は 1 個以上あるものとする。また，LIST 内のデータは昇順にアクセスできるように格納されている。

指定したデータが見つかった場合は,そのポインタの値を,見つからない場合は-1 を GR0 に格納して主プログラムに戻る。

GR1→	LIST+0		何も格納されていない
	+1	6	先頭のデータのポインタ
	+2	#0012	←データ
	+3	10	←ポインタ
	+4	#000A	
	+5	12	
	+6	#0007	
	+7	4	
	+8	#00BC	
	+9	9999	←リストの最後
	+10	#001F	
	+11	14	
	+12	#000B	
	+13	2	
	+14	#002C	
	+15	8	

図 1　LIST 番地以降のデータ例

表 1　レジスタの内容

GR0	指定したデータ
GR1	LIST の絶対番地

［プログラム 1］

```
100  SRC      START
110           LD      GR2,1,GR1  ;LIST内の先頭データの添え字をGR2に設定
120  LOOP     ADDL    GR2,GR1    ;ポインタを絶対番地に変換
130           [   (1)   ]        ;指定したデータと比較
140           JZE     FOUND
150           JMI     NOTF
160           [   (2)   ]        ;次のポインタを設定
170           CPL     GR2,=9999
180           JZE     NOTF
190           JUMP    LOOP
200  FOUND    SUBL    GR2,GR1    ;絶対番地をポインタに変換
210           LD      GR0,GR2
220           JUMP    EX
230  NOTF     LAD     GR0,-1
240  EX       RET
              END
```

＜設問１＞　プログラム１中の［　　］に入れるべき適切な字句を解答群から選べ。

(1) の解答群

ア．**CPA GR0,0,GR1**　　イ．**CPA GR0,0,GR2**

ウ．**CPA GR0,1,GR1**　　エ．**CPA GR0,1,GR2**

(2) の解答群

ア．**LD GR0,0,GR2**　　イ．**LD GR0,1,GR2**

ウ．**LD GR2,0,GR2**　　エ．**LD GR2,1,GR2**

＜設問２＞　次のデータの削除に関する問に答えよ。

図１のようなデータ例で，データ#001F を削除した後の状態に関する記述として正しいものを解答群から選べ。ただし，削除後もデータは昇順にアクセスできるようにする。

(3) の解答群

ア．**(LIST+1)**番地の内容が 10 に変わる

イ．**(LIST+3)**番地の内容が 14 に変わる

ウ．**(LIST+8)**番地の内容が 14 に変わる

エ．**(LIST+9)**番地の内容が 18 に変わる

プログラム 2 はプログラム 1 に修正を加え，削除処理ができるようにした。

[プログラム 2]

```
100  SRC      START
110           LD      GR2,1,GR1   ;LIST 内の先頭データの添え字を GR2 に設定
120  LOOP     ADDL    GR2,GR1     ;ポインタを絶対番地に変換
130           (1)                 ;指定したデータと比較
140           JZE     FOUND
150           JMI     NOTF
160           (4)                 ;前のポインタ(絶対番地）を退避
170           (2)                 ;次のポインタを設定
180           CPL     GR2,=9999
190           JZE     NOTF
200           JUMP    LOOP
210  FOUND    CALL    DELETE
220           JUMP    EX
230  NOTF     LAD     GR0,-1
240  EX       RET
250           END
260  ;
270  DELETE   START
280           (5)
290           ST      GR2,1,GR3   ;前データのポインタを再設定
300           RET
310           END
```

＜設問３＞　プログラム 2 中の［　　］に入れるべき適切な字句を解答群から選べ。
ただし，(1)，(2) はプログラム 1 と同じ命令が入るものとする。

(4) の解答群

ア．**LD GR0,GR1**　イ．**LD GR1,GR2**　ウ．**LD GR2,GR3**　エ．**LD GR3,GR2**

(5) の解答群

ア．**LAD GR2,1,GR2**　イ．**LAD GR3,1,GR3**

ウ．**LD GR2,1,GR2**　エ．**LD GR3,1,GR3**

＜メモ欄＞