文部科学省後援

情報システム試験

平成19年度後期 情報検定

＜実施　平成20年2月3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41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～13 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～22 ページ
・アセンブラの問題	23 ページ～25 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	26 ページ～31 ページ
・Ｊａｖａの問題	32 ページ～37 ページ
・アセンブラの問題	38 ページ～41 ページ

必須問題

問題１　次の流れ図の説明を読んで，設問に答えよ。

［流れ図の説明］

基本選択法により配列Ｔ［ｉ］（ｉ＝０，１，･･･Ｎ－１）の要素を昇順に並び替える。基本選択法は配列要素の最小値を選択することを繰り返すことで，配列要素を昇順に整列するアルゴリズムである。配列ＴはＮ個の要素を持ち，あらかじめ数値データが格納されているものとする。なお，配列の添字は０から始まるものとする。

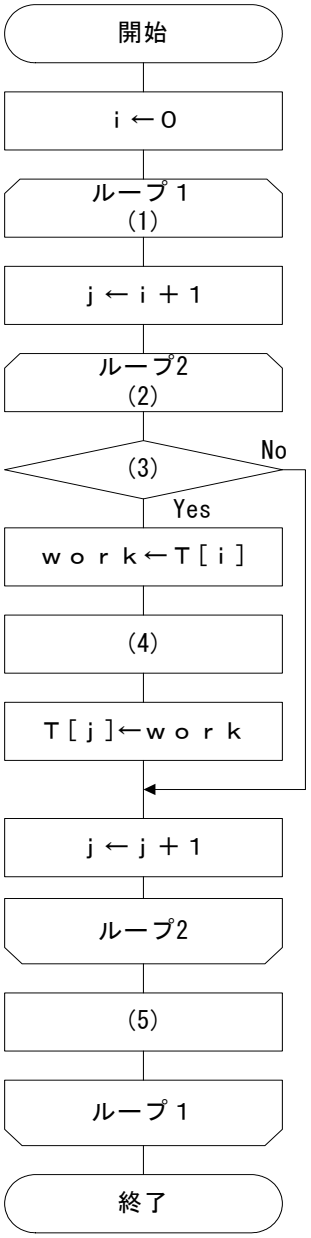


図１　流れ図

＜設問＞　流れ図中の（1）～（5）に入れるべき字句を解答群から選べ。

(1)，(2) の解答群

ア．ｊ＜Ｎ―１の間繰り返す　　イ．ｉ＜Ｎ―１の間繰り返す
ウ．ｊ＜Ｎ―２の間繰り返す　　エ．ｉ＜Ｎ―２の間繰り返す
オ．ｊ＜Ｎ―３の間繰り返す　　カ．ｉ＜Ｎ―３の間繰り返す
キ．ｊ＜Ｎの間繰り返す

(3) の解答群

ア．ｉ＞ｊ　　イ．ｉ＜ｊ
ウ．Ｔ［ｉ］＞Ｔ［ｊ］　　エ．Ｔ［ｉ］＜Ｔ［ｊ］
オ．Ｔ［ｉ］＞Ｔ［ｉ＋１］　　カ．Ｔ［ｉ］＜Ｔ［ｉ＋１］

(4)，(5) の解答群

ア．ｊ←ｊ＋１　　イ．ｉ←ｉ＋１　　ウ．ｗｏｒｋ←ｗｏｒｋ＋１
エ．ｗｏｒｋ←Ｔ［ｊ］　　オ．Ｔ［ｊ］←ｗｏｒｋ　　カ．Ｔ［ｉ］←Ｔ［ｊ］
キ．Ｔ［ｊ］←Ｔ［ｉ］

問題を読みやすくするために,
このページは空白にしてあります。

問題２　月次売上一覧表作成に関する説明を読んで，設問に答えよ。

Ａ社は国内にいくつかの支店を有し，さらに各支店は販売拠点としての営業所を多数配置している。各営業所は，その日の売上データをネットワークにより本社へ送信し，本社はこれらのデータをとりまとめている。

本社では月末に，とりまとめておいた１ヶ月分の売上データをもとに，売上ファイルを作成し，図１で示した「月次売上一覧表」を作成するため，図２の流れ図に従って処理を行っている。

［ 売上データ（送信データ）および売上ファイルのレイアウト ］

支店コード	営業所コード	日付	売上金額

［ 月次売上一覧表の印字イメージ ］

（ 月 次 売 上 一 覧 表 ）

支店コード	営業所コード	日付	売上金額
S10	E1010	2008/02/01	¥13,500
S10	E1010	2008/02/02	¥11,800
:	:	:	:
S10	E1010	2008/02/28	¥11,000
	<<営業所合計>>		¥831,000
S10	E1080	2008/02/01	¥12,000
S10	E1080	2008/02/02	¥13,450
:	:	:	:
S10	E1080	2008/02/27	¥18,900
	<<営業所合計>>		¥967,230
	<<支店合計>>		¥7,876,770
:	:	:	:
:	:	:	:
S80	E8010	2008/02/02	¥11,000
S80	E8010	2008/02/03	¥9,340
:	:	:	:
S80	E8080	2008/02/28	¥20,300
	<<営業所合計>>		¥988,230
	<<支店合計>>		¥7,345,200
	<<総合合計>>		¥88,578,880

図１　月次売上一覧表

［ 月次売上一覧表作成の流れ図 ］

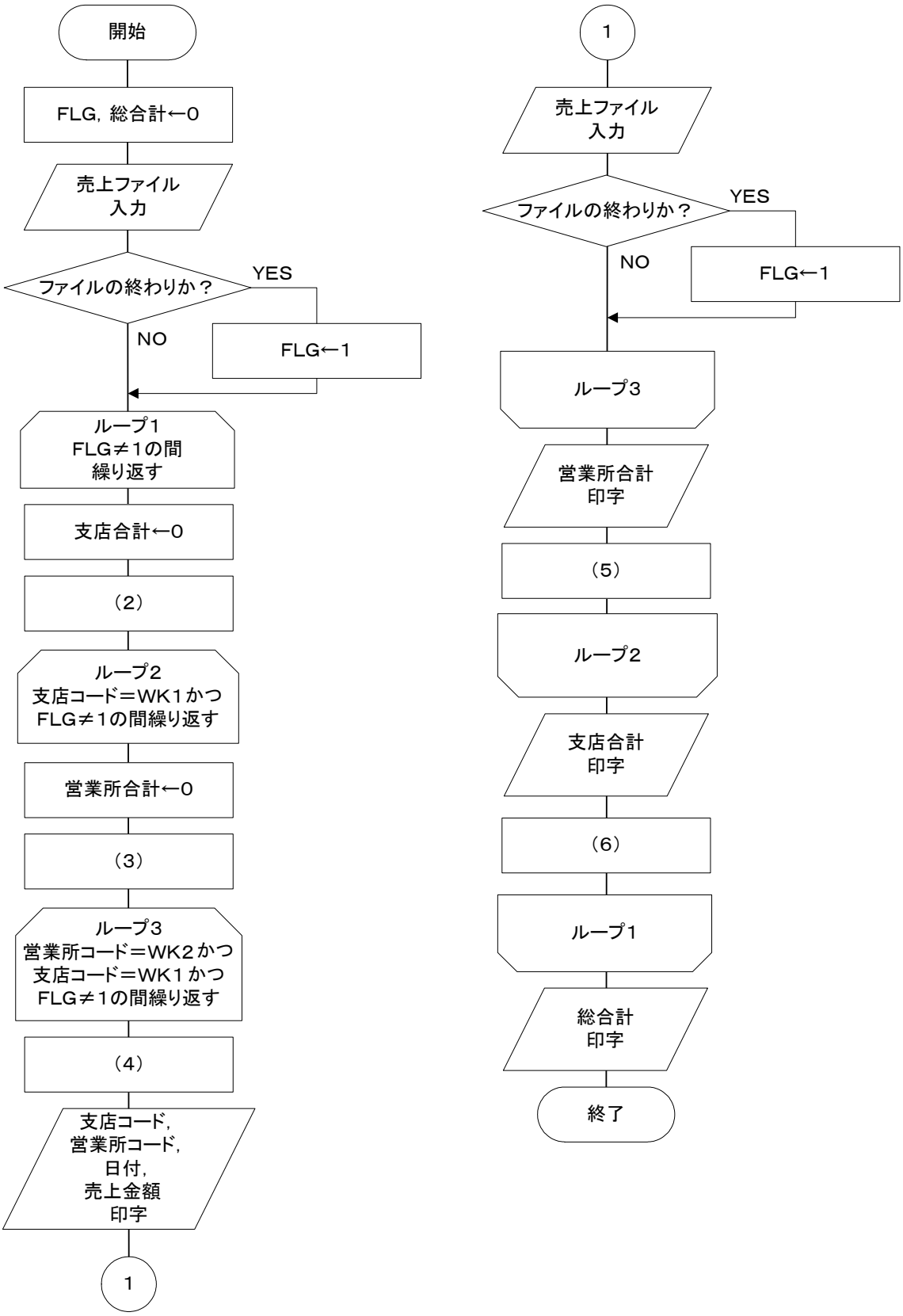


図２　流れ図

＜設問１＞　図１の月次売上一覧表のイメージどおりにするためには，処理時の売上ファイルレコードはどのような並びになっていなければならないか，支店コード・営業所コード・日付それぞれについて並びの適切な組合せを解答群より選べ。

(1) の解答群

	支店コード（第１キー）	営業所コード（第２キー）	日付（第３キー）
ア	順不同	順不同	順不同
イ	昇順	順不同	順不同
ウ	昇順	昇順	昇順
エ	降順	降順	降順

＜設問２＞　図２の流れ図中の（2）～（6）に入れるべき処理を解答群より選べ。

(2) ～ (6) の解答群

ア．総合計 ← 総合計 ＋ 営業所合計
イ．総合計 ← 総合計 ＋ 支店合計
ウ．支店合計 ← 営業所合計 ＋ 総合計
エ．支店合計 ← 支店合計 ＋ 営業所合計
オ．営業所合計 ← 営業所合計 ＋ 売上金額
カ．営業所合計 ← 営業所合計 ＋ 支店金額
キ．ＷＫ１ ← 支店コード
ク．ＷＫ２ ← 支店コード
ケ．ＷＫ１ ← 営業所コード
コ．ＷＫ２ ← 営業所コード

＜設問３＞　売上ファイルにレコードがないまま，図２の流れ図に従って処理してしまった。この場合の処理結果として適切に述べているものを解答群より選べ。

(7) の解答群

ア．営業所合計のみが０として印字される。
イ．支店合計のみが０として印字される。
ウ．総合計のみが０として印字される。
エ．営業所合計，支店合計が０として印字される。
オ．営業所合計，支店合計，総合計が０として印字される。
カ．何も印字されないで終了する。
キ．永久ループとなってしまう。

問題３　次の魔方陣に関する説明および流れ図を読み設問に答えよ。

N×N 個のマス目（N は奇数とする）の中に 1～N×N までの数値を入れたとき，縦・横・斜めの数値の合計が等しい表を魔方陣と呼ぶ。以下の説明は縦 N 行・横 N 列の魔方陣の作成法の例を示したものである。ただし，2 次元配列の行・列の添字は 0 から始まるものとする。

［説明］

$N \times N$ の魔方陣を作成するのに用いる２次元配列 a を用意する。２次元配列 a はダミーの領域を含めて $\left(N+\frac{N}{2}\times 2\right)\times\left(N+\frac{N}{2}\times 2\right)$ の大きさとする。（注意：$\frac{N}{2}$ は小数点以下切捨て）

まず，中央行の最左列のマス目を初期値１の格納位置とする。右斜め上方向に進みながら，１つずつ値を増やし，N までの数値の格納位置とする。次に，初期値１を格納した位置から右斜め下のマス目に移動し，右斜め上方向の N 個のマス目を N+1 から２N までの格納位置とする。その後も１つずつ増やした N 個の値の格納位置を右斜め方向とする。

ただし，値の格納位置がダミーの領域になった場合には格納位置を調整する。右にはみ出たときは左に，左にはみ出したときは右に，上にはみ出したときは下に，下にはみ出したときは上に，N マス移動し魔方陣のマス目内に格納する。

以下の図１－⓪～図１－⑩は 3×3 の魔方陣の作成例であり，網掛けの部分がダミーの領域である。

① 初期値１を格納する位置は中央行の最左列(a[2][0])とする。１を a[2][0]に格納しようとするが，魔方陣の枠から左にはみ出しているので右へ３マス移動し１を格納する。
② 右斜め上に進み，a[1][1]に 2 を格納する。
③ 右斜め上に進み，a[0][2]へ 3 を格納しようとするが，魔方陣の枠から上にはみ出しているので下へ 3 マス移動し 3 を格納する。
④ 格納位置を a[3][1]へ移動する。a[3][1]へ 4 を格納する。
⑤ 右斜め上に進み，a[2][2]に 5 を格納する。
⑥ 右斜め上に進み，a[1][3]に 6 を格納する。
⑦ 格納位置を a[4][2]へ移動する。7 を a[4][2]に格納しようとするが，魔方陣の枠から下にはみ出しているので上へ 3 マス移動し 7 を格納する。
⑧ 右斜め上に進み，a[3][3]へ 8 を格納する。
⑨ 右斜め上に進み，a[2][4]へ 9 を格納しようとするが，魔方陣の枠から右にはみ出しているので左へ 3 マス移動し 9 を格納する。

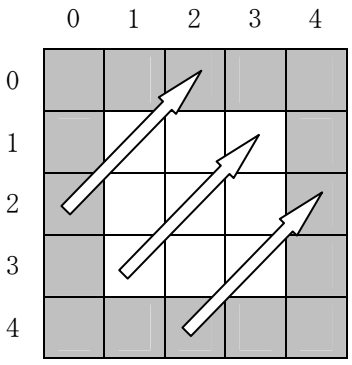


図１－⓪　魔方陣

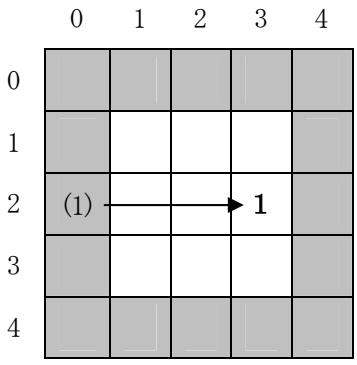


図１－①　魔方陣

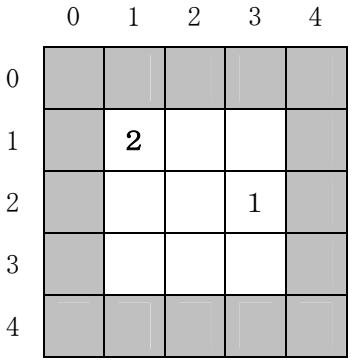


図１－②　魔方陣

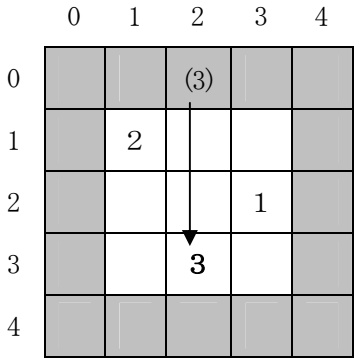


図１－③　魔方陣

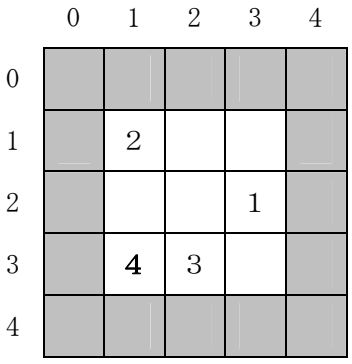


図１－④　魔方陣

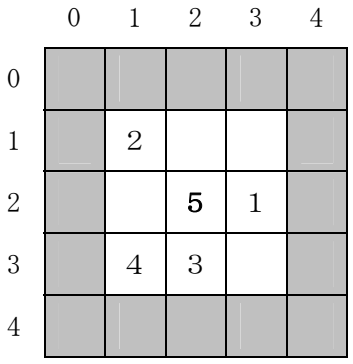


図１－⑤　魔方陣

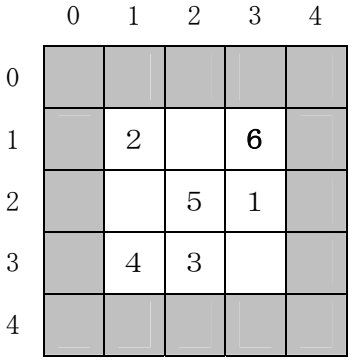


図１－⑥　魔方陣

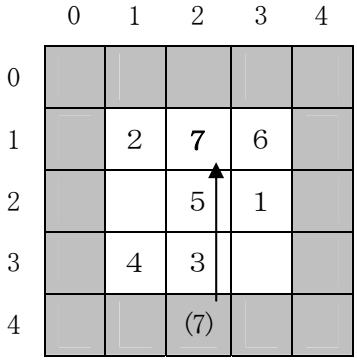


図１－⑦　魔方陣

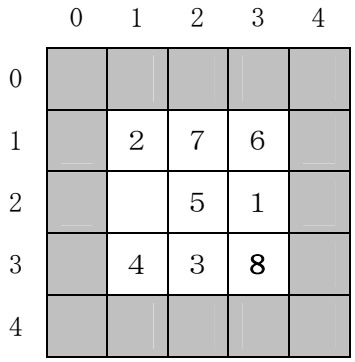


図１－⑧　魔方陣

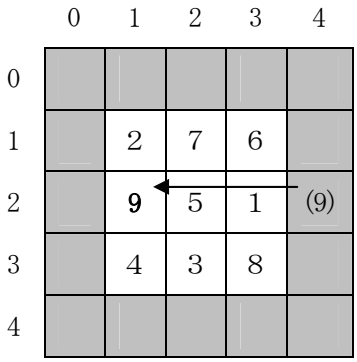


図１－⑨　魔方陣

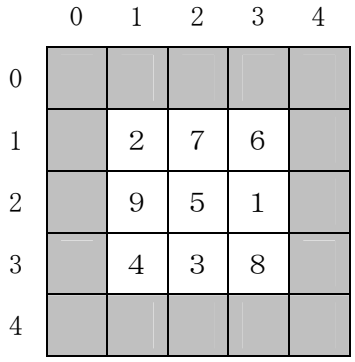
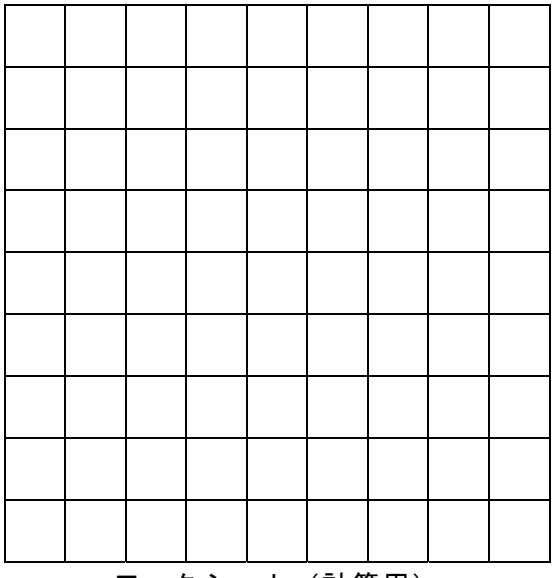


図１－⑩　魔方陣（完成）



ワークシート（計算用）

※このワークシートは設問１の（2）および設問２で利用して下さい。

＜設問１＞　次の記述の［　　］に適切な数字を解答群から選べ。

7×7 の魔方陣を作成するとき，ダミーの領域を含めた２次元配列の行および列の数は［ (1) ］である。また，5×5 の魔方陣を作成するとき，ダミーの領域のマス目の総数は［ (2) ］である。

(1)，(2) の解答群

ア．　７　　イ．　９　　ウ．　１１　　エ．　１３　　オ．　１５
カ．　２４　　キ．　３９　　ク．　５６　　ケ．　７５　　コ．　９６

＜設問２＞　説明の手順を用いて 5×5 の魔方陣を作成したときの完成図（図２）**α** の位置に格納される数値（(3)），**β** の位置に格納される数値（(4)），**γ** の位置に格納される数値（(5)）を解答群から選べ。

α	16	9	22	15
20	8	21	14	**β**
7	25	13	**γ**	19
24	12	5	18	6
11	4	17	10	23

図２　５×５の魔方陣

(3) ～ (5) の解答群

ア．　１　　イ．　２　　ウ．　３　　エ．　４

［流れ図］

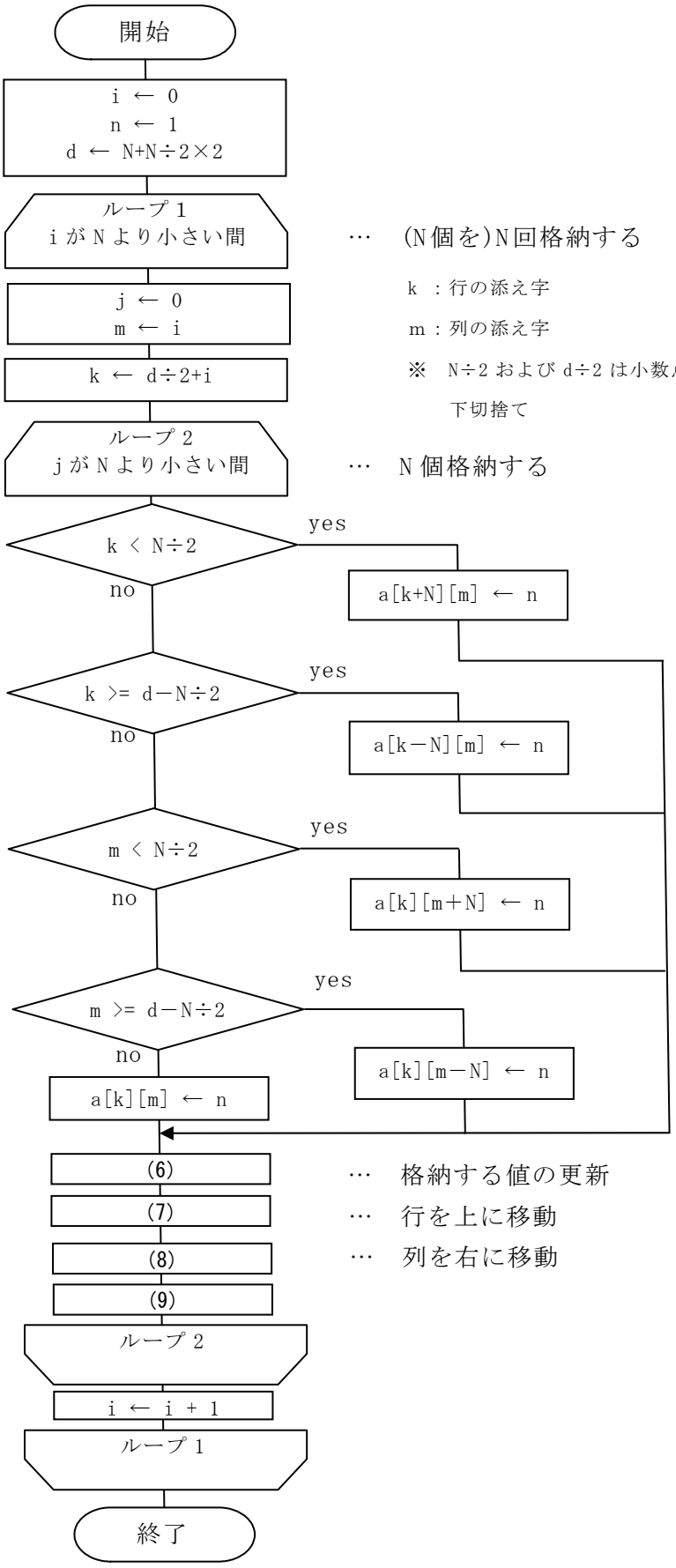


図３　魔方陣作成の流れ図

＜設問３＞　図３の流れ図中の［　　］に適切な式を解答群から選べ。

(6) の解答群

ア．n ← n + 1　　イ．n ← n − 1
ウ．n ← i + 1　　エ．n ← i − 1

(7) の解答群

ア．k ← m + 1　　イ．k ← m − 1
ウ．k ← k + 1　　エ．k ← k − 1

(8) の解答群

ア．m ← m + 1　　イ．m ← m − 1
ウ．m ← k + 1　　エ．m ← k − 1

(9) の解答群

ア．j ← i + 1　　イ．j ← i − 1
ウ．j ← j + 1　　エ．j ← j − 1

問題を読みやすくするために,
このページは空白にしてあります。

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プログラムに関する説明を読んで，設問に答えよ。

ある教科の点数（100 点満点）を入力し，昇順に表示するプログラムを作成した。

[データ構造の説明]

・点数を昇順に表示するために，配列 data と配列 index を用意する。
・点数は入力した順に data へ格納する。ただし，data の先頭には-1 を格納しておく。
・点数の順序を決めるため，index には次の点数の要素を示す添字を格納する。

例えば「30,10,20」と順に点数を入力すると，図１のように配列の内容は変化していく。20 点まで登録した時点で，先頭の index には，最小値である 10 点を示す配列 data の添字 2 が格納され,最大値である 30 点の index には,末尾を示す 0 が格納される。これにより，index を配列の先頭から順にたどると，点数を昇順に表示できる。

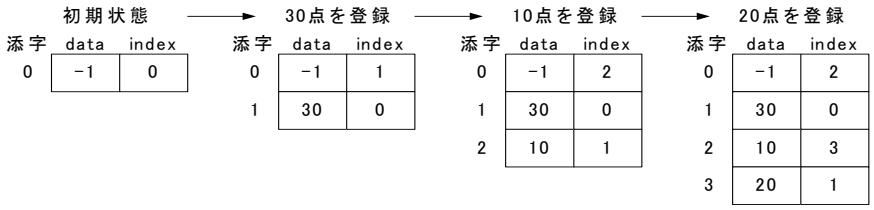


図１　配列内の点数と添字の状態

＜設問１＞　さらに配列に 25 点を図２のように追加した。このとき配列の　(1)　に当てはまる適切なものを解答群から選べ。

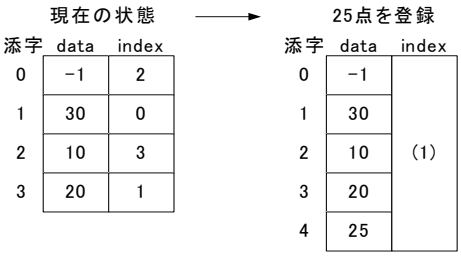


図２　25点の追加

(1) の解答群

ア.

2
0
1
4
3

イ.

2
0
3
1
4

ウ.

2
0
3
4
1

エ.

2
0
4
1
3

［プログラムの説明］

このようなデータ処理を実現するために，次のようなプログラムを作成した。ただし，**data** と **index** は構造体 **PATH** のメンバとして扱い，配列として宣言する。

初期設定

・構造体配列の先頭の **data** は-1，**index** は 0 とする。
・点数を格納する位置は添字 **next** で示し，初期値は 1 とする。

処理概要

・データ入力の終了として 999 が入力されるまで，順次点数を入力し，**check** 関数で点数の範囲チェックを行う。
　エラー時は，エラーメッセージを表示し再入力する。
・入力点数の順序を決めるために，**entryPath** 関数を呼び出す。
・最後に，点数を昇順に表示するために，**printPath** 関数を呼び出す。
・入力データ件数は，50 件を超えないものとする。

［出力例］

```
10 -> 20 -> 25 -> 30
```

［関数仕様］

int entryPath(PATH p[], int next, int in)

引　数：**p**　　… 点数と順序を格納する構造体配列へのポインタ。
　　next　… 次の点数を格納する位置の添字。
　　in　　… 点数
機　能：**in** を **p[next]** のメンバ **data** に格納すると共に，点数を昇順に登録するための位置を探索し，**index** を適切に設定する。
戻り値：**next+1**

void printPath(PATH p[])

引　数：**p**　　… 点数と順序が格納された構造体配列へのポインタ。
機　能：配列 **p** の **index** をたどりながら順に点数を表示する。
戻り値：なし

int check(int in)

引　数：**in**　　… 点数
機　能：**in** の範囲（0～100）をチェックする。
戻り値：範囲内の場合は **1**，それ以外は **0**

[プログラム]

```
#include<stdio.h>
#define MAX 50              // 登録データ数の上限
#define END 999
typedef struct tagPATH{
    int data;               // 点数
    int index;              // 添字
} PATH;
int entryPath(PATH p[], int next, int in);
void printPath(PATH p[]);
int check(int in);
int main(void)
{
    PATH p[MAX + 1];        // 構造体 PATH の配列
    int next, in;
    p[0].data = -1;
    p[0].index = 0;
    (2) ;
    printf("データ -> ");
    scanf("%d", &in);
    while(in != END){
        if(check(in)){
            next = entryPath(p, next, in);
        }else{
            printf("点数範囲エラー¥n");
        }
        printf("データ -> ");
        scanf("%d", &in);
    }
    printPath(p);
    return 0;
}

int entryPath(PATH p[], int next, int in)
{
    int i = 0, j;
 /* 点数を昇順に並べるための位置を探索 */
    while(p[i].index != 0){
        j = p[i].index;                 // 次の点数の添字
    /* 前の点数≦入力点数＜後の点数 */
        if(p[i].data <= in && in < p[j].data)      break;
        i = j;
    }
 /* 点数の格納と添字の設定 */
    p[next].data = in;                          // 入力点数の格納
    p[next].index = (3) ;
    p[i].index = next;                          // 入力点数位置の添字の格納
    return next + 1;
}
```

```
int check(int in)
{
    return  (4)  ;
}

void printPath(PATH p[])
{
    int i = 0, j;
    while(p[i].index != 0){
        j = p[i].index;  // 次の点数を示す配列の添字
        printf("%d", p[j].data);
        if(  (5)  )      printf(" -> ");
        i = j;
    }
    printf("¥n");
}
```

＜設問２＞　プログラム中の［　　］に入れるべき適切な字句を（2）～（5）の解答群から選べ。

(2) の解答群

ア．**next = -1**　　イ．**next = 0**

ウ．**next = 1**　　エ．**next = END**

(3) の解答群

ア．**p[i].index**　　イ．**p[j].index**

ウ．**p[next].index**　　エ．**p[0].index**

(4) の解答群

ア．**in >= 0 || in <= 100**　　イ．**in >= 0 && in <= 100**

ウ．**in <= 0 || in >= 100**　　エ．**in <= 0 && in >= 100**

(5) の解答群

ア．**p[i].data != 0**　　イ．**p[j].data != 0**

ウ．**p[i].index != 0**　　エ．**p[j].index != 0**

選択問題Ａ　Javaの問題

次の Java プログラムに関する説明を読んで，設問に答えよ。

［プログラムの説明］

複数のチームが総当たり戦で試合を行う JK リーグに対して，試合ごとの結果を入力していくことで，最終的なチームの勝数を表示するプログラムである。なお，JK リーグでは試合に引き分けという結果はない。表１に結果の例を示す。

表１　JK リーグの試合結果

対戦相手 / チーム名	A チーム	B チーム	C チーム	勝ち数
A チーム		3 － 1	4 － 1	2
B チーム	1 － 3		1 － 2	0
C チーム	1 － 4	2 － 1		1

［クラスの仕様］

Team クラス

説明	チームの情報を扱うクラス
フィールド	**name**：チーム名，**winCount**：勝ち数
コンストラクタ	引数で渡されたチーム名をフィールドにセットする。
メソッド	**setCount**　：試合結果に応じて勝ち数を加算する。
	dispCount　：チームの勝ち数を出力する。

League クラス

説明	JK リーグ全体の情報を扱うクラス
フィールド	**teams**：各チームの情報，**registIdx**：チーム情報の保持数
コンストラクタ	引数で渡されたチーム数分の領域を確保する。
メソッド	**setRegist**　：チームをセットする。
	setResult　：試合結果をセットする。
	dispResult：総当たり戦の結果を出力する。

［プログラム］

```
class Team{
    private String name;
    private int winCount = 0;

    Team(String name){
        this.name = name;
    }
    public void setCount(int getPoint, int lostPoint){
        if(getPoint > lostPoint){
            winCount = [ (1) ];
        }
    }
    public void dispCount(){
        System.out.println(name + "の勝ち数：" +  winCount);
    }
}
```

```
class League{
    private Team[] teams;
    private int registIdx = 0;

    League(int teamCnt){
        teams = new Team[teamCnt];
    }
    public void setRegist(Team team){
        teams[registIdx] = team;
        registIdx = [ (2) ];
    }
    public void setResult(Team t1, int t1Point, Team t2, int t2Point){
        t1.setCount(t1Point, t2Point);
        t2.setCount(t2Point, t1Point);
    }
    public void dispResult(){
        for(int i = 0; i < teams.length; i++){
            teams[i].dispCount();
        }
    }
}

//実行クラス
class Game{
    public static void main(String[] args){
        League jkLeague = new League(3);
        Team aTeam = new Team("Aチーム");
        Team bTeam = new Team("Bチーム");
        Team cTeam = new Team("Cチーム");

        jkLeague.setRegist(aTeam);
        jkLeague.setRegist(bTeam);
        jkLeague.setRegist(cTeam);

        jkLeague.setResult(aTeam, 3, bTeam, 1);
        jkLeague.setResult(aTeam, 4, cTeam, 1);
        jkLeague.setResult(bTeam, 1, cTeam, 2);

        jkLeague.dispResult();
    }
}
```

＜設問１＞　プログラム中の［　　］に入れるべき適切な字句を（1），（2）の解答群から選べ。

(1) の解答群

ア．`winCount + 1`　　イ．`getPoint`

ウ．`getPoint + 1`　　エ．`getPoint - lostPoint`

(2) の解答群

ア．`0`　　イ．`1`

ウ．`registIdx + 1`　　エ．`registIdx - 1`

＜設問２＞　このプログラムを実行した場合，出力される結果として適切なものを(3)の解答群から選べ。

(3) の解答群

ア.
```
A チームの勝ち数：2
C チームの勝ち数：1
B チームの勝ち数：0
```

イ.
```
A チームの勝ち数：2
B チームの勝ち数：0
C チームの勝ち数：1
```

ウ.
```
B チームの勝ち数：0
C チームの勝ち数：1
A チームの勝ち数：2
```

エ.
```
C チームの勝ち数：1
B チームの勝ち数：0
A チームの勝ち数：2
```

＜設問３＞　このプログラムを使って別のリーグの試合結果を扱いたい。ただし，ここでは表２に示すように「引き分け（同点）」という試合結果があり得る。以下の各問に答えよ。

表２　別のリーグの試合結果

対戦相手 / チーム名	A チーム	B チーム	C チーム	勝ち数	引き分け数
A チーム		3 － 1	1 － 1	1	1
B チーム	1 － 3		1 － 2	0	0
C チーム	1 － 1	2 － 1		1	1

(4) 現状の **Team** クラス，**League** クラスをそのまま用い，実行クラス内で

```
jkLeague.setResult(aTeam, 1, cTeam, 1)
```

のように **League** クラスの **setResult** メソッドの引数に，引き分けとなった試合の結果を渡した場合，どのように処理されるか。適切なものを（4）の解答群から選べ。

(4) の解答群

ア．両チームとも勝ち数が加算される　　イ．両チームとも勝ち数は加算されない
ウ．**aTeam** のみ勝ち数が加算される　　エ．**cTeam** のみ勝ち数が加算される

(5) 引き分けを処理するため，**Team** クラス内で引き分け数を扱うフィールド **drawCount** を追加することにした。このフィールドに関する説明として，適切でないものを（5）の解答群から選べ。

(5) の解答群

ア．**drawCount** のデータ型を **int** 型とし，0 で初期化する
イ．**Team** クラスの **setCount** メソッドで，両チームの得点に応じて加算する
ウ．**Team** クラスの **dispCount** メソッドで，引き分け数も出力対象とする
エ．**League** クラスの **setResult** メソッドの引数で，**drawCount** を使用する必要がある

選択問題Ａ　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。

〔アルゴリズムの説明〕

整数 2 から 100 までの中から素数を求める手順を以下に示す。

① 先頭番地をＴＡＢＬＥ番地とした 99 語の連続領域に整数２から 100 までを格納する。

ＴＡＢＬＥ番地以降

2	3	4	5	6	7	8	9	…	95	96	97	98	99	100

② ＴＡＢＬＥ番地の先頭から順に 0 でない要素を見つける。0 でない要素が見つかったら，その要素の倍数をすべて 0 にする。

（例－１）最初に見つかる 0 でない要素は２なので，2 を除く 2 の倍数をすべて 0 にする。

2	3	0	5	0	7	0	9	…	95	0	97	0	99	0

（例－２） 次の 0 でない要素は 3 なので，3 を除く 3 の倍数をすべて 0 にする。

2	3	0	5	0	7	0	0	…	95	0	97	0	0	0

③ ②の操作を要素が１１を超えるまで繰り返し行う。

④ ＴＡＢＬＥから始まる要素の中で，0 でない要素をＡＮＳ番地以降に格納する。

以上のアルゴリズムで，ＡＮＳ番地以降に 2 から 100 までの中の素数を求めることができる。

〔プログラム〕

```
PGM       START
          LAD     GR1,0
          LAD     GR2,2
LOOP0     ST      GR2,TABLE,GR1        ; TABLE 番地以降に整数値を格納
          ADDA    GR1,=1
          ADDA    GR2,=1
          CPA     GR1,C99
          |       (1)                |
          LAD     GR0,0
          LAD     GR1,0
LOOP1     LD      GR2,GR1
LOOP2     CPA     GR0,TABLE,GR2
          JNZ     L0
          ADDA    GR2,=1
          JUMP    LOOP2
L0        CPA     GR2,C10
          |       (2)                |  ; 素数を取り出す処理へ分岐
          LD      GR3,TABLE,GR2
          LD      GR1,GR2
LOOP3     ADDA    GR2,GR3              ; 素数の倍数を 0 にする処理
          CPA     GR2,C99
          JPL     LOOP4
          JZE     LOOP4
          |       (3)                |
          JUMP    LOOP3
LOOP4     ADDA    GR1,=1
          JUMP    LOOP1
OUTPUT    LAD     GR0,0
          LAD     GR1,0
          LAD     GR2,0
LOOP5     CPA     GR0,TABLE,GR1
          JZE     L1                   ; 0 でない整数値が素数
          LD      GR3,TABLE,GR1
          |       (4)                |  ; 素数を ANS 番地以降に格納
          ADDA    GR2,=1
L1        ADDA    GR1,=1
          CPA     GR1,C99
          JMI     LOOP5
          RET
```

```
TABLE     DS      99
C99       DC      99
ANS       DS      30
C10       DC      10
          END
```

＜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．　JUMP　LOOP0　　イ．　JMI　　LOOP0　　ウ．　JPL　　LOOP1
エ．　JZE　　LOOP0

(2) の解答群

ア．　JMI　　LOOP5　　イ．　JPL　　LOOP5　　ウ．　JMI　　L1
エ．　JPL　　L1　　オ．　JMI　　OUTPUT　　カ．　JPL　　OUTPUT

(3) の解答群

ア．　LD　　GR0,TABLE,GR2　　イ．　ST　　GR0,TABLE,GR2
ウ．　LD　　GR3,TABLE,GR2　　エ．　ST　　GR3,TABLE,GR2

(4) の解答群

ア．　LD　　GR0,GR2　　イ．　LAD　　GR2,1,GR3
ウ．　ST　　GR0,TABLE,GR2　　エ．　ST　　GR3,ANS,GR2
オ．　LD　　GR2,ANS,GR3

＜設問２＞　このプログラムを実行したとき，同じ箇所に何回も 0 が格納されることがある。ＴＡＢＬＥ番地以降で 84 が格納されている箇所には何回 0 が格納されるか (5) の解答群より選べ。

(5) の解答群

ア．　0 回　　イ．　1 回　　ウ．　2 回　　エ．　3 回　　オ．　4 回

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んで，設問に答えよ。

［処理の概要］

流れ図を描くためのエディタを作成することにした。ここではあらかじめテンプレートに用意した流れ図記号の基本形状から，移動と拡大によって個々の流れ図記号を生成する関数の実装を考えた。

表　テンプレートに用意した流れ図記号の属性

記号ID	名称	基本形状	拡大方向 縦	拡大方向 横	回転
1	端子		—	○	—
2	処理		○	○	—
3	判断		○	○	—
4	ループ始端		○	○	—
5	ループ終端		○	○	—
6	データ		○	○	—
7	線		○	—	○
8	結合子		—	—	—

○ … 指定可能
— … 指定不可

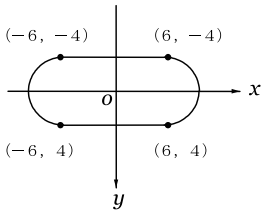


図１　テンプレートの例

テンプレートに用意した流れ図記号を表に示す。拡大方向とは，個々の記号を生成する際に，基本形状を拡大することができる方向のことである。例えば，端子記号は横方向のみ拡大ができる。また，線記号は90°のみ回転させることができる。

［流れ図記号の表現方法］

記号の中心を原点とした基本形状のデータ（各頂点の座標等）を，テンプレートとして用意しておく。

中心座標，縦の拡大率，横の拡大率，回転角の組合わせによって，テンプレートを移動・変形させ，個々の記号を生成する。

［関数の仕様］

Symbol getSymbol(int ID, Coord p, float magV, float magH, int angle)

引　数：**ID** … 記号 ID，**p** … 中心座標，
　　magV … 縦方向の倍率，**magH** … 横方向の倍率，**angle** … 回転角

機　能：与えられた引数から頂点座標を計算し，記号を生成する。

戻り値：生成された記号。
　　引数が不当なときはエラーとし，**ID** に 0 を設定。

この関数で引数のエラーと判断する条件を次に示す。ただし，結合子に対する倍率の指定等，記号の種類によって意味をもたない引数については，これらの条件を適用しない。

(1) **ID** が 1～8 のいずれでもないとき
(2) 倍率として 1.0 未満の値が指定されたとき
(3) 回転角として 0° または 90° 以外が指定されたとき

＜設問１＞　引数が不当である **getSymbol** 関数の呼出しを (1) の解答群から選べ。ただし，変数 **p** には適切に値が設定されているものとする。

(1) の解答群

ア. **getSymbol(1, p, 0.5, 1.0, 0)**
イ. **getSymbol(2, p, 0.5, 1.0, 0)**
ウ. **getSymbol(7, p, 1.0, 0.5, 90)**
エ. **getSymbol(8, p, 1.0, 0.5, -90)**

［プログラム１］

```
/* 座標を管理する構造体 */
  typedef struct {
     int x, y;
  } Coord;
/* 記号の管理情報を保持する構造体 */
  typedef struct {
     int ID;                    // 記号 ID
     Coord vertex[6];           // 頂点等の座標
  } Symbol;
/* 基本形状のデータ（テンプレート） */
  Coord template[9][6] = { {0},              // 添字調整用のダミー
     {{6,4},{-6,4},{-6,-4},{6,-4}},          // 端子
     {{9,4},{-9,4},{-9,-4},{9,-4}},          // 処理
     {{9,0},{ 0,4},{-9, 0},{0,-4}},          // 判断
     {{9,4},{-9,4},{-9, 0},{-7,-4},{7,-4},{9,0}},    // ループ始端
     {{7,4},{-7,4},{-9, 0},{-9,-4},{9,-4},{9,0}},    // ループ終端
     {{6,4},{-9,4},{-6,-4},{9,-4}},          // データ
     {{0,3},{0,-3}},                         // 線
     {{0,3},{0,-3}} };                       // 結合子
  int np[] = {0, 4, 4, 4, 6, 6, 4, 2, 2};  // 頂点等の数
```

```
Symbol getSymbol(int ID, Coord p, float magV, float magH, int angle)
{
    Symbol s;
    int i, hold;
/* エラーチェック */
    if ( ID < 1 || ID > 8 )
        s.ID = 0;
    else if ( (2) )
        s.ID = 0;
    else if ( ID >= 2 && ID <= 6 && (magV < 1.0 || magH < 1.0) )
        s.ID = 0;
    else if ( ID == 7 && (magV < 1.0 || angle != 0 && angle != 90) )
        s.ID = 0;
    else
        (3);                          // エラーでないとき
/* 記号毎の移動・変形 */
    switch (s.ID) {
    case 1:   // 端子
        for ( i = 0; i < np[s.ID]; i++ ) {
            s.vertex[i].x = p.x + template[s.ID][i].x * magH;
            s.vertex[i].y = p.y + template[s.ID][i].y;
        }
        break;
    case 2: case 3: case 4: case 5: case 6: // 処理 ～ データ
        for ( i = 0; i < np[s.ID]; i++ ) {
            s.vertex[i].x = p.x + template[s.ID][i].x * magH;
            s.vertex[i].y = p.y + template[s.ID][i].y * magV;
        }
        break;
    case 7:   // 線
        for ( i = 0; i < np[s.ID]; i++ ) {
            s.vertex[i].x = template[s.ID][i].x;
            s.vertex[i].y = template[s.ID][i].y * magV;
            if ( angle == 90 ) {          // 回転
                hold = s.vertex[i].x;
                s.vertex[i].x = s.vertex[i].y;
                s.vertex[i].y = hold;
            }
            s.vertex[i].x += p.x;
            s.vertex[i].y += p.y;
        }
        break;
    case 8:   // 結合子
        s.vertex[0] = template[s.ID][0];
        s.vertex[1] = template[s.ID][1];
        break;
    }
    return s;
}
```

＜設問２＞　プログラム中の［　　］に入れるべき適切な字句を（2），(3) の解答群から選べ。

(2) の解答群

ア．**ID == 1 && magH < 1.0**　　イ．**ID == 1 && magV <= 1.0**

ウ．**ID == 1 || magH <= 1.0**　　エ．**ID == 1 || magV < 1.0**

(3) の解答群

ア．**s.ID = 0**　　イ．**s.ID = ID**

ウ．**s.ID = ID + 1**　　エ．**s.ID++**

＜設問３＞　線記号の基本形状を 2 倍して 90° 回転させたものを生成するために，次のように変数 **p** と **s** を宣言して **getSymbol** 関数を呼び出した。得られた結果を **(s.vertex[0].x, s.vertex[0].y)-(s.vertex[1].x, s.vertex[1].y)** の形式で表現したとき，適切なものを（4）の解答群から選べ。

```
Coord  p = {30, 20};
Symbol s;
s = getSymbol(7, p, 2.0, 0.0, 90);
```

(4) の解答群

ア．**(30, 14)-(30, 26)**　　イ．**(30, 26)-(30, 14)**

ウ．**(24, 20)-(36, 20)**　　エ．**(36, 20)-(24, 20)**

［リストによる記号間の連結情報の管理］

プログラムの機能を拡張して，線記号を除く流れ図記号の連結情報をリストで管理し，簡単なエラーチェックをすることにした。リストを構成するノードは，次に示す構造体を用いて表現する。

```
typedef struct node {
   struct node *next; // 後続するノードへのポインタ
   Symbol *s;         // 記号を管理する構造体へのポインタ
} Node;
```

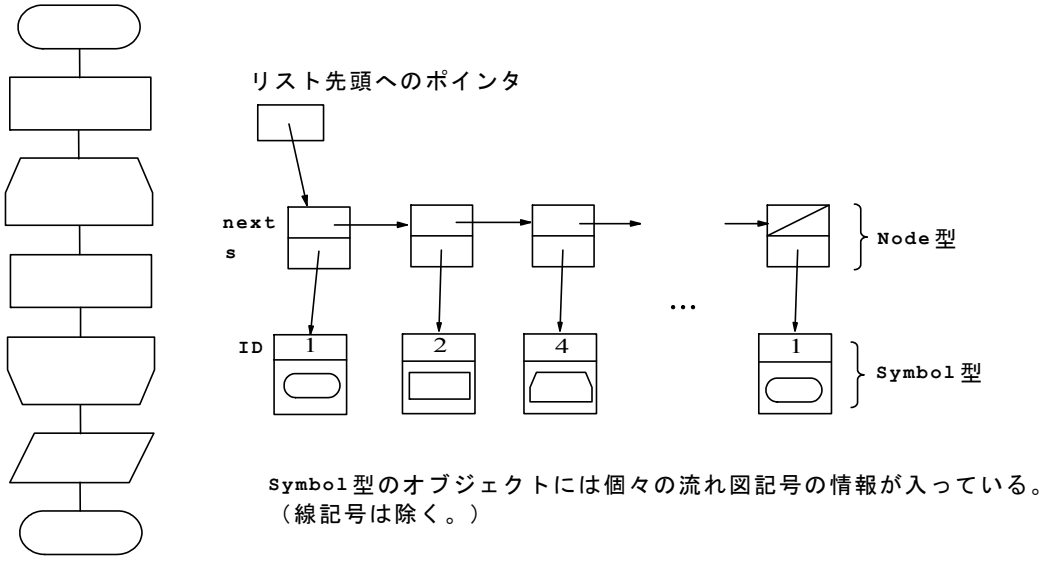


図２　流れ図の例とリストの概念図

図２に示す通り，**Symbol** 型で表現された個々の記号は，リストを構成するノードからポインタによって指されている。リストを先頭からたどることで，記号の連結順序を調べることができる。

［関数の仕様］

次に示す **check** 関数は，リストを先頭からたどり，記号の連結順を調べる。関数内で用いられているマクロは適切に宣言されているものとする。

チェックするエラーは次の２項目とする。

①端子記号で始まっていない，または端子記号で終わっていない。

②ループ始端記号の個数と，ループ終端記号の個数が一致していない。

int check(Node *head)

引　数：リスト先頭へのポインタ

機　能：リストを先頭からたどり，エラーチェックを行う。

戻り値：**1** … ①のみに該当するとき

2 … ②のみに該当するとき

3 … ①と②の両方に該当するとき

0 … ①，②のいずれにも該当しないとき

［プログラム２］

```
    int check(Node *p)
    {
        Node *q;
        int errcode = 0, level = 0;

        if ( p->s->ID != 1 )          // 先頭が端子でないとき
            (6) ;
α ⟹  while ( p != NULL ) {
            if ( p->s->ID == 4 )       // ループ始端のとき
                level++;
            else if ( p->s->ID == 5 ) // ループ終端のとき
                level--;
            (7) ;                      // 現在注目しているノードのアドレスを保持
            p = p->next;
        }
        if ( q->s->ID != 1 )          // 末尾が端子でないとき
            errcode |= 1;
        if ( (8) )                    // ループ始端と終端の個数が一致しないとき
            errcode |= 2;
        return errcode;
    }
```

＜設問４＞　図２に示す流れ図の例を **check** 関数で処理した場合，α行の制御式の評価回数は何回か。適切な値を（5）の解答群から選べ。

(5) の解答群

ア．7　　イ．8

ウ．13　　エ．14

＜設問５＞　プログラム中の［　　］に入れるべき適切な字句を（6）～（8）の解答群から選べ。

(6) の解答群

ア．**errcode = 1**　　イ．**errcode = 2**

ウ．**errcode = 3**　　エ．**errcode |= 3**

(7) の解答群

ア．**p++**　　イ．**p = q**

ウ．**q = p**　　エ．**q = q->next**

(8) の解答群

ア．**level == 0**　　イ．**level == 1**

ウ．**level != 0**　　エ．**level < 0**

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。

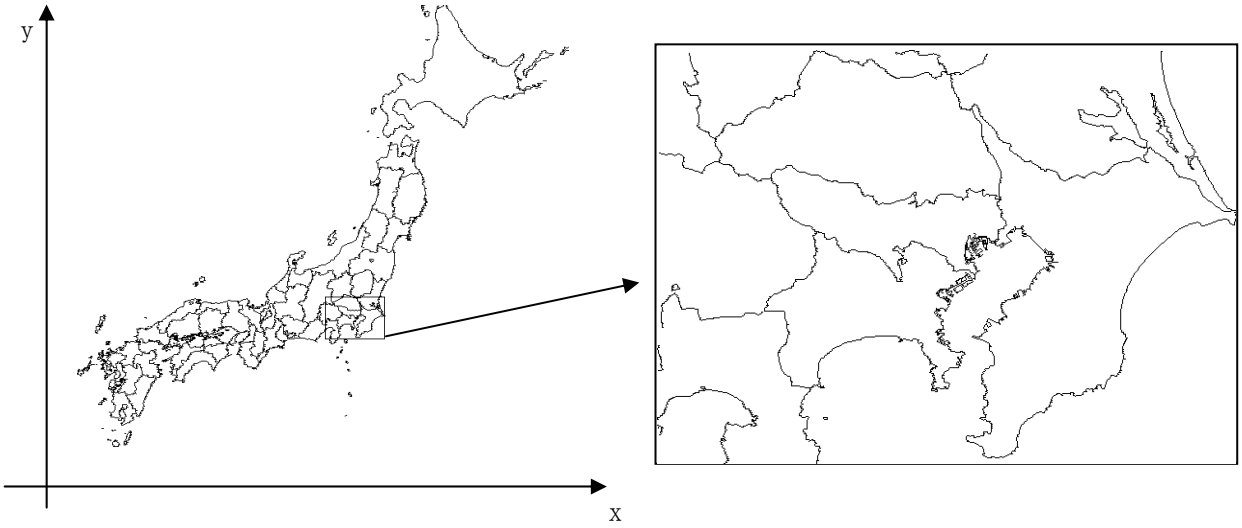


図 1

［プログラムの説明１］

地図を表示するため，広域地図から矩形領域を 1 箇所だけ抜き出すことを考える。矩形領域は平行移動したり拡大縮小したりといった調整をすることでさまざまな領域を抜き出せるようにしたい。そのため，以下に示す矩形などを扱うクラス群を作成した。なお，プログラムでは建物などの対象物をすべて点として扱うこととする。

［クラスの仕様］

Coord クラス

説明	地図上の点を**(x,y)**の座標で扱うクラス	
フィールド	**x, y**	：座標の各成分
コンストラクタ	座標をフィールドにセットする。	
メソッド	**scale**	：**(x,y)**の各成分に引数の値をかける。
	translate	：**(x,y)**の各成分を引数の値だけ移動する。

Rect クラス

説明	地図上の矩形領域を対角線上の頂点（始点と終点）で扱うクラス	
フィールド	**r**	：矩形のインスタンスを扱う。
	s, e	：始点と終点
コンストラクタ	矩形の始点（左下），終点（右上）をセットする。	
メソッド	**getRect**	：矩形のインスタンスを生成する。
	getCenter	：矩形の中心位置を返す。
	scale	：矩形を拡大縮小する。
	translate	：始点・終点を引数の値だけ移動する。
	isInside	：引数に指定した点が矩形領域内かどうかを判定する。

[プログラム]

```
//座標を扱うクラス
class Coord{
    double x, y;

    Coord(double x, double y){
        this.x = x;
        this.y = y;
    }
    //座標(x,y)の数値をmag倍する
    void scale(double mag){
        x *= mag;
        y *= mag;
    }
    //座標(x,y)を(tx,ty)だけ移動
    void translate(double tx, double ty){
        x += tx;
        y += ty;
    }
}

//矩形を扱うクラス
class Rect{
    private static Rect r = null;      //矩形のインスタンスを扱う
    Coord s, e;                        //始点（左下）と終点（右上）の座標

    private Rect(double sx, double sy, double ex, double ey){
        s = new Coord(sx, sy);
        e = new Coord(ex, ey);
    }
    //矩形のインスタンスを生成
    static Rect getRect(double sx, double sy, double ex, double ey)
    throws Exception{
        if(r == null){
            r = new Rect(sx, sy, ex, ey);
        }else{
            throw new Exception("矩形は既に存在します。引数は無効です。");
        }
        return r;
    }
    //矩形の中心位置を返す
    public Coord getCenter(){
        return new Coord((s.x + e.x) / 2, (s.y + e.y) / 2);
    }
    //矩形の始点・終点をそれぞれ(tx,ty)だけ移動
    public void translate(double tx, double ty){
        s.translate(tx, ty);
        e.translate(tx, ty);
    }
```

```
    //矩形の中心で矩形を拡大縮小
    public void scale(double mag){
        Coord c = getCenter();  //矩形の中心位置を取得する
        translate(-c.x, -c.y);  //矩形の中心を原点に移動
        s.scale(mag);           //始点の値を mag 倍にする
        e.scale(mag);           //終点の値を mag 倍にする
        translate(c.x, c.y);    //矩形の中心を元の位置に戻す
    }
    //指定した点が矩形領域内かどうかを返す
    public boolean isInside(Coord p){
        return s.x <= p.x && p.x <= e.x && s.y <= p.y && p.y <= e.y;
    }
}
```

＜設問１＞　地図上の矩形を操作するためには，**Rect** クラスのインスタンスを生成して移動や拡大縮小を行うメソッドを呼び出す必要がある。操作の例を示す図２を見て，次の各問に答えよ。

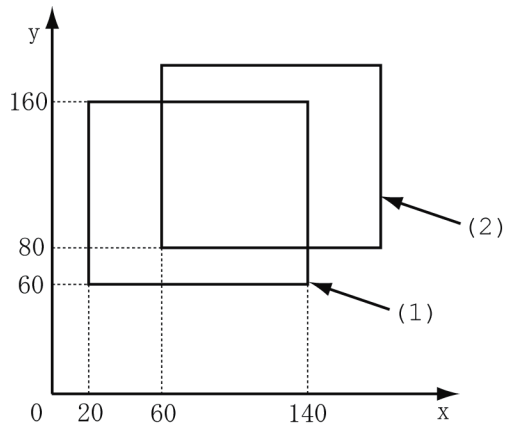


図２

(1) 図２中の（1）で示す範囲を指定する **Rect** クラスのインスタンスを以下のプログラム中の（1）で生成したい。記述として適切なものを（1）の解答群から選べ。

```
class TestMap{
    public static void main(String[] args){
        Rect r = null;
        try{
            r = [ (1) ];
                ⋮
```

(1) の解答群

ア．**new Rect(20.0, 60.0, 140.0, 160.0)**

イ．**new Rect(20.0, 160.0, 140.0, 60.0)**

ウ．**Rect.getRect(20.0, 60.0, 140.0, 160.0)**

エ．**Rect.getRect(20.0, 160.0, 140.0, 60.0)**

(2) 上記で作成した矩形を図２中の（2）の位置へ平行移動するためのメソッドの呼び出しとして適切なものを（2）の解答群から選べ。

(2) の解答群

ア. **r.translate(20.0, 40.0);**　　イ. **r.translate(40.0, 20.0);**

ウ. **r.translate(60.0, 80.0);**　　エ. **r.translate(80.0, 60.0);**

(3) 上記で移動した矩形の大きさ（面積）を４倍にするためのメソッドの呼び出しとして適切なものを（3）の解答群から選べ。

(3) の解答群

ア. **r.scale(2.0);**　　イ. **r.scale(0.5);**

ウ. **r.scale(4.0);**　　エ. **r.scale(0.25);**

＜設問２＞　駅など目印となる対象物の座標**(x,y)**が中心となるように矩形領域を移動したい。そのため，以下に示す **setCenter** メソッドを **Rect** クラスへ追加することにした。［　　］に入れるべき適切な字句を（4）の解答群から選べ。

```
public void setCenter(double x, double y){
    Coord c = getCenter();
    translate( (4) );
}
```

(4) の解答群

ア. **x, y**

イ. **-x, -y**

ウ. **x + c.x, y + c.y**

エ. **x - c.x, y - c.y**

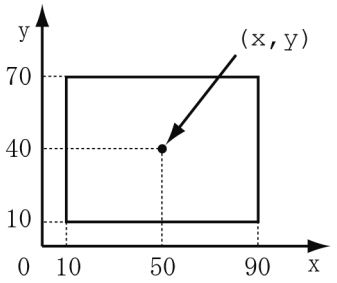


図３

＜設問３＞　建物などを１点であらわすとき，その建物の位置が矩形領域の範囲内にあるかどうかを **isInside** メソッドで判別することにした。使用例として，図４の駅ビルが範囲内にあるかどうかを確認するコードを記述したとき，［　　］に入れるべき適切な字句を（5）の解答群から選べ。なお変数 **r** は **Rect** クラスの変数で，インスタンスがすでに渡されているものとする。

```
if(r.isInside( (5) )){
    System.out.println("範囲内です");
}else{
    System.out.println("範囲外です");
}
```

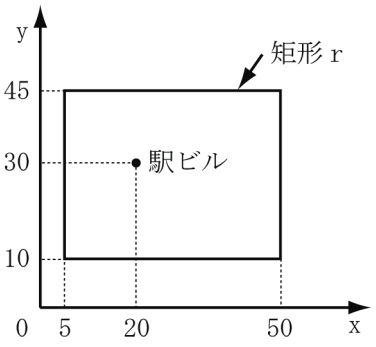


図４

(5) の解答群

ア．**20.0, 30.0**

イ．**new Rect(5.0, 10.0, 50.0, 45.0)**

ウ．**Rect.getRect(5.0, 10.0, 20.0, 30.0)**

エ．**new Coord(20.0, 30.0)**

［プログラムの説明２］

図５のように，地図上の「川」や「道路」「鉄道」などをそれらの中心線に複数の点を配置して，点の集まりとして扱うことにした。ただし，これらの点はすべてが指定した矩形領域内に収まるとは限らないため，図５の点２～７のように指定した矩形領域内の点だけを抜き出す機能をもつ **Poly** クラスを追加作成した。

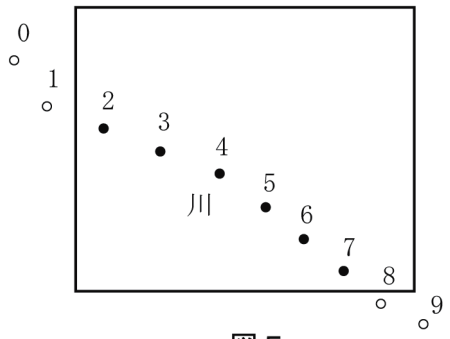


図５

［クラスの仕様］

Poly クラス

説明	対象物を点の集まりで扱うクラス	
フィールド	**c**	：対象物全体を構成する点の配列
コンストラクタ	対象物の位置をフィールドにセットする。	
メソッド	**innerRegion**	：領域内の点を返す。
	getInsideCount	：領域内の点の数を返す。
	getPosition	：領域内の点の内の１点の位置を返す。

［プログラム］

```
class Poly{
    Coord[] c;

    Poly(Coord[] c){
        this.c = c;
    }
```

```
    //領域内のすべての点を返す
    Coord[] innerRegion(Rect r){
        int cnt = 0;
        int n = getInsideCount(r);
        if(n == 0) return null;
        Coord[] c = new Coord[n];
        for(int i = 0; i < [ (6) ]; i++){
            if(r.isInside(this.c[i])){
                c[cnt++] = this.c[i];
            }
        }
        return c;
    }
    //領域内の点の数を返す
    private int getInsideCount(Rect r){
        int cnt = 0;
        for(int i = 0; i < c.length; i++){
            if(r.isInside(c[i])){
                cnt++;
            }
        }
        return cnt;
    }
    //名前を付与するための１点の位置を取り出す
    [ (7) ] getPosition(Rect r){
        int n = getInsideCount(r);
        if(n == 0) return null;
        return innerRegion(r)[n / 2]; //配列の１点を返す
    }
}
```

＜設問４＞　プログラム中の［　　］に入れるべき適切な字句を（6），(7) の解答群から選べ。

(6) の解答群

ア．**c.length**　イ．**this.c.length**　ウ．**cnt**　エ．**getInsideCount(r)**

(7) の解答群

ア．**int**　イ．**Rect**　ウ．**Coord[]**　エ．**Coord**

＜設問５＞　図５に示した「川」の例では **getPosition** メソッドの戻り値はどの点を示すか。図中に示された数字の中で適切なものを（8）の解答群から選べ。

(8) の解答群

ア．**3**　イ．**4**　ウ．**5**　エ．**7**

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み，設問に答えよ。

［プログラムの説明］

文字列の暗号化を行う。ここでは，もとの文字列を平文，暗号化された文字列を暗号文，暗号化および復号化のキーとする文字列をパスワード文とする。なお，平文，暗号文，パスワード文ともに，文字データ１語の上位８ビットは必ず０である。

（ａ）暗号化のアルゴリズム

平文とパスワード文の１語ずつの排他的論理和を行い，結果の下位８ビットを16進数２桁の文字列として出力する。

なお，平文の文字数よりパスワード文の文字数が少ない場合は，パスワード文の文字列が何回も繰り返すものとして処理を行う。また，平文の文字数よりパスワード文の文字数が多い場合は，パスワード文の途中までを使用する。

最終的に暗号文に変換された文字数は平文の２倍の数になる。

（例）平文として‘JKEN’，パスワード文として‘PAS’を与えた場合，出力する文字列は‘1A0A161E’となる。

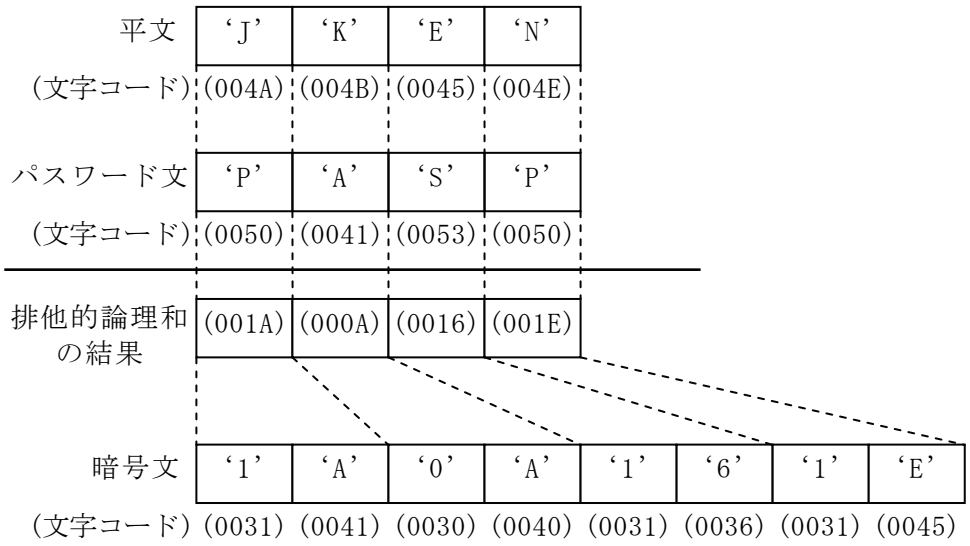


図１　実行例

（b）文字コード

ここで扱う文字コードは以下の表のとおりである。

表１　文字コード表

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
20	間隔	!	“	#	$	%	&	‘	(	)	*	+	,	−	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

＜設問１＞　暗号化と復号した文字列に関する次の記述中の［　　］に入るべき適切な字句を解答群から選べ。

平文が‘CASL’，パスワード文が‘@@@’の場合，暗号文は［(1)］である。また，暗号文が‘010203’，パスワード文が‘@@@’であれば，平文は［(2)］である。

(1) の解答群

ア．00010203　　イ．01040C1F

ウ．021E1C03　　エ．0301130C

(2) の解答群

ア．ABC　　イ．BCD

ウ．CDE　　エ．DEF

＜設問２＞　次の暗号化を行う副プログラム ANGO 中の［　　］に入るべき適切な命令を解答群から選べ。

［副プログラム ANGO の説明］

副プログラム ANGO へ渡されるパラメータは以下の形式となっており,このパラメータが格納されている領域の先頭番地を GR1 に格納してから ANGO が呼び出される。

(GR1)＋0	平文が格納されている連続領域の先頭番地を格納している
＋1	平文の文字数を格納している
＋2	パスワード文が格納されている連続領域の先頭番地を格納している
＋3	パスワード文の文字数を格納している
＋4	暗号文を格納する連続領域の先頭番地を格納している
＋5	暗号文の文字数を格納する

図２　ANGO へ渡されるパラメータ

また，ANGO の中で呼び出す副プログラム CVCHAR の仕様は次のとおりである。

機　　能	GR0 に格納された 0～15 の値を 16 進数表記の文字データに変換した結果を GR0 に格納する。							
受け取る値(GR0)	0	1	2	3	4	5	6	7
返す値(GR0)	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’	‘7’
受け取る値(GR0)	8	9	10	11	12	13	14	15
返す値(GR0)	‘8’	‘9’	‘A’	‘B’	‘C’	‘D’	‘E’	‘F’

［プログラム］

```
ANGO    START
        RPUSH
        LD      GR2, 0, GR1     ; 平文の先頭アドレスを取得
        LD      GR3, 2, GR1     ; パスワード文の先頭アドレスを取得
        LD      GR4, 4, GR1     ; 暗号文格納先の先頭アドレスを取得
        LD      GR5, =0         ; 平文のカウンタ初期設定
        LD      GR6, =0         ; パスワード文のカウンタ初期設定
LOOPA   LD      GR7, 0, GR2     ; 平文から１文字取り出す
        XOR     GR7, 0, GR3     ; 変換
        LD      GR0, GR7        ; 16 進数 2 桁のうち上位桁の値を変換
        [   (3)   ]
        CALL    CVCHAR
        ST      GR0, 0, GR4
        LD      GR0, GR7        ; 16 進数 2 桁のうち下位桁の値を変換
        [   (4)   ]
        CALL    CVCHAR
        ST      GR0, 1, GR4     ; 暗号文を格納
        ADDL    GR2, =1
        ADDL    GR3, =1
        [   (5)   ]
        ADDL    GR5, =1
        ADDL    GR6, =1
```

```
        CPA    GR6, 3, GR1      ; パスワード文の長さを超えたか
        JMI    SKIPA
               (6)
        LD     GR6, =0
SKIPA   CPL    GR5, 1, GR1      ; 平文の最後まで処理したかを判断
               (7)
        SUBL   GR4, 4, GR1      ; 暗号文の文字数を計算する
        ST     GR4, 5, GR1
        RPOP
        RET
        END
```

(3) の解答群

ア. **SLL GR0, 4**　　イ. **SLL GR0, 8**

ウ. **SRL GR0, 4**　　エ. **SRL GR0, 8**

(4) の解答群

ア. **AND GR0, =#000F**　　イ. **AND GR0, =#00F0**

ウ. **AND GR0, =#00FF**　　エ. **AND GR0, =#FF00**

(5) の解答群

ア. **ADDL GR4, =1**　　イ. **ADDL GR4, =2**

ウ. **ADDL GR4, 1, GR4**　　エ. **ADDL GR4, 2, GR4**

(6) の解答群

ア. **LD GR3, 0, GR1**　　イ. **LD GR3, 1, GR1**

ウ. **LD GR3, 2, GR1**　　エ. **LD GR3, 3, GR1**

(7) の解答群

ア. **JOV LOOPA**　　イ. **JZE LOOPA**

ウ. **JPL LOOPA**　　エ. **JMI LOOPA**

＜メモ欄＞