

1. 次のように配列 `validNumbers` にライセンス認証番号（整数 6 桁 100000~999999）のリストが格納されています。キーボードからライセンス認証番号を入力させ、番号リストに載っているかどうかを検査して、番号リストに載っている場合は「正規製品です」と出力するコードを作成しなさい。番号リストの終端はライセンス認証番号にない数値 `-1` とします。ヒント：while 文を用いて配列の終端まで（`-1` が出てくるまで）次々と if 文を用いて番号をチェックしていきます

```
int[] validNumbers={134324, 454323, 814823, 421914, 404347, 134552, 182276, 782818, 341807, 130404, -1};
```

2. 次のプログラムは配列 `array` に格納されている 10 個の整数を左右反転して配列 `rev_array` に入れるプログラムです。空欄を埋めてプログラムを完成させなさい。

ヒント：添え字を工夫して逆順になるようにコピーしましょう

（ソースプログラム）

```
class Assignment10_1
{
    public static void main(String[] args){
        int i;
        int[] array = {12,54,2,-7,30,75,-34,91,27,-62};
        int[] rev_array = new int[10];

        System.out.println("配列の左右を反転します");
        System.out.print("反転前の配列 ");
        for(i=0;i<10;i++)
            System.out.print(array[i]+" ");
        System.out.println();

        System.out.print("反転後の配列 ");
        for(i=0;i<10;i++)
            System.out.print(rev_array[i]+" ");
        System.out.println();
    }
}
```

（実行例）

配列の左右を反転します

反転前の配列 12 54 2 -7 30 75 -34 91 27 -62

反転後の配列 -62 27 91 -34 75 30 -7 2 54 12

3. 10個の整数 x_i をキーボードから入力してその最大値、最小値、平均、分散を求めるプログラムを作成しなさい。

ヒント：まずは10個の整数を配列に代入しましょう

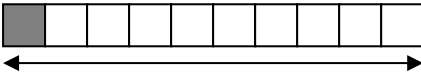
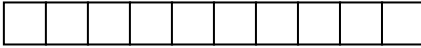
$$\text{(平均)} \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{(分散)} s^2 = \frac{1}{n} \sum_{i=1}^n (\bar{x} - x_i)^2 \quad n \text{ 整数のデータの数}$$

4. 配列に格納されている10個の整数の値を昇順に並べ替えるプログラムを作成しなさい。ここでは、配列を{12, 54, 2, -7, 30, 75, -34, 91, 27, -62}で初期化しましょう。

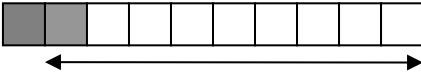
ヒント：2重の繰返し文を利用します

(選択ソートアルゴリズム)

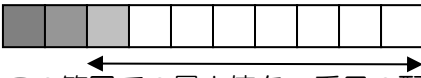
与えられた配列



この範囲での最小値を先頭の配列要素へ移動する(最小値と先頭の値を"交換"する)



この範囲での最小値を2番目の配列要素へ移動する(最小値と2番目の値を"交換"する)



この範囲での最小値を3番目の配列要素へ移動する(最小値と3番目の値を"交換"する)
以後、最後の配列要素まで繰り返すと配列要素は昇順に並び。

5. 2つの3次元ベクトルの内積とそれぞれの大きさを求めなさい。また、2つのベクトルの成す角度を求めなさい。ここでは、各ベクトルを配列で{1.0, 2.0, 3.0}、{3.0, 2.0, 1.0}として初期化します。

ヒント：まずはベクトルのままで計算式を考えてみましょう

(平方根、三角関数を求める関数)

```
double ans, a, rad;
```

```
ans = Math.sqrt(a);
```

値 a の平方根

```
ans = Math.sin(rad);
```

角度 rad(ラジアン)の正弦値

```
ans = Math.cos(rad);
```

角度 rad(ラジアン)の余弦値

```
ans = Math.asin(a);
```

正弦値 a となる角度(ラジアン)

```
ans = Math.acos(a);
```

余弦値 a となる角度(ラジアン)

ただし、角度の単位はラジアン

$$\text{(ベクトルの内積)} \vec{X} \cdot \vec{Y} = \sum_{i=1}^n x_i y_i = |\vec{X}| |\vec{Y}| \cos(\theta) \quad n, \theta \text{ ベクトルの要素数と成す角度}$$

$$\text{(ベクトルの大きさ)} |\vec{X}| = \sqrt{\vec{X} \cdot \vec{X}}$$

6. 下のように初期化された 3 つの配列において、

```
int[] num1 = {1, 4, 0, 3, 2, 5};
```

```
int[] num2 = {2, 4, 1, 3, 2, 1};
```

```
int[] add = {0, 0, 0, 0, 0, 0};
```

配列 num1 と配列 num2 の各配列要素を足し、その結果を配列 add の同じ場所の配列要素に代入しなさい。例えば、

```
num1[3] + num2[3] → add[3]
```

さらに、配列 add の各配列要素を画面に出力しなさい。

ヒント：配列要素ごとに前から順番に計算していきましょう

(実行例)

2 つの配列を足し算します

3 8 1 6 4 6

7. 1 から 5 の 5 段階評価でアンケートを 16 名の学生に対して行った。16 名分のアンケート結果は以下のものであった。

1, 3, 5, 2, 1, 3, 4, 1, 1, 2, 3, 4, 5, 3, 3, 2

このアンケート結果を用いて次のように配列を初期化し、下の集計の形式で出力しなさい。

```
int[] data = {1,3,5,2,1,3,4,1,1,2,3,4,5,3,3,2};
```

ヒント：選択肢毎にカウント用の変数を準備して個数を数えましょう

(集計形式)

アンケートを集計します

選択肢 1: 0件

選択肢 2: 0件

選択肢 3: 0件

選択肢 4: 0件

選択肢 5: 0件

(実行例)

アンケートを集計します

選択肢 1: 4 件

選択肢 2: 3 件

選択肢 3: 5 件

選択肢 4: 2 件

選択肢 5: 2 件

8. int 型の配列要素を 50 個もつ配列 array_now と配列 array_next を宣言し、以下のコードを用いて配列 array_now のみを 0 と 1 のパターンで初期化しなさい。中央付近に 1 が 1 つ、その他は 0 とします。

```
int[] array_now=new int[50];
int[] array_next=new int[50];
for(int i=0;i<50;i++){
    if(i==25)array_now[i]=1;
    else array_now[i]=0;
}
```

そして、配列 array_now に下に示す書き換え規則を適用し、その結果を配列 array_next に保存しなさい。i 番目の配列要素 array_next[i]には、array_now[i-1]と array_now[i]、array_now[i+1]を参照して書き換え規則に照らし合わせ対応する 0 または 1 を代入します。

000	001	010	011	100	101	110	111
↓	↓	↓	↓	↓	↓	↓	↓
0	1	0	1	1	0	1	0

両端の配列要素 array_next[0]と array_next[49]については周期的境界条件を用います。

- array_now[49]、array_now[0]、array_now[1]を参照し、array_next[0]に値を代入
- array_now[48]、array_now[49]、array_now[0]を参照し、array_next[49]に値を代入

配列 array_next のすべての配列要素に代入が終わったら、この配列を配列 array_now にコピーし戻し、再度書き換え規則を適用して配列 array_next に結果を代入します。

これを 24 回繰り返します。

この繰り返しの度に配列 array_now のパターンを 1 行に表示しなさい。このとき、0 を空白 ' ' として、1 を '*' として表示すると<<素敵なパターン>>が描き出されます。

9. 福岡県の1年間の降水量(mm)を月ごとに棒グラフで表示しなさい。1月から12月までの各月の降水量(mm)は以下のようであった。

月	1	2	3	4	5	6	7	8	9	10	11	12
降水量 mm	68	71	112	116	142	254	277	172	178	73	84	59

この降水量を用いて次のように配列を初期化し、下のグラフのように10mm単位で■を表示します。例えば、68mmは■が6個で、7mmは0個です。

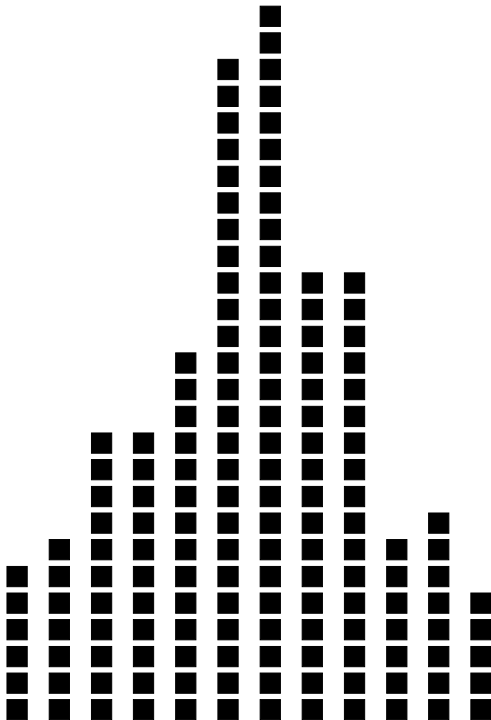
```
int[] rainfall={68,71,112,116,142,254,277,172,178,73,84,59};
```

ヒント：まず、最大の降水量を探しましょう。次に、この最大の降水量を超えない、10mm単位の最大のボーダーラインを計算します。たとえば、最大の降水量が156mmとすると、10mm単位の最大のボーダーラインは150mmです。

```
int border = 10*(max/10); // 変数 max [int 型] は最大の降水量です
```

最後に、グラフを上から下に向かって出力していきます。降水量が上で設定したボーダーライン以上の月に■を表示し、ボーダーラインを10mm下げて0mmになるまでこれを繰り返します。

(実行例)



10. 石とりゲームを作成しましょう。まず、3つの石の山があります。最初それぞれの山には5つの石が積まれています。二人のプレイヤーが交互に、好きな山から1~3個の好きな数だけ必ず石を取ります。最後に石を取らなかったプレイヤーの勝ちです。

ヒント：3つの石の山を int 型の配列で管理しましょう。それぞれの配列要素に代入される整数がその山の石の数に対応します。例えば、

```
int[] stones=new int[3];
```


とした場合、stones[0]は最初の山にある石の数を表します。


(実行例)

1: ■■■■■■

2: ■■■■■■

3: ■■■■■■


〔プレイヤー1〕石を取る山(1-3)を入力してください?1 


〔プレイヤー1〕取る石の数(1-3)を入力してください?3 

1: ■■

2: ■■■■■■

3: ■■■■■■


〔プレイヤー2〕石を取る山(1-3)を入力してください?2 


〔プレイヤー2〕取る石の数(1-3)を入力してください?2 

1: ■■

2: ■■■■

3: ■■■■■■


〔プレイヤー1〕石を取る山(1-3)を入力してください?3 


〔プレイヤー1〕取る石の数(1-3)を入力してください?3 

1: ■■

2: ■■■■

3: ■■


〔プレイヤー2〕石を取る山(1-3)を入力してください?1 


〔プレイヤー2〕取る石の数(1-3)を入力してください?2 

1:

2: ■■■■

3: ■■


〔プレイヤー1〕石を取る山(1-3)を入力してください?2 


〔プレイヤー1〕取る石の数(1-3)を入力してください?3 

1:

2:

3: ■■


〔プレイヤー2〕石を取る山(1-3)を入力してください?3 


〔プレイヤー2〕取る石の数(1-3)を入力してください?1 

1:

2:

3: ■

〔プレイヤー1〕石を取る山(1-3)を入力してください?3 

〔プレイヤー1〕取る石の数(1-3)を入力してください?1 

プレイヤー2の勝ちです♪