文部科学省後援

情報システム試験

平成20年度前期 情報検定

＜実施　平成20年9月7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45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20年10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～10 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	12 ページ～15 ページ
・Ｊａｖａの問題	16 ページ～21 ページ
・アセンブラの問題	22 ページ～24 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	25 ページ～32 ページ
・Ｊａｖａの問題	33 ページ～42 ページ
・アセンブラの問題	43 ページ～45 ページ

必須問題

問題１　次のバブルソートに関する記述を読み，設問に答えよ。

［バブルソートの説明］

バブルソートは隣り合う２つの要素を比較し，昇順または降順になるように要素を交換しながら並び替えを行うものである。

配列 DATA[i]（i＝１～ｎ）の要素を昇順に並べ替える場合，次のような手順で行う。

（昇順にソートする手順）

① 要素の先頭から順に隣り合う要素（DATA[i]と DATA[i+1]）を比較し，DATA[i]のほうが大きい場合，要素を入れ替える。

② ①の手順をｉの値を１つずつ増やしながら，ｉの値が n-1 になるまで繰り返す。

③ ①～③の手順中，一度も入れ替えが発生しなかった場合は処理を終了する。

④ ｎの値を１つずつ減らしながら，ｎの値が１になるまで①～③の手順を繰り返す。

（例）

配列 DATA[]の要素数が５個の場合

1	2	3	4	5
3	2	1	5	4

①，②の手順を１回行った後

1	2	3	4	5
2	1	3	4	5

←最後の要素が確定する

①，②の手順を２回行った後

1	2	3	4	5
1	2	3	4	5

←最後から２つまでの要素が確定する

①，②の手順を３回行った後

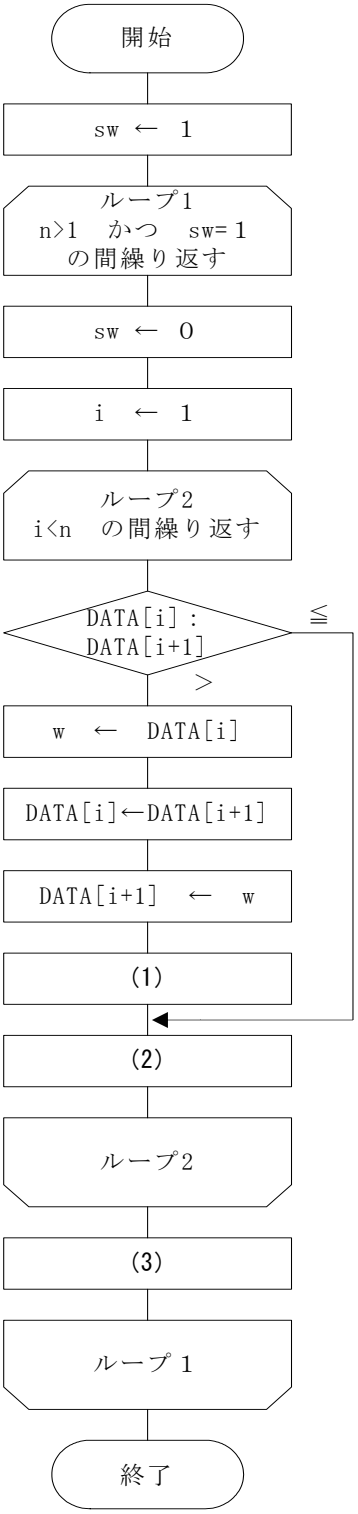
1	2	3	4	5
1	2	3	4	5

←入れ替えが発生しなかったのですべて確定し終了する

＜設問１＞　次のバブルソートの流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。

［流れ図の説明］

要素数ｎ個の配列 DATA をバブルソートで昇順に並び替えるものである。なお，配列の要素位置は１から始まる。



(1) ～ (3) の解答群

ア．i ← i + 1　　イ．i ← i − 1

ウ．sw ← 0　　エ．sw ← 1

オ．n ← n − 1　　カ．n ← n + 1

キ．w ← DATA[i+1]

ク．w ← DATA[n]

＜設問２＞　配列要素が次のように設定されたとき，ソート終了時点での n の値を(4)の解答群から選べ。

配列 DATA　n の初期値は 7

5	1	3	7	6	9	8

(4) の解答群

ア．1　　イ．2　　ウ．3　　エ．4　　オ．5　　カ．6

問題２　商品別集計に関する次の記述を読み，各設問に答えよ。

Ｊ社は全国に 150 店舗をもつ販売会社である。売上情報は POS レジで１件ごとに入力され，図１に示す売上ファイルに１日単位で蓄積される。

支店コード	年月日（YY/MM/DD）	商品コード	売上数量

図１　売上ファイル

Ｊ社では支店ごとの商品別売上数量を累積している。売上マスタファイルの形式は図２のとおりである。売上マスタファイルは第１キーを支店コード，第２キーを商品コードとして昇順にソートされている順編成ファイルである。

支店コード	商品コード	累積売上数量

図２　売上マスタファイル

また，Ｊ社では毎日の商品別売上高を印刷し，次の仕入れに役立てている。商品の単価は図３に示す商品マスタファイルに格納されている。商品マスタファイルは商品コードの昇順にソートされている順編成ファイルである。

商品コード	商品名	単価

図３　商品マスタファイル

［商品別累積売上数の集計と商品別売上高表の印字］

売上ファイルを集計し，売上マスタファイルと突き合わせながら累積売上数量を更新する。商品マスタファイルを参照しながら商品別売上高表を印字する。

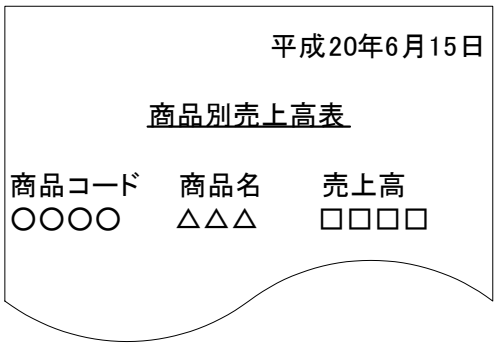


図４　商品別売上高表

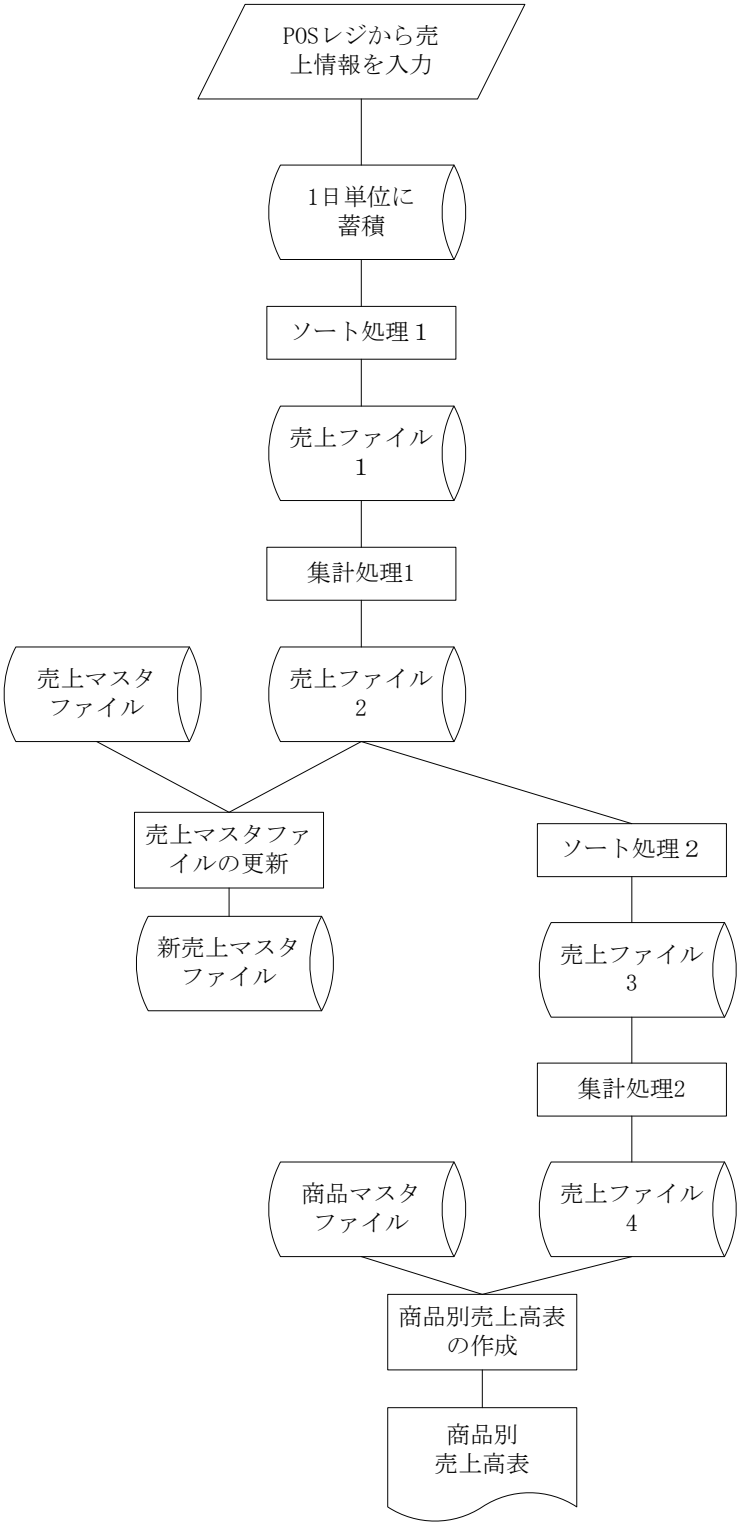


図５　売上マスタファイルの更新と商品別売上高表の作成処理

＜設問１＞　ソート処理１およびソート処理２で行う処理内容を解答群から選べ。ただし，ソート処理１の内容を(1)とし，ソート処理２の内容を(2)とする。

(1)，(2)の解答群

ア．支店コードで昇順にソートする。
イ．商品コードで昇順にソートする。
ウ．売上数量の降順にソートする。
エ．第１キーを支店コード，第２キーを商品コードとして昇順にソートする。
オ．第１キーを売上数量の降順，第２キーを支店コードの昇順にソートする。

＜設問２＞　集計処理１および集計処理２で行う処理内容を解答群から選べ。ただし，集計処理１の内容を(3)とし，集計処理２の内容を(4)とする。

(3)，(4)の解答群

ア．同一商品の売上高を集計する。
イ．同一商品の売上数量を集計する。
ウ．支店別に同一商品の売上高を集計する。
エ．支店別に同一商品の売上数量を集計する。
オ．支店別に総売上高を集計する。

問題３　リストに関する次の記述を読み，設問に答えよ。

次のような双方向のポインタを持つリスト構造のデータがある。Head には先頭の商品コードの要素番号が，Tail には最後尾の商品コードの要素番号が格納されている。Head から次ポインタをたどることによって商品コードの昇順にアクセスができ，Tail から前ポインタをたどることによって商品コードの降順にアクセスができる。

先頭の商品コードの前ポインタと最後尾の商品コードの次ポインタには NULL が設定される。また，データが格納されていない商品コードには NULL が設定されており，ポインタ部は不定である。

Head

7

Tail

6

要素番号	商品コード	次ポインタ	前ポインタ
1	001B	3	7
2	002C	5	9
3	001C	4	1
4	002A	9	3
5	003B	6	2
6	004A	NULL	5
7	001A	1	NULL
8	NULL		
9	002B	2	4
10	NULL		

図１　リスト構造のデータ

＜設問１＞　商品コードの追加・削除に関する次の記述中の［　　］に入れるべき適切な字句を解答群から選べ。解答は重複して選んでもよい。

［追加処理］

図１のリスト構造のデータで，要素番号 8 の位置に商品コード 003A を追加した場合，要素番号 2 の次ポインタと要素番号 5 の前ポインタに［ (1) ］を設定し，要素番号 8 の次ポインタに［ (2) ］，前ポインタには［ (3) ］を設定する。

［削除処理］

図１のリスト構造のデータで，商品コード 001C のデータを削除するには要素番号［ (4) ］の次ポインタを 4 に設定し，要素番号 4 の前ポインタを［ (5) ］に設定する。要素番号 3 の商品コードには NULL を設定する。

(1) ～ (5) の解答群

ア．1　　イ．2　　ウ．3　　エ．4　　オ．5

カ．6　　キ．7　　ク．8　　ケ．9　　コ．10

＜設問２＞　追加処理に関する次の流れ図中の［　　］に入れるべき適切な字句を解答群から選べ。なお，データを格納する領域は十分確保されているものとする。

［流れ図の説明］

リスト構造を２次元配列 LIST[i,k]で表す。まず，LIST 中の空き領域を探索し，その場所に追加データを格納する。次に，商品コードが昇順になるように挿入位置を探索する。

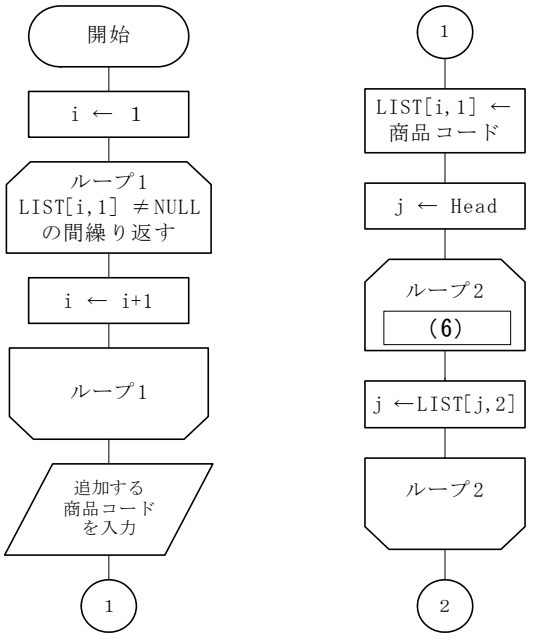
挿入位置が見つかったら,関連するポインタを修正する。LIST 中には１件以上のデータが既に格納されているものとする。

LIST[i,k]は列要素に商品コード，次ポインタ，前ポインタを順に設定する。

k →

i ↓ 要素番号	［商品コード］ 1	［次ポインタ］ 2	［前ポインタ］ 3
1	001B	3	7
2	002C	5	9
3	001C	4	1
4	002A	9	3
5	003B	NULL	2
6	NULL		
7	001A	1	NULL
8	NULL		
9	002B	2	4
10	NULL		

図２　２次元配列 LIST[i,k]



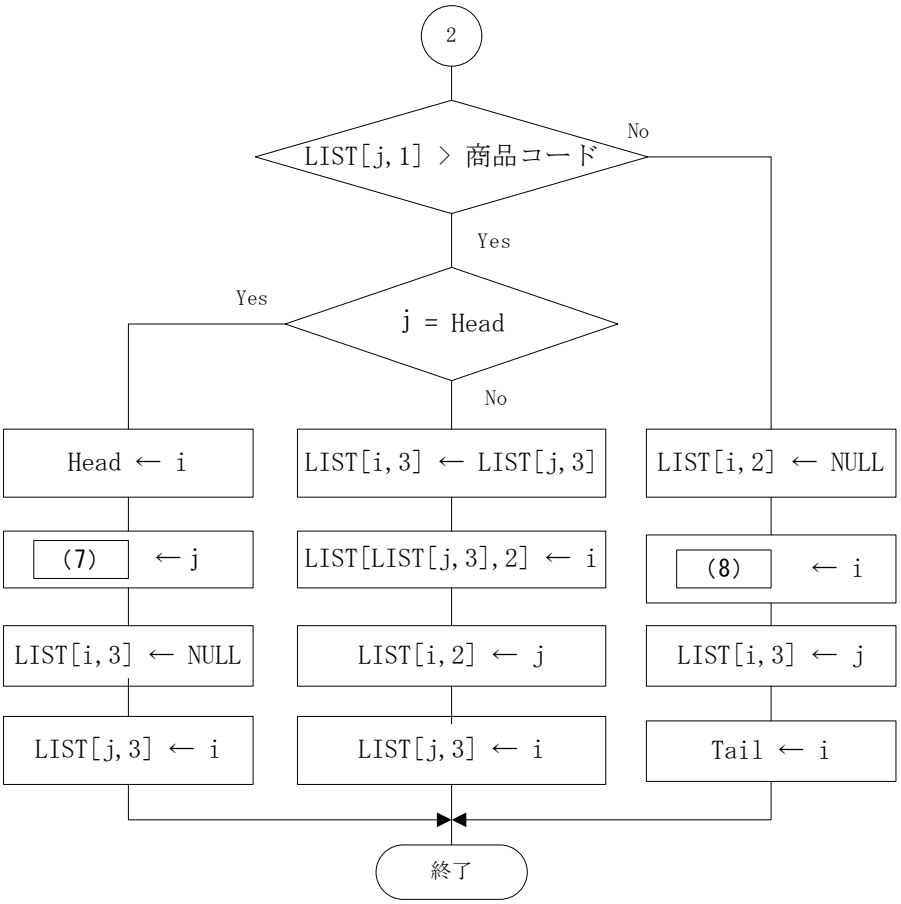


図３　追加処理

(6) の解答群

ア．LIST[j,1]＜商品コード の間繰り返す
イ．LIST[j,1]≠NULL の間繰り返す
ウ．LIST[j,1]＜商品コード　または　LIST[j,2]≠NULL の間繰り返す
エ．LIST[j,1]＜商品コード　かつ　LIST[j,2]≠NULL の間繰り返す

(7)，(8) の解答群

ア．LIST[i,1]　　イ．LIST[i,2]　　ウ．LIST[i,3]
エ．LIST[j,1]　　オ．LIST[j,2]　　カ．LIST[j,3]

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Ｃ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］

自動販売機に投入したお金の種類（金種）と商品の金額から，おつりの金種枚数を計算する **KinshuKeisan** である。

なお，この自動販売機で扱う金種は 1000 円，500 円，100 円，50 円，10 円の５種類である。

また，この自動販売機は，ある金種で必要な枚数が自動販売機内に無い場合でも，その金種より低額の金種でおつりの金額が全てまかなえる場合は，おつりを出す。

（例）100 円が 3 枚必要だったが 100 円が 2 枚しか自動販売機には無い。しかし 50 円が 2 枚以上自動販売機内にあるので，100 円 2 枚と 50 円 2 枚をおつりとして出す。

［金種計算の方法］

おつりの金額を被除数とし，額面の高い金種から順に除数とし除算を行う。その結果，商が除算に使った金種に必要な枚数となり，剰余は次の金種の除数となる。この操作を最後の金種まで繰り返すことでおつりに必要な金種の枚数が計算できる。

（例）おつりが 1230 円だった場合

1230 ÷ 1000 = 1 ･･･ 230　　1000 円は 1 枚

230 ÷ 500 = 0 ･･･ 230　　500 円は 0 枚

230 ÷ 100 = 2 ･･･ 30　　100 円は 2 枚

30 ÷ 50 = 0 ･･･ 30　　50 円は 0 枚

30 ÷ 10 = 3 ･･･ 0　　10 円は 3 枚

なお，必要な枚数が自動販売機内に無い場合は，不足する金額を余りに加える。

［グローバル変数の説明］

このプログラムでは２つのグローバル変数を使う。

１．金種を格納する配列 **KINSHU**

	0	1	2	3	4
KINSHU	1000	500	100	50	10

あらかじめ自動販売機で扱う金種の値を金額の大きい順に格納している。

２．自動販売機内にある金種ごとの枚数を格納する配列 **R_MAISU**

	0	1	2	3	4
R_MAISU	1000 円の枚数	500 円の枚数	100 円の枚数	50 円の枚数	10 円の枚数

自動販売機内に存在する金種ごとの枚数を金種の大きい順に格納している。

［関数の仕様］

int KinshuKeisan(int Kingaku, int EntMaisu[], int RetMaisu[])

引数：**Kingaku**　･･･ 商品の金額

EntMaisu[]･･･ 自動販売機に投入した金種ごとの枚数

0	1	2	3	4
1000 円の枚数	500 円の枚数	100 円の枚数	50 円の枚数	10 円の枚数

RetMaisu[]･･･ おつりとして渡す金種の枚数（結果）

0	1	2	3	4
1000 円の枚数	500 円の枚数	100 円の枚数	50 円の枚数	10 円の枚数

機能：**Kingaku** に格納された金額と **EntMaisu[]**に格納された金種の枚数からおつりを計算し，おつりに必要な金種枚数を計算して **RetMaisu[]**に格納する。

戻り値：購入できるときは 0，購入できないときは-1 を返す。購入できないとき，**RetMaisu** の内容は不定である。

（購入できないときの例）

商品の金額は 300 円だが，投入したのが 100 円 2 枚だった場合や，おつりに必要な 10 円の枚数は 4 枚だが自動販売機内にある 10 円の枚数は 2 枚だった場合など。

［プログラム］

```
#include <stdio.h>
int KINSHU[5] = {1000, 500, 100, 50, 10};
int R_MAISU[5]={2, 8, 10, 5, 14};
int KinshuKeisan(int Kingaku, int EntMaisu[], int RetMaisu[])
{
    int i, j;
    long sogaku, oturi;
    int wr_maisu[5]; /* 販売機の金種ごとの枚数を格納する作業用配列 */
```

```
    /* 投入した金種の総額を計算し配列の初期化をする */
    sogaku = 0;
    for(i=0; i<5; i++) {
        /* 総額の集計 */
        [   (1)   ];
        /* 投入された金額を含めて自動販売機内の金種枚数を集計 */
        wr_maisu[i] = R_MAISU[i] + EntMaisu[i];
        /* 結果を格納する領域の初期化 */
        RetMaisu[i] = 0;
    }
    /* 投入金額が不足していればエラーとして戻る */
    if ([   (2)   ]) return(-1);
    /* おつりの計算 */
    oturi = sogaku - Kingaku;
    /* おつりが無ければ戻る */
    if (oturi == 0) {
        for(i=0; i<5; i++) R_MAISU[i] = wr_maisu[i];
        return (0);
    }
    /* おつりの金種を配列 RetMaisu に求める */
    for(i=0; i<5; i++) {
        RetMaisu[i] = oturi / KINSHU[i];
        oturi = oturi - RetMaisu[i] * KINSHU[i];
        if ([   (3)   ]) {
            /* 必要な金種枚数が無い場合 */
            oturi += KINSHU[i] * (RetMaisu[i] - wr_maisu[i]);
            RetMaisu[i] = wr_maisu[i];
            wr_maisu[i] = 0;
        } else
            /* 必要な金種枚数がある場合 */
            [   (4)   ];
    }
    if ([   (5)   ])
        return(-1);
    else {
        /* 販売機内の金種ごとの枚数を設定 */
        for (i=0; i<5; i++) R_MAISU[i] = wr_maisu[i];
        return(0);
    }
}
```

＜設問＞　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. `sogaku += EntMaisu[i] * 10`

イ. `sogaku += Kingaku - EntMaisu[i]`

ウ. `sogaku += EntMaisu[i] + RetMaisu[i]`

エ. `sogaku += EntMaisu[i] * KINSHU[i]`

(2) の解答群

ア. `sogaku > Kingaku`　　イ. `sogaku < Kingaku`

ウ. `sogaku >= Kingaku`　　エ. `sogaku <= Kingaku`

(3) の解答群

ア. `RetMaisu[i] > wr_maisu[i]`

イ. `RetMaisu[i] < wr_maisu[i]`

ウ. `RetMaisu[i] >= wr_maisu[i]`

エ. `RetMaisu[i] <= wr_maisu[i]`

(4) の解答群

ア. `wr_maisu[i] = 0`

イ. `wr_maisu[i] = RetMaisu[i]`

ウ. `wr_maisu[i] -= RetMaisu[i]`

エ. `wr_maisu[i] += RetMaisu[i]`

(5) の解答群

ア. `oturi > 0`　　イ. `oturi < 0`

ウ. `oturi >= 0`　　エ. `oturi <= 0`

選択問題Ａ　Javaの問題

次のJava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］

コンピュータ販売業Ｊ社の在庫管理システムでは，販売するコンピュータを登録して，顧客が選択する２つのオプションの組合せによって割引を行い，販売金額を提示する。

選択されたオプションと割引の関係を表１に示す。

表１　オプションと割引の関係表

		デスクトップ型コンピュータ		ノート型コンピュータ	
		割引率	値引額	割引率	値引額
オプション１	適用	10%	20,000 円	15%	30,000 円
オプション２	適用				
オプション１	適用	10%	－	15%	－
オプション２	非適用				
オプション１	非適用	－	20,000 円	－	30,000 円
オプション２	適用				
オプション１	非適用	－	－	－	－
オプション２	非適用				

・販売金額は，販売金額＝元値×(１－割引率)－値引額で求める。(小数点以下切捨て)

［クラスの仕様］

DiscountPara クラス

説明	割引に対する情報を扱うクラス
フィールド	**rate**：割引率，**amount**：割引金額
コンストラクタ	引数で渡された各値をフィールドにセットする。
メソッド	**getRate**　：割引率を返す
	getAmount　：値引額を返す

Comp クラス

説明	デスクトップ型とノート型のコンピュータに対する共通の情報を扱う抽象クラス
フィールド	**cpuName**：CPU の名前，**memory**：メモリの容量，**hdd**：ハードディスクの容量，**price**：販売金額
コンストラクタ	引数で渡された各値をフィールドにセットする。
メソッド	**toString**　：フィールドの内容を加工して文字列で返す
	getPrice　：販売金額を返す
	setPrice　：販売金額を **price** に設定する

Desktop クラス

説明	デスクトップ型コンピュータに対する情報を扱うクラス
コンストラクタ	引数で渡された各値をフィールドにセットする
メソッド	**discount**　：引数である 2 つのオプションの有無によって割引率と値引額を求めて **DiscountPara** クラスのインスタンスを生成する

Note クラス

説明	ノート型コンピュータに対する情報を扱うクラス
フィールド	**size**：コンピュータの大きさ
コンストラクタ	引数で渡された各値をフィールドにセットする。
メソッド	**discount**　：引数である 2 つのオプションの有無によって割引率と値引額を求めて **DiscountPara** クラスのインスタンスを生成する

PriceUtil クラス

説明	値引処理を行うクラス
メソッド	**discountPrice**：割引後の販売金額を求めてその値を設定する

クラス **Test** はテスト用のメインプログラムである。実行結果を図 1 に示す。

```
＊＊＊定価＊＊＊
P4-3.2 : 512 : 120 : 64000
C2D-2500 : 1024 : 300 : 89000
PM-1.4 : 512 : 80 : 130000 : B5
Ce-3.0 : 256 : 40 : 118000 : A4
＊割引後＊(Option1=true,Option2=true)
P4-3.2 : 512 : 120 : 37600
C2D-2500 : 1024 : 300 : 60100
PM-1.4 : 512 : 80 : 80500 : B5
Ce-3.0 : 256 : 40 : 70300 : A4
```

図 1　実行結果

[プログラム]

```
class DiscountPara{
    private double rate;
    private int amount;
    DiscountPara(double rate,int amount){
        this.rate=rate;
        this.amount=amount;
```

```
    }
    double getRate(){
        return rate;
    }
    int getAmount(){
        return amount;
    }
}

abstract class Comp{
    private String cpuName;
    private int memory;
    private int hdd;
    private int price;
    Comp(String cpuName,int memory,int hdd,int price){
        this.cpuName = cpuName;
        this.memory = memory;
        this.hdd = hdd;
        this.price = price;
    }
    public String toString(){
        return cpuName + " : " + memory + " : "
            + hdd + " : " + " : " + price;
    }
    int getPrice(){
        return price;
    }
    void setPrice(int price){
        this.price=price;
    }
    [   (1)   ];   //抽象メソッド discount の宣言
}

class Desktop extends Comp{
    Desktop(String cpuName,int memory,int hdd,int price){
        super(cpuName,memory,hdd,price);
    }
    DiscountPara discount(boolean option1,boolean option2){
        if(option1)
            if(option2)
```

```
                return new DiscountPara(0.1,20000);
            else
                return new DiscountPara(0.1,0);
        else
            if(option2)
                return new DiscountPara(0.0,20000);
            else
                return new DiscountPara(0.0,0);
    }
}

class Note extends Comp{
    String size;
    Note(String cpuName,int memory,int hdd,String size,int price){
        super(cpuName,memory,hdd,price);
        this.size = size;
    }
    public String toString(){
        return [      (2)      ];
    }
    DiscountPara discount(boolean option1,boolean option2){
        if(option1)
            if(option2)
                return new DiscountPara(0.15,30000);
            else
                return new DiscountPara(0.15,0);
        else
            if(option2)
                return new DiscountPara(0.0,30000);
            else
                return new DiscountPara(0.0,0);
    }
}

class PriceUtil{
    void discountPrice(Comp comp,boolean option1,boolean option2){
        [      (3)      ];                    //適用される割引を求める
        comp.setPrice([      (4)      ]);     //販売金額の計算
    }
}
```

```
class Test3{
    public static void main(String[] args){
        Comp[] comp = {
            new Desktop("P4-3.2",512,120,64000),
            new Desktop("C2D-2500",1024,300,89000),
            new Note("PM-1.4",512,80,"B5",130000),
            new Note("Ce-3.0",256,40,"A4",118000),
        };
        boolean op1=true,op2=true;          //割引オプションの設定
        System.out.println("＊＊＊定価＊＊＊");
        for(int i=0;i<comp.length;i++)
            System.out.println(comp[i]);
        PriceUtil di=new PriceUtil();
        for(int i=0;i<comp.length;i++)
            di.discountPrice(comp[i],op1,op2);
        System.out.println
                  ("＊割引後＊(Option1="+op1+",Option2="+op2+")");
        for(int i=0;i<comp.length;i++)
            System.out.println(comp[i]);
      }
}
```

＜設問＞　プログラム中の［　　］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. `abstract void discount()`

イ. `abstract DiscountPara discount(boolean option1,boolean option2)`

ウ. `abstract DiscountPara discount()`

エ. `abstract void discount(boolean option1,boolean option2)`

(2) の解答群

ア. `cpuName + " : " + memory + " : " + hdd  + " : " + price+ " : "`

イ. `cpuName + " : " + memory + " : " + hdd  + " : " + price+ " : " + size`

ウ. `super.toString()`

エ. `super.toString() + " : " + size`

(3) の解答群

ア. `DiscountPara d = new DiscountPara()`

イ. `comp.discount(option1,option2)`

ウ. `comp.discount()`

エ. `DiscountPara d=comp.discount(option1,option2)`

(4) の解答群

ア. `comp.getPrice()*(1-d.getRate())-d.getAmount()`

イ. `(int)comp.getPrice()*(1-d.getRate())-d.getAmount()`

ウ. `(int)(comp.getPrice()*(1-d.getRate()))-d.getAmount()`

エ. `comp.getPrice()*(int)(1-d.getRate())-d.getAmount()`

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］

副プログラム SEARCH は 16 ビットのデータの中から，指定された部分ビット列を探索するプログラムである。

１．主プログラムは，パラメタ領域の先頭番地を GR1 に設定して，SEARCH を呼ぶ。
パラメタ形式は次のとおりである。

(GR1+0)	16 ビットのデータ	
	部分ビット列	先頭から設定
	部分ビット列の長さ	ビット数

２．部分ビット列が見つかった場合はその先頭のビット位置を GR0 に格納し，見つからない場合は -1 を GR0 に格納し主プログラムに戻る。

(例)　パラメタが次のように設定された場合

(GR1+0)	#AB07	
	#B687	先頭から設定
	4	ビット数

(16 ビットのデータ)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	1	1	0	0	0	0	0	1	1	1

(部分ビット列)

1	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1

部分ビット列の長さを４とすると，網掛けのビットが一致するので，GR0 に 4 を設定して主プログラムに戻る。

[プログラム]

```
100  SEARCH  START
110          RPUSH
120          LD    GR0,0,GR1     ;16 ビットのデータ
130          LD    GR2,1,GR1
140          LD    GR3,2,GR1     ;部分ビット列の長さ
150          LD    GR4,=#8000
160          (1)                 ;マスクビットの生成
170          AND   GR2,GR4       ;部分ビット列の取り出し
180          LAD   GR5,16
190          SUBA  GR5,GR3
200          LD    GR6,GR5
210  LOOP    LD    GR7,GR0
220          AND   GR7,GR4
230          CPL   GR7,GR2
240          JZE   FND
250          (2)
260          SUBA  GR5,=1
270          JPL   LOOP
280          (3)
290          JUMP  NFND
300  FND     SUBA  GR6,GR5
310          LD    GR0,GR6
320          JUMP  EXIT
330  NFND    LD    GR0,=-1
340  EXIT    RPOP
350          RET
360          END
```

＜設問１＞　プログラム中の［　　］に入れるべき適切な字句を解答群から選べ。

(1)，(2) の解答群

ア．`SRL  GR4,0,GR3`　　イ．`SRL  GR4,-1,GR3`　　ウ．`SRA  GR4,0,GR3`

エ．`SRA  GR4,-1,GR3`　　オ．`SLL  GR0,1`　　カ．`SRL  GR0,1`

(3) の解答群

ア．`JOV  LOOP`　　イ．`JMI  LOOP`　　ウ．`JZE  LOOP`

エ．`JNZ  LOOP`

＜設問２＞　パラメタが下記のように設定されたとき，GR0 に設定される値を(4)の解答群から選べ。

パラメタ

(GR1+0)	#0ADC	
	#B73C	先頭から設定
	5	ビット数

(4) の解答群

ア．-1　　イ．0　　ウ．3　　エ．4　　オ．5　　カ．6

選択問題Ｂ　C言語の問題

次のＣプログラム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１参照）

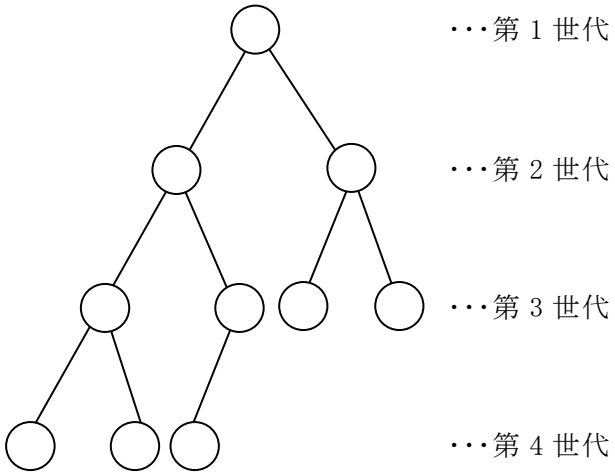


図１　完全２分木の例

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）または，（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２参照）

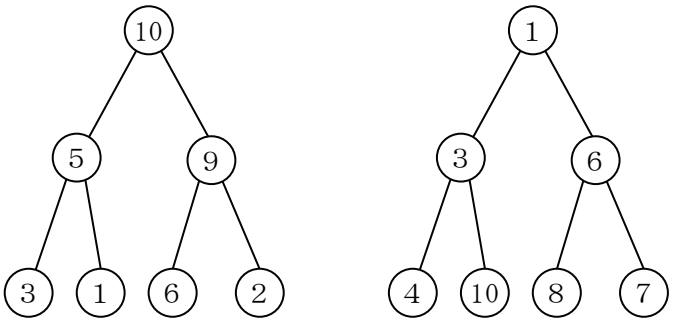


図２　ヒープ構造の例

＜設問１＞　次の完全２分木の配列表現に関する記述中の［　　］に入れるべき最も適切な字句を解答群から選べ。なお，除算はすべて小数点以下切り捨てとする。また，配列の先頭の添字を１とする。

木は，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，ポインタを使用せず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，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。

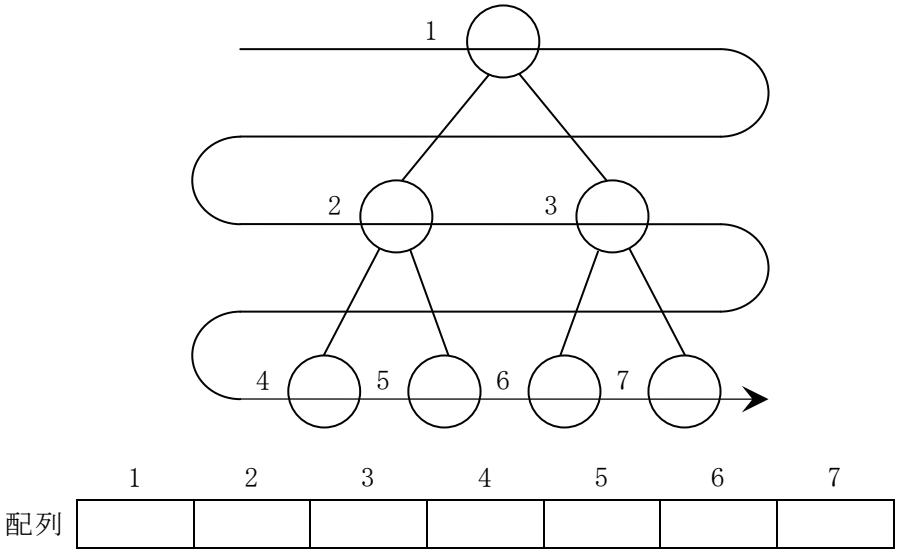


図３　完全２分木と配列との対応関係

ルートからみて，左の子となるノードの配列要素の添字は２であり，右の子となるノードの配列要素の添字は３である。一般的に添字ｉのノードの左の子となるノードの配列要素の添字は［　(1)　］，右の子となるノードの配列要素の添字は［　(2)　］と表現できる。

同様に，添字 ｉ のノードの親となるノードの配列要素の添字は［　(3)　］と表現でき，配列を使った完全２分木のデータ処理をプログラムで簡単に記述することが可能になる。

(1) ～ (3) の解答群

ア．(ｉ － 1) × 2　　イ．(ｉ － 1) × 2 ＋ 1
ウ．ｉ × 2　　エ．ｉ × 2 ＋ 1
オ．(ｉ － 1) ÷ 2　　カ．(ｉ － 1) ÷ 2 ＋ 1
キ．ｉ ÷ 2　　ク．ｉ ÷ 2 ＋ 1

［ヒープ構造の作成に関する説明］

ヒープ構造は完全２分木であるので，配列を使って表現することができる。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。

・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。ルートに該当する先頭の配列要素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配列要素はヒープ構造であると考えられる。残りの（Ｎ－1）個の配列要素はヒープ構造ではない。

・次に，２番目の配列要素がヒープ構造になるように自分の親である先頭の配列要素と比較し，親よりも大きければ先頭と２番目の配列要素の入れ替えを行い，新しいヒープ構造を作る。（走査 1）

・以下，ヒープ構造になっていない３番目からＮ番目までの配列要素について同様の処理を，比較する親がなくなるか，または，親の方が大きくなるまで繰り返す。（走査 2～走査（Ｎ－1））

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。

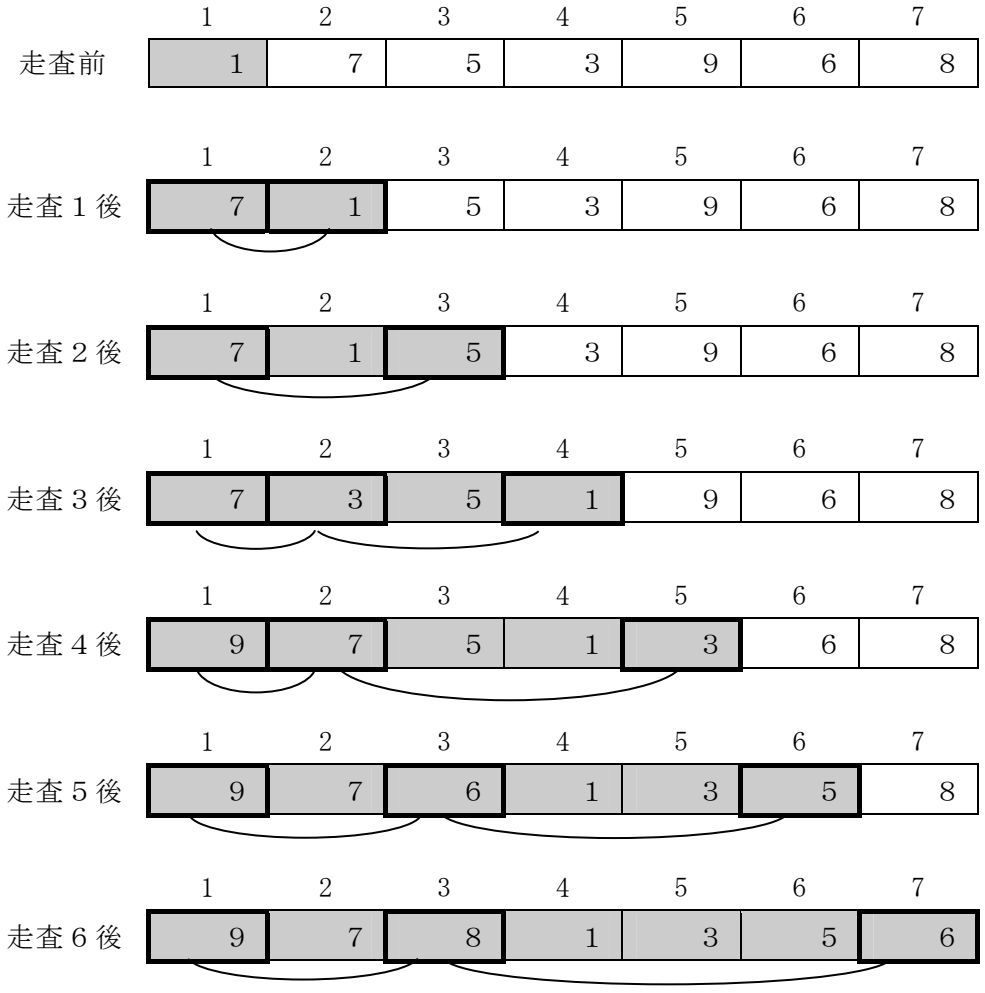


図４　ヒープ構造の作成

＜設問２＞　図５の配列データがヒープ構造になった後の配列の内容として適切なものを(4)の解答群から選べ。なお，配列の添字の先頭は１とする。

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

図５

(4) の解答群

ア.

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

イ.

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

ウ.

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

エ.

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

［ヒープソートの考え方］

N 個のデータを配列に入力し，ヒープ構造を作る。なお，配列の添字の先頭は 1 とする。また，ここでのヒープ構造は，（親の値）≧（子の値）とする。（ヒープ構造作成）

配列にヒープ構造を作ると，ルートとなる先頭の配列要素に一番大きな値が入る。そこで，先頭の配列要素と末尾（ｘ番目）の配列要素を入れ替えると，一番大きな値が末尾の配列要素に格納され，この部分はソート済みになる。

残りの（N－1）個のデータは未整列の状態であり，かつ入れ替えによりヒープ構造でなくなったため，以下の①と②の動作を未整列の配列要素がなくなるまで繰り返す。

①配列の未整列部分についてヒープ構造を作り直す（ヒープ構造再編成）

②先頭の配列要素と未整列部分の末尾の配列要素を入れ替える（データ交換）

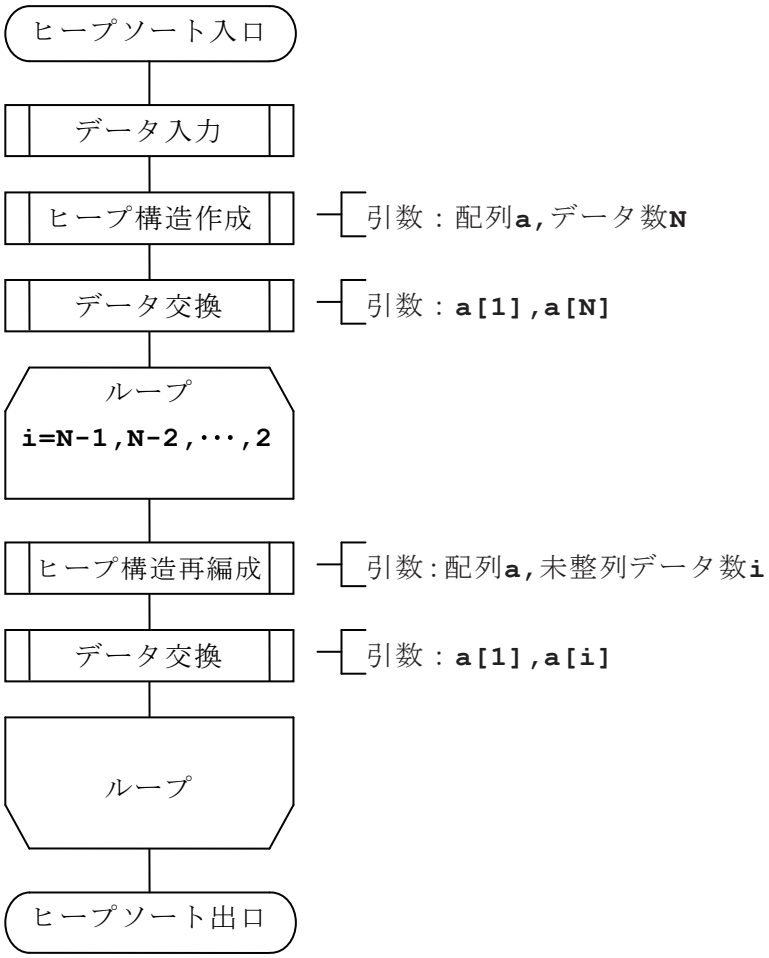


図６　ヒープソートのアルゴリズム

＜設問３＞　次のヒープソートを行うＣ言語プログラム中の［　　］に入るべき適切な字句を解答群から選べ。

［関数仕様］

void HeapMake(int a[], int n)

引　数：**a[]** ･･･　配列の先頭アドレス
　　n　･･･　ヒープ構造対象のデータ数

機　能：指定された配列のデータにより，ヒープ構造を作成する。

戻り値：なし

void HeapReset(int a[], int i)

引　数：**a[]** ･･･　配列の先頭アドレス
　　i　･･･　ヒープ構造対象のデータ数

機　能：指定された配列のデータにより，ヒープ構造を再編成する。

戻り値：なし

void HeapSwap(int *a, int *b)

引　数：***a**　･･･　配列要素のアドレス
　　***b**　･･･　配列要素のアドレス

機　能：指定された配列要素同士を入れ替える。

戻り値：なし

［プログラム］

```
#include <stdio.h>
void HeapMake(int [], int);
void HeapReset(int [], int);
void HeapSwap(int *, int *);
void main(void)
{
    int a[] = {0,1,5,7,3,9,6,8}; /* a[0]はダミーで，データ数は７個 */
    int i;
    HeapMake(a, 7);
    HeapSwap(&a[1], &a[7]);
    for(      (5)      ) {
        HeapReset(a, i);
        HeapSwap(&a[1], &a[i]);
    }
}
```

```
void HeapMake(int a[], int n)
{
    int i, j, k;
    for(i=2; i<=n; i++) {
        j = i;
        [ (6) ];          /* 親の位置を計算 */
        while(k > 0) {
            if (a[k] >= a[j]) break;
            HeapSwap(&a[k], &a[j]);
            j = k;
            [ (6) ];      /* 親の位置を計算 */
        }
    }
}
void HeapReset(int a[], int i)
{
    int j, left, right, k;
    j = 1;       /* 親の位置 */
    left = 2;   /* 子の位置（要素位置の小さい方） */
    right = 3;  /* 子の位置（要素位置の大きい方） */
    k = left;
    while( left <= i ) {
        /* 2つの子のうち大きい方の要素位置をkに求める */
        if (left != i)
            if (a[left] < a[right]) [ (7) ];
        /* 親の値が小さければ交換する */
        if ( [ (8) ] ) HeapSwap(&a[j], &a[k]);
        j = k;
        [ (9) ];
        [ (10) ];
        k = left;
    }
}
void HeapSwap(int *a, int *b)
{
    int w;
    w = *a;
    *a = *b;
    *b = w;
}
```

(5) の解答群

ア. `i = 6; i > 1; i--`　　イ. `i = 7; i > 1; i--`

ウ. `i = 1; i < 6; i++`　　エ. `i = 1; i < 7; i++`

(6) の解答群

ア. `k = j`　　イ. `k = 2 * j`

ウ. `k = j / 2`　　エ. `k = 2 * j + 1`

(7) の解答群

ア. `k = left`　　イ. `k = right`

ウ. `k = left + right`　　エ. `k = (left + right) / 2`

(8) の解答群

ア. `a[j] < a[k]`　　イ. `a[j] > a[k]`

ウ. `a[left] < a[right]`　　エ. `a[left] > a[right]`

(9), (10) の解答群

ア. `left = j`　　イ. `left = 2 * j`

ウ. `right = left`　　エ. `right = left + 1`

選択問題Ｂ　Javaの問題

次のJavaプログラムの説明を読んで，設問に答えよ。

［プログラム１の説明］

リスト１～５に示すプログラム１は，ファイル及びフォルダの階層構造を作成し，その構造と各ファイル及びフォルダの容量を表示するプログラムである。なお実行結果を図１に示す。

```
[root] [60]
[fola] [20]
[folc] [0]
fily (20)
[folb] [30]
filz (30)
filx (10)
```

図１　実行結果

プログラム１および２中で使用している **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** パッケージに存在する）

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。）

プログラム１および２中では，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］

Item クラス

説明	フォルダとファイルの情報に対する共通の情報を扱う抽象クラス
フィールド	**name**：フォルダ又はファイルの名前
コンストラクタ	名前をフィールドに設定する

FileInfo クラス

説明			ファイルの情報を扱うクラス
フィールド			**size**：ファイルの容量
コンストラクタ			名前と容量をフィールドに設定する
メソッド	**getSize**	：	容量を返す
	showInfo	：	名前と容量を表示する

FolderInfo クラス

説明			フォルダの情報を扱うクラス
フィールド			**item**：フォルダやファイルのインスタンスを扱う **ArrayList**
コンストラクタ			名前をフィールドに設定する
メソッド	**getSize**	：	容量を返す
	showInfo	：	名前と容量を表示する
	add	：	ファイルやフォルダを item に追加する
	nameCheck	：	引数で受け取ったファイル又はフォルダのインスタンスと，同一種類で同一名のインスタンスが **item** にあるかを確認する

［リスト１］

```
abstract public class Item{
    protected String name;
    public String getName(){
        return name;
    }
    abstract public long getSize();
    abstract public void showInfo();
}
```

［リスト２］

```
public class FileInfo extends Item{
    private long size;
    public FileInfo(String name,long size){
        this.name = name;
        this.size = size;
    }
    public long getSize(){
        return size;
    }
    public void showInfo(){
        System.out.printf("%s (%d)¥n",getName(),getSize());
    }
}
```

［リスト３］

```
import java.util.ArrayList;
public class FolderInfo extends Item{
    ArrayList<Item> item = new ArrayList<Item>();
    public FolderInfo(String name){
        this.name = name;
    }
    public void add(Item obj){
        String type;
        if(nameCheck(obj))
            (1)          ;       //itemにインスタンスの追加を行う
        else{
            //FolderInfoのインスタンスであればtrue
            if(obj instanceof FolderInfo)
                type = "フォルダ";
```

```
            else
                type = "ファイル";
            System.err.printf("%s は同じフォルダ内に同名の%s があるため追
                加できません。¥n",obj.getName(),type);
        }
    }
    public long getSize(){
        long size = 0;
        //item 内のインスタンスに対する size の合計を求める
        for(Item i:item)
            [  (2)  ];
        return size;
    }
    public boolean nameCheck(Item obj){
        boolean f = true;
        for(Item i:item){
            if(i.getName().equals(obj.getName())){
                if((i instanceof FileInfo && obj instanceof FileInfo) ||
                (i instanceof FolderInfo && obj instanceof FolderInfo))
                    //同じ名前で同じクラスのインスタンスであれば false
                    f = false;
            }
        }
        return f;
    }
    public void showInfo(){
        //名前と容量の表示
        System.out.printf("[%s] [%d]¥n",getName(),getSize());
        //item 内にあるインスタンスの名前と容量の表示
        for(Item i:item)
            [  (3)  ];
    }
}
```

［リスト４］

```
public class Test{
    public static void main(String[]args){
        FolderInfo d1 = new FolderInfo("root");
        FolderInfo d2 = new FolderInfo("fola");
        FolderInfo d3 = new FolderInfo("folb");
```

```
        FolderInfo d4 = new FolderInfo("folc");
        FileInfo f1 = new FileInfo("filx",10);
        FileInfo f2 = new FileInfo("fily",20);
        FileInfo f3 = new FileInfo("filz",30);
        d1.add(d2);
        d1.add(d3);
        d2.add(d4);
        d1.add(f1);
        d2.add(f2);
        d3.add(f3);
        d1.showInfo();
    }
}
```

＜設問１＞　プログラム１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．**showInfo()**　　イ．**i.showInfo()**　　ウ．**item.showInfo()**
エ．**add(obj)**　　オ．**obj.add(item)**　　カ．**item.add(obj)**
キ．**size = getSize()**　　ク．**size += getSize()**
ケ．**size = i.getSize()**　　コ．**size += i.getSize()**

＜設問２＞　プログラム１のリスト４をリスト５に変更して実行した場合の標準出力の表示として適切なものを(4)の解答群から選べ。

［リスト５］

```
public class Test{
    public static void main(String[]args){
        FolderInfo d1 = new FolderInfo("root");
        FolderInfo d2 = new FolderInfo("fola");
        FolderInfo d3 = new FolderInfo("folb");
        FolderInfo d4 = new FolderInfo("fola");
        FileInfo f1 = new FileInfo("filx",20);
        FileInfo f2 = new FileInfo("fily",30);
        FileInfo f3 = new FileInfo("filx",20);
        FileInfo f4 = new FileInfo("fily",50);
        d1.add(d2);
        d1.add(d3);
        d1.add(d4);
        d2.add(f1);
```

```
        d2.add(f2);
        d3.add(f3);
        d4.add(f4);
        d1.showInfo();
    }
}
```

(4) の解答群

ア.

```
[root] [120]
[fola] [50]
filx (20)
fily (30)
[folb] [20]
filx (20)
[fola] [50]
filx (50)
```

イ.

```
[root] [120]
[fola] [50]
filx (20)
fily (30)
[folb] [20]
filx (20)
```

ウ.

```
fola は同じフォルダ内に同名のフォルダがあるため追加できません。
[root] [70]
[fola] [50]
filx (20)
fily (30)
[folb] [20]
filx (20)
```

エ.

```
fola は同じフォルダ内に同名のフォルダがあるため追加できません。
```

[プログラム２の説明]

リスト６～９に示すプログラム２は,プログラム１にファイル及びフォルダのパス(経路）を表示する機能を追加したプログラムである。なお，実行結果を図２に示す。

```
[root] [60]     /
[fola] [20]     /fola/
[folc] [0]      /fola/folc/
fily (20)       /fola/fily
[folb] [30]     /folb/
filz (30)       /folb/filz
filx (10)       /filx
```

図２　実行結果２

［リスト６］

```
abstract class Item{
    protected String name;
    protected String pass = "/";            //パス
    public String getName(){
        return name;
    }
    public String getPass(){                //パスを返す
        return pass;
    }
    abstract public void setPass(String s);
    abstract public long getSize();
    abstract public void showInfo();
}
```

［リスト７］

```
class FileInfo extends Item{
    private long size;
    public FileInfo(String name,long size){
        this.name = name;
        this.size = size;
    }
    public long getSize(){
        return size;
    }
    public void showInfo(){
        System.out.printf
            ("%s (%d) ¥t%s¥n",getName(),getSize(),getPass());
    }
    public void setPass(String s){
        //引数で受け取ったフォルダのパスに自分の名前を繋げてパスに設定
        (5) ;
    }
}
```

［リスト８］

```
import java.util.*;
class FolderInfo extends Item{
    ArrayList<Item> item = new ArrayList<Item>();
    public FolderInfo(String name){
```

```
        this.name = name;
    }
    public void add(Item obj){
        String type;
        obj.setPass(getPass());        //自分のパスを obj のパスに繋げる
        if(nameCheck(obj))
            (1)          ;             //item にインスタンスの追加を行う
        else{
            //FolderInfo のインスタンスであれば true
            if(obj instanceof FolderInfo)
                type = "フォルダ";
            else
                type = "ファイル";
            System.err.printf("%s は同じフォルダ内に同名の%s があるため
                追加できません。¥n",obj.getPass(),type);
        }
    }
    public long getSize(){
        long size = 0;
        //item 内のインスタンスに対する size の合計を求める
        for(Item i:item)
            (2)          ;
        return size;
    }
    public boolean nameCheck(Item obj){
        boolean f=true;
        for(Item i:item){
            if(i.getName().equals(obj.getName())){
            if((i instanceof FileInfo && obj instanceof FileInfo)||
            (i instanceof FolderInfo && obj instanceof FolderInfo))
                //同じ名前で同じクラスのインスタンスであれば false
                f = false;
        }
    }
    return f;
}
public void showInfo(){
    //名前と容量とパスの表示
    System.out.printf("[%s] [%d] ¥t%s¥n"
    ,getName(),getSize(),getPass());
```

```
            //item 内にあるインスタンスの名前と容量とパスの表示
            for(Item i:item)
                 (3)  ;
    }
    public void setPass(String s){
            //受け取ったパスに自分の名前と/を繋げてパスに設定
             (6)  ;
             for(Item i:item)
                 //item内にあるインスタンスのsetPassに自分のパスを引数として渡す
                 (7)  ;
    }
}
```

［リスト９］

```
public class Test{
    public static void main(String[]args){
        FolderInfo d1 = new FolderInfo("root");
        FolderInfo d2 = new FolderInfo("fola");
        FolderInfo d3 = new FolderInfo("folb");
        FolderInfo d4 = new FolderInfo("folc");
        FileInfo f1 = new FileInfo("filx",10);
        FileInfo f2 = new FileInfo("fily",20);
        FileInfo f3 = new FileInfo("filz",30);
        d1.add(d2);
        d1.add(d3);
        d2.add(d4);
        d1.add(f1);
        d2.add(f2);
        d3.add(f3);
        d1.showInfo();
    }
}
```

＜設問３＞　プログラム２の［　　］にいれるべき適切な字句を解答群から選べ。

(5)，(6) の解答群

ア．pass = pass+s　　イ．pass = s+pass;　　ウ．pass = name+s

エ．pass = s+name　　オ．pass = s+name+"/"

カ．pass = "/"+name+s　　キ．pass = pass+s+"/"

ク．pass = "/"+s+pass

(7) の解答群

ア. **setPass(pass)**

イ. **setPass(s)**

ウ. **i.setPass(pass)**

エ. **i.setPass(s)**

選択問題Ｂ　アセンブラの問題

次のＣＡＳＬⅡプログラムに関する説明を読み，設問に答えよ。

［プログラムの説明］

副プログラム MULTI は乗算を行うプログラムである。

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

パラメタ

(GR1+0)	被乗数
	乗数

2. 乗算の結果は GR0 に設定するが,処理中にオーバーフローが発生した場合はエラーメッセージを表示し，GR0 は呼び出された時と同じ状態で主プログラムに戻る。
3. 乗算のアルゴリズムは次のとおりとする。
   ① 結果を格納する作業領域を 0 で初期化する。
   ② 乗数を右へ 1 ビットシフトする。
   ③ ②で，あふれたビットが 1 ならば被乗数を作業領域に加算し，0 なら加算しない。
   ④ 被乗数を左へ 1 ビットシフトする。
   ⑤ 乗数が 0 でない間②～④を繰り返す。

(計算例)

被乗数が#000A，乗数が#0006 の場合

```
      0000 0000 0000 1010   被乗数
  ×)  0000 0000 0000 0110   乗数
00    0000 0000 0001 0100
+)000 0000 0000 0010 1000
000   0000 0000 0011 1100   結果
```

4. 乗数，被乗数ともに正の整数とする。

[プログラム]

```
100    MULTI    START
110             ST   GR0,WORK
120             LAD  GR0,0
130             LD   GR2,0,GR1
140             LD   GR3,1,GR1
150    LOOP     SRL  GR3,1        ;乗数の最下位ビットを調べる
160             [   (1)   ]
170             JUMP NEXT
180    KASAN    ADDA GR0,GR2
190             JOV  ERR
200    NEXT     SLA  GR2,1
210             [   (2)   ]       ; オーバーフローしたか？
220             LD   GR3,GR3
230             [   (3)   ]       ; 終了判定
240             JUMP EXIT
250    ERR      LD   GR0,WORK
260             OUT  OBUF,OLEN
270    EXIT     RET
280    WORK     DS   1
290    OBUF     DC   'OVERFLOW'
300    OLEN     DC   8
310             END
```

＜設問１＞　プログラム中の［　　］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. **JZE LOOP**	イ. **JNZ LOOP**	ウ. **JMI LOOP**
エ. **JPL KASAN**	オ. **JMI KASAN**	カ. **JOV KASAN**
キ. **JNZ ERR**	ク. **JPL ERR**	ケ. **JOV ERR**

＜設問２＞ パラメタが次のように設定されて副プログラム MULTI が呼び出されたとき，180 行の命令は何回実行されるか。(4)の解答群から選べ。

(GR1+0)	#3FFF
	#000F

(4) の解答群

ア．0 回　　イ．1 回　　ウ．2 回　　エ．3 回　　オ．4 回

＜設問３＞　被乗数，乗数が負数のときも対応できるようにプログラムを次のように変更した。プログラム中の［　　］に入れるべき適切な命令を (5)，(6)の解答群から選べ。

[130 行と 140 行を次のように変更]

```
        LAD     GR4,0
        LD      GR2,0,GR1
        JPL     L1
        LAD     GR2,0
        SUBA    GR2,0,GR1
        XOR     GR4,=#0001
L1      LD      GR3,1,GR1
        JPL     LOOP
        LAD     GR3,0
        SUBA    GR3,1,GR1
        XOR     GR4,=#0001
```

乗数と被乗数の符号を調べ、同符号なら **0** を，異符号なら **1** を **GR4** に設定する。

[230 行と 240 行の間に次の命令を挿入]

```
        LD      GR4,GR4
        (5)
        XOR     GR0,=#FFFF
        (6)
```

GR4 の内容により,**GR0** の符号を反転する。

(5)，(6) の解答群

ア．**JMI　EXIT**　　イ．**JZE　EXIT**　　ウ．**JNZ　EXIT**

エ．**SUBA　GR0,0,GR1**　　オ．**SUBA　GR0,1,GR1**　　カ．**ADDA　GR0,=1**