# Contents



# 1 chapter practice

## 1.1 Cramer

```java
/*
請利用克萊姆(Cramer)公式，嘗試解出2*2的線性方程式
ax + by = e
cx + dy = f

輸入：
輸入a b e
與 c d f 分別代表兩個線性方程式的常數與答案
其形態皆為double

輸出：
請利用Cramer解出x與y二值
其形態皆為double
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        double x1 = 0, y1 = 0, a1 = 0, x2 = 0, y2 = 0, a2 = 0;

        x1 = input.nextDouble();
        y1 = input.nextDouble();
        a1 = input.nextDouble();
        x2 = input.nextDouble();
        y2 = input.nextDouble();
        a2 = input.nextDouble();

        // 克萊姆公式
        // [a b][x] = [e]
        // [c d][y] = [f]
        // x = ed - bf / ad - bc
        // y = af - ec / ad - bc
        double x = 0, y = 0;
        x = ( ( a1 * y2 ) - ( y1 * a2 ) ) / ( ( x1 * y2 ) - ( y1 * x2 ) );
        y = ( ( x1 * a2 ) - ( a1 * x2 ) ) / ( ( x1 * y2 ) - ( y1 * x2 ) );
        System.out.println(x + " " + y);
    }
}
```

## 1.2 calculation

```java
/*
請撰寫一個程式，讀取0-1000其中一位數，然後將其的每一位數相乘。

輸入
為一介於0<= num <=1000的Integer

輸出
形態為Integer
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();

        // 先把字串內的每位數單獨拆分進 array 內
        String[] array = str.split("");

        Integer ans = new Integer(1);
        for( int i = 0; i < str.length(); i++ ){

            // 用 Integer.parseInt() 把 array[i] 內的字符轉數字
            ans = Integer.parseInt(array[i]) * ans;
        }

        System.out.println(ans);
    }
}
```

## 1.3 distance

```java
/*
請撰寫一個程式，提示使用者輸入兩點座標(x1,y1)及(x2,y2)，並顯示兩
    請注意，您可以利用Math.pow(a,0.5)來計算√a。

輸入
x1,y1: 1.5 -3.4
x2,y2: 4 5

輸出
8.764131445842194

計算兩點間距離的公式為
(x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) 開根號
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
```

```
        String str = input.nextLine();

        // 先把字串內的每位數單獨拆分進 array 內
        String[] array = str.split("");

        Integer ans = new Integer(1);
        for( int i = 0; i < str.length(); i++ ){

            // 用 Integer.parseInt() 把 array[i] 內的字符轉數字
            ans = Integer.parseInt(array[i]) * ans;
        }

        System.out.println(ans);
    }
}
```

## 1.4 Heron's formula

```
/*
請撰寫一程式，提示使用者輸入三角形的三個頂點座標(x1,y1)、(x2,y2)、(x3,y3)，接著顯示該三角形的面積。
s = (a + b + c) /2
A = √ s(s - a)(s - b)(s - c)

輸入
三角形三個點: 1.5 -3.4 4.6 5 9.5 -3.4
輸出
面積為33.600

輸出格式為小數點後三位
若三點不為三角形請輸出"Not Triangle"
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    private static Math Precision;

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        double x1 = 0, y1 = 0, x2 = 0, y2 = 0, x3 = 0, y3 = 0;

        x1 = input.nextDouble();
        y1 = input.nextDouble();
        x2 = input.nextDouble();
        y2 = input.nextDouble();
        x3 = input.nextDouble();
        y3 = input.nextDouble();

        double[] array = new double[3];
        array[0] = Math.sqrt( Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2) );
        array[1] = Math.sqrt( Math.pow(x3 - x2, 2) + Math.pow(y3 - y2, 2) );
        array[2] = Math.sqrt( Math.pow(x3 - x1, 2) + Math.pow(y3 - y1, 2) );

        // 判斷是否為三角形
        // Arrays.sort(陣列名稱) -> 自動排序
        Arrays.sort(array);

        if ( array[0] + array[1] <= array[2] ){
            System.out.println("Not Triangle");
        }
        else{
            // 海龍公式
            double s = 0;
            for( int i = 0; i < 3; i++ ){
                s = s + array[i];
            }
            s = s / 2;

            double a = 0;
            a = Math.sqrt( s * (s-array[0]) * (s-array[1]) * (s-array[2]) );

            // 利用 BigDecimal 去取小數點並四捨五入
            BigDecimal ans = new BigDecimal(a);
            System.out.println( ans.setScale(3, RoundingMode.HALF_UP));
        }


    }
}
```

## 1.5 year month day

```
/*
請撰寫一程式，提示使用者輸入月份與年份，接著顯示該月份的天數。
    舉個例子，如果使用者月份輸入2，年份輸入2012，此程式應顯示"February 2012 had 29 days"。如果使用者月份輸入3，年份輸入2015，程式顯示"March 2015 had 31 days"。
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int month = input.nextInt();
        int year = input.nextInt();

        // 分別存 30 天的月份 和 31 天的月份 以及他們的英文月份名
        int[] thirty = {1,3,5,7,8,10,12};
        String[] thirty_eng = {"January", "March", "May", "July", "August", "October", "December"};

        int[] thirty_one = {4,6,9,11};
        String[] thirty_one_eng = {"April", "June", "September", "November"};

        // 查詢他是否是 30 天的月份
        // 最後放 return 後面的其他程式碼就不會執行了
        for( int i = 0; i < thirty.length; i++){
            if(month == thirty[i]){
                System.out.println(thirty_eng[i] + " " + year + " had 31 Days" );
                return;
            }
        }
        for( int i = 0; i < thirty_one.length; i++){
            if(month == thirty_one[i]){
                System.out.println(thirty_one_eng[i] + " " + year + " had 31 Days" );
                return;
            }
        }
        if( year % 4 == 0){
            System.out.println("February " + year + " had 29 Days");
            return;
        }
        System.out.println("February " + year + " had 28 Days");

    }
}
```

## 1.6 circle distance

```
/*
請撰寫一個程式，提示使用者輸入一個點座標(x,y)，接著檢視該點是否在半徑為 10 的圓內，
計算距離的公式

(x*x + y*y) 開根號

輸入
4 5
9 9

輸出
(4.0,5.0) is in the circle
(9.0,9.0) is not in the circle
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double x = input.nextDouble();
        double y = input.nextDouble();

        //是否在半徑 10 以內
        if( Math.sqrt( Math.pow(x, 2) + Math.pow(y, 2) ) <= 10 ){
            // 如果要輸出 ("%f %d", a, b) 要用 -> System.out.printf
            // %.1f 控制小數點後 1 位
            System.out.printf("(%.1f,%.1f) is in the circle" ,x ,y);
        }
        else{
            System.out.printf("(%.1f,%.1f) is not in the circle" ,x ,y);
        }
        System.out.println();
    }
}
```

## 1.7 Regular polygon area

```
/*
正多邊形為帶有n個邊，各邊等長，各角度也相同的多邊形(也就是
A = (n*s^2) / 4*tan(pi/n)

請撰寫一程式，使用者輸入邊數，以及正多邊形的邊長，接著顯示

輸入
n=邊數: 5
s=邊長: 6.5

輸出
74.69017017488385
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        double n = input.nextDouble();
        double s = input.nextDouble();

        // java 有內建 tan 和 PI 在 Math 裡 太酷了ㄅ
        System.out.println( (n*s*s) / (4*Math.tan(Math.PI/n)) );
    }
}
```

## 1.8 binary to decimal

```
/*
請撰寫一程式，使用者輸入二進位數字，然後印出其對應的十進位數字。
    輸入 0111 1001 1100 輸出 7 9 12
*/

import java.util.Scanner;
import java.util.StringTokenizer;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        String bin = input.nextLine();
        StringTokenizer st = new StringTokenizer(bin, " " );

        while(st.hasMoreTokens()){
            int two = 1, deci = 0;
            String str = st.nextToken();
            for( int i = str.length()-1; i >= 0; i--){
                deci += ( str.charAt(i) - '0' )*two;
                two *= 2;
            }
            System.out.println(deci);
        }

    }
}
```

## 1.9 substring

```
/*
請撰寫一程式，使用者輸入二個字串，然後顯示二個字串是否為第一個字
輸入 string s1:ABCD string s2:BC
輸出 BC is a substring of ABCD

輸入 string s1:ABCD string s2:CA
輸出 CA is not a substring of ABCD
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        String str = input.nextLine();

        // 已確認 : 切割後的空白不會存進 array
        String[] array = str.split(" ");

        // 原來可以用 str.contains 去找子字串
            我還以為只能找字元..
        if(array[0].contains(array[1])){
            System.out.println(array[1] + " is a substring of " + array[0]);
        }
        else{
            System.out.println(array[1] + " is not a substring of " + array[0]);
        }

    }
}
```

## 1.10 prime or gcd

```java
/*
請撰寫一程式，讀取一個整數，並以遞增的順序顯示所以的因數
    比方說，假設輸入整數為120，此程式的輸出結果應該是要2
但當輸入的整數為質數時，需輸出1與輸入的質數。
此題假設1為質數
*/

import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String[] args) {
        boolean[] prime = new boolean[10000000+5];
        Scanner input = new Scanner(System.in);

        int n = input.nextInt();

        for( int i = 2; i <= n; i++ ){
            if(!prime[i]){
                for( int j = i * i; j <= n; j += i ){
                    prime[j] = true;
                }
            }
        }

        if( !prime[n] ){
            System.out.println("1 " + n );
        }
        else{
            int num = n;
            for( int i = 2; i < n; i++ ){
                while( num % i == 0 ){
                    num /= i;
                    System.out.print( num != 1 ? i +
                        " " : i + "\n" );
                }
                if( num == 1 ){
                    break;
                }
            }
        }
    }
}
```

## 1.11 pi calculation

```java
/*
pi = 4( 1 + 1/3 - 1/5 + 1/7 +... + (-1)^i+1^/2*i-1)
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        double pi = 0;
        for( int i = 1; i <= n; i++ ){
            // 依據題意
            pi += Math.pow(-1.0, (double)i+1)/(2*i-1);
        }
        System.out.println(pi*4);

    }
}
```

## 1.12 factorial

```java
/*
e = 1/0! + 1/1! + 1/2! + ... 1/n!
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        // 設個陣列存每階的值
        double[] e = new double[10000000+5];
        e[0] = 1;
        Scanner input = new Scanner(System.in);
        long n = input.nextLong();

        double ans = 1;
        for( long i = 1; i <= n; i++ ){
            // i! = i*(i-1)!
            e[(int)i] = i * e[(int)i-1];
            ans += 1/e[(int)i];
        }
        System.out.println(ans);
    }
}
```

## 1.13 Almanac

```java
/*
請撰寫一程式，提示使用者輸入年份及該年份第一天為一星期的哪一天，
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int y, week, k = 0;
        String[] month = {"January", "February",
            "March", "April", "May", "June", "July",
            "August", "September", "October",
            "November", "December"};
        int[] days = {31, 28, 31, 30, 31, 30, 31, 31,
            30, 31, 30, 31};

        y = input.nextInt();
        week = input.nextInt();

        boolean flag = false;
        for( int i = 0; i < 12; i++ ){
            System.out.println( "          " +
                month[i] + " " + y + "\n" +
                    "-----------------------------\n"
                        + " Sun Mon Tue Wed Thu Fri
                        Sat");
            k = 0;
            for( int j = 1; j <= days[i]; k++){
                if( !flag && k != week ){
                    System.out.print("    ");
                    continue;
                }

                flag = true;
                if( k <= 6 ){
                    System.out.printf("%4d", j++);
                }
                else{
                    System.out.printf("\n%4d", j++);
                    k = 0;
                }
            }
            flag = false;
            if( k > 6 ){
                k = 0;
            }
            week = k;
            System.out.print( i!= 11 ? "\n\n" : "\n");
```

```
        }
    }

}
```

## 1.14 perfect number

```
/*
將一正整數的所有公因數加總，其總和與正整數相等，則稱此數為
    課本練習５・33
本題不需任何輸入，請嘗試輸出所有 0<num< 10000的完美數
*/

import java.util.Vector;

public class Main {

    public static void main(String[] args) {

        // 用 vector 存因數
        Vector<Integer> v = new Vector<Integer>();
        for( int i = 1; i < 10000; i++ ){
            for( int j = 1; j < i; j++ ){
                if( i % j == 0 ){
                    v.add(j);
                }
            }
            // 加總公因數
            int count = 0;
            for (Integer integer : v) {
                count += integer;
            }
            // 是否為完美數
            if( count == i ){
                System.out.println(i);
            }
            v.clear();
        }
    }
}
```

## 1.15 maximum appear times

```
/*
請撰寫一程式，讀取一連串的整數，找出其最大值，並計算該值出現的次數
*/

import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Vector<Integer> num = new Vector<Integer>();

        // 用 vector 來放輸出
        while(input.hasNextInt()){
            int n = input.nextInt();
            num.add(n);
        }
        // 找最大值和出現次數
        int maxi = 0, count = 1;
        for (Integer integer : num) {
            if (integer > maxi) {
                count = 1;
                maxi = integer;
            } else if (integer == maxi) {
                count++;
            }
        }
        System.out.println(maxi + " " + count);
    }
}
```

## 1.16 leap year

```
/*
撰寫一程式，提示使用者輸入一年份，然後判斷它是否為閏年，當輸入-
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        while(input.hasNextInt()){
            int n = input.nextInt();
            if( n < 0 ){
                break;
            }
            // 閏年的規律
            else if( n % 4 == 0 && n % 100 != 0){
                System.out.println("It is leap
                    year.");
            }
            else {
                if (n % 400 == 0) {
                    System.out.println("It is leap
                        year.");
                } else {
                    System.out.println("It is not
                        leap year.");

                }
            }
        }
    }
}
```

# 2 exam

## 2.1 BigDecimal Calculator

```
/*
Problem
Write a program to input an expression string in
    which the operands and operators are separated by
    zero or more spaces. For example, 3.5*4+3 and 3.5
    + 4 % 3 are acceptable expressions. The operator
    in the expression might be +, -, *, /, and %.
    Your program must print out the expression and
    its computing result. The sample output for the
    input expression 3.5*4+3 is shown below:
3.5 * 4 + 3 = 17

Requirement
Write a static method BigDecimal calculate(String
    exp) to compute the expression and return a
    BigDecimal result. The operands should be stored
    as BigDecimal in this method. You have to use the
    arithmetic operators provided by the BigDecimal
    class to calculate the expression. (未依規定，以
    0 分計)

Input
There are many input lines. Each line has an input
    expression Exp. There are three operands and two
    operators in Exp.

Output
```

```java
/*
For each input expression Exp, please output the
    expression and its computing result. Note that
    all tokens are separated by a space character.
    (小數點以下印一位)
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Arrays;
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main {

    public static BigDecimal calculate(String exp){

        StringTokenizer st = new
            StringTokenizer(exp,"0123456789. ");
        StringTokenizer ma = new StringTokenizer(exp,
            "+-*/% ");

        String[] operator = new String[5];
        String[] digits = new String[5];
        Arrays.fill(operator, "0");
        Arrays.fill(digits, "0");

        int i = 0;
        boolean flag = false;
        while( st.hasMoreTokens() ){

            if( exp.charAt(0) == '-' && !flag ){
                String haha = st.nextToken();
                flag = true;
                continue;
            }
            else{
                operator[i] = st.nextToken();
                i++;
            }

        }
        int j = 0;
        while( ma.hasMoreTokens() ){
            digits[j] = ma.nextToken();
            j++;
        }

        boolean e = false;
        for( int k = 0; k < j-1; k++ ){
            if(exp.charAt(0) == '-' ||
                operator[k].equals("0") ){
                break;
            }
            System.out.print(digits[k] + " ");
            System.out.print(operator[k] + " ");
            e = true;
        }
        if( e ){
            System.out.print(digits[j-1] + " = ");
        }

        boolean yep = false;
        if( i == 2 ){
            if( operator[1].equals("*") ||
                operator[1].equals("/")  ||
                operator[1].equals("%") ){
                if( operator[0].equals("+") ||
                    operator[0].equals("-") ) {
                    yep = true;
                    String tp;
                    tp = operator[1];
                    operator[1] = operator[0];
                    operator[0] = tp;

                    String str;
                    str = digits[1];
                    digits[1] = digits[0];
                    digits[0] = str;

                    str = digits[2];
                    digits[2] = digits[1];
                    digits[1] = str;
                }

            }
        }
        BigDecimal sum = new BigDecimal(digits[0]);
        if( exp.charAt(0) == '-'){
            BigDecimal tmp = BigDecimal.ZERO;
            sum = tmp.subtract(sum);

        }


        i = 0;

        for( int k = 1; k < j; k++ ){
            BigDecimal b = new BigDecimal(digits[k]);
            if( operator[i].equals("+") ){
                sum = sum.add(b);
            }
            else if( operator[i].equals("-") ){
                if(yep){
                    sum = b.subtract(sum);
                }
                else{
                    sum = sum.subtract(b);
                }
            }
            else if( operator[i].equals("*") ){
                sum = sum.multiply(b);
            }
            else if( operator[i].equals("/") ){
                sum = sum.divide(b, 1 ,
                    RoundingMode.CEILING);
            }
            else if( operator[i].equals("%") ){
                sum = sum.remainder(b);
            }
            i++;
        }

        return sum;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        while( input.hasNext() ){
            String exp = input.nextLine();
            BigDecimal ans = calculate(exp);
            // BigDecimal 四捨五入的寫法 .setScale(
                想要留取的位數, RoundingMode.HALF_UP)
            System.out.println(ans.setScale(1,
                RoundingMode.HALF_UP));
        }
    }
}
```

## 2.2 Magic Square

```java
/*
Problem
If you have good observations skills, you may found
    that building a Magic Square is simple. A Magic
    Square has only an odd number N of rows and
    columns where N < 100. A Magic Square is created
    by integer numbers in the range from 1 to N2 ,
    with a peculiar property, the “sum of the
    numbers” in each row, column and diagonal are
    the same.

For the case n = 3,
```

```
M. Square          Rows            Columns         Diagonals
4 9 2              4+9+2 = 15      4+3+8 = 15      2+5+8 = 15
3 5 7              3+5+7 = 15      9+5+1 = 15      4+5+6 = 15
8 1 6              8+1+6 = 15      2+7+6 = 15

Input
Each line contains an Integer N denoting an N * N Magic Square.

Output
如果 N 是偶數則輸出 " It is not an odd number. " 。
如果 N 是奇數則先輸出直橫列的加總數字，再輸出 N * N 數字陣列,每個數字以%5d 格式輸出。每組測資間請空一行。
*/

import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        boolean flag = false;

        while(input.hasNext()){
            int n = input.nextInt();
            if( flag ){
                System.out.println();
            }

            if( n % 2 == 0 ){
                System.out.println("It is not an odd number.");
                flag = true;
                continue;
            }

            int[][] square = new int[100+5][100+5];
            for( int i = 0; i < n; i++ ){
                Arrays.fill( square[i], 0);
            }

            int r = n-1, c = n/2;
            square[r][c] = 1;
            for( int i = 2; i <= n*n; i++ ){

                int rn = r+1;
                int cn = c+1;
                if( rn >=n ){
                    rn = 0;
                }
                if( cn >= n ){
                    cn = 0;
                }
                if( square[rn][cn] != 0 ){
                    rn = r-1;
                    cn = c;
                    if( rn < 0 ){
                        rn = n-1;
                    }
                }
                square[rn][cn] = i;
                r = rn;
                c = cn;
            }
            int sum = 0;
            for( int i = 0; i < n; i++ ){
                sum += square[0][i];
            }
            System.out.println(sum);
            for( int i = 0; i < n; i++ ){
                for( int j = 0; j < n; j++ ){
                    System.out.printf("%5d",
                        square[i][j]);
                }
                System.out.println();
            }
            flag = true;
        }

    }
}
```

## 2.3 Prime Factorization

```
/*
Problem
輸入數字 N (資料型態為 Integer)，
    請輸出該數字的所有質因數及其次方。例如 N=360=23*32 *5。此題數字可能會有質數出現。

Requirement
請撰寫以下兩個 static methods： (未依規定，以 0 分計)

1. boolean [] PrimeArray(long N) {….}
    which returns an array A of Boolean values, where
        A[i] is true if i is a prime number,
        otherwise, A[i] is false if i is not a prime
        number. Note that A.length = N+1; Hint: if n
        is a prime number, then n * j is not a prime,
        where j >= 2;

2. String PrimeFactorization(long N) {…}
    which returns a string of prime factorization for
        the number N. For example, if N = 360, the
        returned string is "2^3 * 3^2 * 5" .

Input
輸入有多列，每列有個整數 N，最多 1000 列。

Output
第一行輸出所有數字中之最大數X及其開根號整數X，其後針對每一組測資輸出 N的質因數分解，將數字
N的所有質因數（及其次方）以小到大方式顯示出來，如質因數之次方數大於 1，以^運算符號顯示，不同質因數間以 * 運算符號互相連接， *運算符號前 後加空格。
*/

import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static boolean [] PrimeArray(long N){

        boolean[] prime = new boolean[1000000];
        Arrays.fill(prime, true);

        for( long i = 2; i <= N; i++ ){
            if(prime[(int)i]){
                for( long j = i * i; j <= N; j += i ){
                    prime[(int)j] = false;
                }
            }
        }
        return prime;
    }
    public static String PrimeFactorization(long N){

        int[] num = new int[1000000];
        Arrays.fill( num, 0 );

        long count = N;
        for( int i = 2; i < N; i++ ){
            while( count % i == 0 ){
                num[i]++;
```

```
                count /= i;
            }
            if( count == 1 ){
                break;
            }
        }

        // 因數轉字串處理
        String[] tp = new String[1000000];
        int j = 0;
        for( int i = 2; i < N; i++ ){
            if( num[i] != 0 ){
                tp[j++] = Integer.toString(i);
                if( num[i] != 1 ){
                    tp[j++] = "^";
                    tp[j++] =
                        Integer.toString(num[i]);
                }
                tp[j++] = " * ";
            }
        }

        // 合併字串 用 str1.concat(str2)
        // 因為最後會多一組 * 所以j-1
        String result = "";
        for( int i = 0; i < j-1; i++ ){
            result = result.concat(tp[i]);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int max = 0;
        int[] N = new int[1000+5];
        int k = 0;

        while( input.hasNextLong() ) {

            N[k] = input.nextInt();
            if (N[k] > max) {
                max = N[k];
            }
            k++;
        }


        // 可以用 method 回傳陣列
            但也要用陣列接他的回傳值
        boolean[] A = PrimeArray(max);
        System.out.println(max + " " +
            (int)Math.sqrt(max) );
        for( int o = 0; o < k; o++ ){

            if( A[N[o]] ){
                System.out.println(N[o]);
            }
            else{
                String factorization =
                    PrimeFactorization(N[o]);
                System.out.println(factorization);
            }
        }
    }
}
```

# 3 homeworks

## 3.1 Print Prime Numbers

```
/*
Write a program to input an integer N and print all
    prime numbers less than N.
Please print 10 prime numbers in each line.
*/

import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        // 建質數表
        // java 中沒有全域變數
            所以這邊建表目前是每次輸入都重建一次
            但不夠漂亮
        boolean[] prime = new boolean[10000000+5];

        // 用 vector 存質數們
        Vector<Integer> v = new Vector<Integer>();
        for( int i = 2; i <= n; i++ ){
            if(!prime[i]){
                v.add(i);
                for( int j = i * i; j <= n; j += i ){
                    prime[j] = true;
                }
            }
        }

        for( int i = 0, k = 0; i < v.size(); i++, k++
            ){

            if( k == 9 || i == v.size() - 1){
                System.out.println(v.get(i));
                // 把 k 歸回去 -1 是因為下次輪迴會先
                    k++
                k = -1;
            }
            else{
                System.out.print(v.get(i) + " ");
            }
        }
        v.clear();
    }
}
```

## 3.2 P1 Infix Notation to Postfix Notation

```
/*
為了要便於中序轉後序的程式運行，請先將算式的數字與運算符號做分割
1. 使用String類別 & 字串輸入
2. 使用StringTokenizer類別 或
    String類別之split()method來切割中序運算式。
3. 使用String [] array 儲存所有tokens。
4. 將數字token轉換成實數，加總後輸出總和。Hint: Use
    BigDecimal in java.math package

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%，算式中數

輸出
第一行輸出運算式的每個運算元，運算元間以空格字元隔開。
第二行輸出運算式的每個運算子，運算子間以空格字元隔開。
第三行印出運算元的總和，小事點以下印三位。
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
```

```
        String str = input.nextLine();
        String token = "+-*/()%";

        // 當字串中帶有 + * | \ 等符號時要寫 \\
            轉義，因為他們在正則表達示中有相應的不同意義
        // limit is 0; array contains all substrings
        // Positive Lookahead or Lookbehind 觀念
        // “((?=@)|(?<=@))” -> 才會把 運算子 還有
            運算元 都單獨切開
        String[] array =
            str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|

        // 預設 ans 賦值 可控制小數點位數 ex 0.000 ->
            小數點後三位
        BigDecimal ans = new BigDecimal("0.000");

        boolean flag = true;
        for( String check : array){
            // string.contains
                一次查詢多個關鍵字的寫法
                要另外存一個數組 token
            // 且在查詢時要以此 for ( String check :
                array ) 去做掃描
            if( !token.contains(check) ){
                // 將字串型態的數字轉成實數並相加
                BigDecimal b = new BigDecimal(check);
                // (boolean 判斷式) ? (true的輸出) :
                    (false 的輸出)
                System.out.print( flag ? check : " "
                    + check);
                flag = false;

                // BigDecimal 內建 .add 可以直接相加
                ans = ans.add(b);
            }
        }
        System.out.println();

        flag = true;
        for( String again : array){
            if( token.contains(again) ){
                System.out.print( flag ? again : " "
                    + again);
                flag = false;
            }
        }
        System.out.println();

        // BigDecimal 四捨五入的寫法 .setScale(
            想要留取的位數, RoundingMode.HALF_UP)
        System.out.println(ans.setScale(3,
            RoundingMode.HALF_UP));
    }
}
```

## 3.3 P2 Infix Notation to Postfix Notation

```
/*
請利用 P1
    中序轉後序-分割的答案輸出後續的算式，並輸出後序運算式

程式要求
1.必須使用java.util.Stack 實作，否則不予計分。
2.請利用part1求得的token 字串陣列轉後序

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%，算

輸出
將後序算式的每個運算元與運算子後加入一個空格字元印出。
格式請詳看sample output
*/
```

```
import java.util.Scanner;
import java.util.Stack;

public class Main {

    public static int icpfunc(String check){
        // icp 是該 token 的順序
        // icp -> +- 1 -> */% 2 -> ( 4
        int icp;
        if(check.equals("(")){
            icp = 4;
            return icp;
        }
        else if(check.equals("*") ||
            check.equals("/") || check.equals("%") ){
            icp = 2;
            return icp;
        }
        else{
            icp = 1;
            return icp;
        }
    }
    public static int ispfunc(String check){
        // isp 是堆疊最上層 stack.peek 中的順序
        // isp -> ( 0 -> +- 1 -> */% 2
        int isp;
        if(check.equals("(")){
            isp = 0;
            return isp;
        }
        else if(check.equals("*") ||
            check.equals("/") || check.equals("%") ){
            isp = 2;
            return isp;
        }
        else{
            isp = 1;
            return isp;
        }
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();
        String token = "+-*/(%";

        // 用 split 切割字串
        String[] array =
            str.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*

        // 使用 stack 儲存運算子
        Stack<String> stack = new Stack<>();
        int icp = 0, isp = 0;

        for (String s : array) {
            // 如果是 ) 則輸出所有 ( 以前的 stack
                內的運算子
            // 但 () 都不輸出
            if (s.equals(")")) {
                while (!stack.peek().equals("(")) {
                    System.out.print(stack.peek() + "
                        ");
                    stack.pop();
                }
                // 把 ) 也 pop 掉
                stack.pop();
            }
            // 如果是 + - * / %
            else if (token.contains(s)) {
                // 運算此運算子的 icp 和 當前 stack
                    頂端的運算子的 isp
                icp = icpfunc(s);
                if (stack.empty()) {
                    stack.push(s);
                }
```

```
                else {
                    isp = ispfunc(stack.peek());
                    if (icp > isp) {
                        stack.push(s);
                    } else {
                        while (icp <=
                            ispfunc(stack.peek())) {
                            System.out.print(stack.peek()
                                + " ");
                            stack.pop();
                            if (stack.empty()) {
                                break;
                            }
                        }
                        stack.push(s);
                    }
                }
            }
            // 如果是運算元則直接輸出
            else {
                System.out.print(s + " ");
            }
        }
        // 最後因為有可能 array
            在切割的時候最後一個值是運算元
        // 那樣就還會有運算子在 stack 還沒輸出
            所以這裡在確認一次
        while(!stack.empty()){
            System.out.print(stack.peek() + " ");
            stack.pop();
        }
        System.out.println();
    }
}
```

## 3.4 P3 Infix Notation to Postfix Notation

```
/*
請利用作業一(Part
    2)中序轉後序-轉後序的答案輸出後續的算式，並輸出運算結果

輸入
每組測資會給予不一定長度的算式，符號包含+,-,*,/,(,),%。

程式要求 (未依規定以0分記)
Write a Class XXXX (name it yourself) (which is not
    the Main class)  for computing the input
    expression.
Class XXXX must include the following three static
    methods
  String[] AAAA(String s) which returns an array of
    string tokens for the input expession s.
String[] BBBB(String s) which returns an array of
    string tokens (Postfixe Expression) for the input
    expression s. Note that BBBB() can call AAAA().
BigDecimal CCCC(String s) which returns the computing
    result of input expresdion s. Note that CCCC()
    can call AAAA() and BBBB().
Please give a meaningful name for AAAA, BBBB, and
    CCCC.

輸出
輸出算式的答案(BigDecimal)
*/

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;
import java.util.Stack;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String str = input.nextLine();

        BigDecimal output;
        // 呼叫並使用另一個叫做 computing 的 class
            的內部 method
        output = computing.caculating(str);

        System.out.println(output.setScale(2,
            RoundingMode.HALF_UP));
    }
}
// 同一份 java 檔案內只能有一個 public class
    所以第二個直接叫 class 就好
class computing {

    // 用來判斷後續先後順序的兩個 method
    public static int icpfunc(String check){
        // icp 是該 token 的順序
        // icp -> +- 1 -> */% 2 -> ( 4
        int icp;
        if(check.equals("(")){
            icp = 4;
            return icp;
        }
        else if(check.equals("*") ||
            check.equals("/") || check.equals("%") ){
            icp = 2;
            return icp;
        }
        else{
            icp = 1;
            return icp;
        }
    }
    public static int ispfunc(String check){
        // isp 是堆疊最上層 stack.peek 中的順序
        // isp -> ( 0 -> +- 1 -> */% 2
        int isp;
        if(check.equals("(")){
            isp = 0;
            return isp;
        }
        else if(check.equals("*") ||
            check.equals("/") || check.equals("%") ){
            isp = 2;
            return isp;
        }
        else{
            isp = 1;
            return isp;
        }
    }

    // 等同於老師要求的第一個 method AAAA
    // 把輸入的 s 做切割後放入陣列 arr 中
    public static String[] splitmethod (String s){
        // 用 split 切割字串
        String[] arr =
            s.split("((?=\\+|-|\\*|/|\\)|\\(|%)|(?<=\\+|-|\\*|/
        return arr;
    }

    // 等同於老師要求的 BBBB
    // 呼叫 splitmethod method
        去切割字串並按照後續排序
    public static String[] postfixe (String s){
        String[] postfixearr;
        String token = "+-*/(%";

        // 呼叫 splitmethod method 切割字串
        // 用 postfixearr 接回傳的陣列裡
        postfixearr = computing.splitmethod(s);

        String[] ans = new String[postfixearr.length];

        // 使用 stack 儲存運算子
```

```
        Stack<String> stack = new Stack<>();
        int icp = 0, isp = 0, i = 0;

        for (String str : postfixearr) {
            // 如果是 ) 則輸出所有 ( 以前的 stack 內的運算子
            // 但 () 都不輸出
            if (str.equals(")")) {
                while (!stack.peek().equals("(")) {
                    ans[i++] = stack.peek();
                    stack.pop();
                }
                // 把 ) 也 pop 掉
                stack.pop();
            }
            // 如果是 + - * / %
            else if (token.contains(str)) {
                // 運算此運算子的 icp 和 當前 stack 頂端的運算子的 isp
                icp = icpfunc(str);
                if (stack.empty()) {
                    stack.push(str);
                }
                else {
                    isp = ispfunc(stack.peek());
                    if (icp > isp) {
                        stack.push(str);
                    } else {
                        while (icp <=
                            ispfunc(stack.peek())) {
                            ans[i++] = stack.peek();
                            stack.pop();
                            if (stack.empty()) {
                                break;
                            }
                        }
                        stack.push(str);
                    }
                }
            }
            // 如果是運算元則直接輸出
            else {
                ans[i++] = str;
            }
        }
        // 最後因為有可能 array 在切割的時候最後一個值是運算元
        // 那樣就還會有運算子在 stack 還沒輸出 所以這裡在確認一次
        while(!stack.empty()){
            ans[i++] = stack.peek();
            stack.pop();
        }
        return ans;
    }

    // 相當於老師要求的 CCCC
    public static BigDecimal caculating (String s){
        String answer = "";
        String[] frompostfixe;
        Stack<String> stack = new Stack<>();
        String token = "+-*/%";

        // 呼叫 postfixe method 去排序後續
        // 而 postfixe method 會先去呼叫 splitmethod method 去切割字串
        // 所以 caculating 可以 call postfixe 和 splitmethod
        frompostfixe = postfixe(s);

        for (String str : frompostfixe) {

            // 如果跑道 frompostfixe 沒放後續字串的地方了 就 break 掉
            if( str == null ){
                break;
            }

            // 當碰到運算子時 先到 stack 去找最上面的兩個運算元
            // 分別讀取後 將兩者皆 pop 出來 ( 因為如此一來他們用過了 )
            // 對他們進行該運算子的運算
            // 最後將目前的運算結果 push 進 stack 裡
            //
            // 這樣就算之後有更要先乘除後加減的數也不會影響到之
            if (token.contains(str)) {

                BigDecimal top = BigDecimal.valueOf(
                    Double.parseDouble(stack.peek())
                    );
                stack.pop();
                BigDecimal sec = BigDecimal.valueOf(
                    Double.parseDouble(stack.peek())
                    );
                stack.pop();

                if (str.charAt(0) == '+'){
                    sec = sec.add( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '-'){
                    sec = sec.subtract( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '*'){
                    sec = sec.multiply( top );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '/'){
                    sec = sec.divide( top, 2,
                        RoundingMode.HALF_UP );
                    stack.push(sec.toString());
                }
                else if (str.charAt(0) == '%'){
                    sec = sec.remainder( top );
                    stack.push(sec.toString());
                }
                // 判斷目前運算的結果是多少 ( 無論是暫時的 還是最終結果
                answer = stack.peek();
            }
            // 如果是運算子就直接 push 進 stack
            else{
                stack.push(str);
            }
        }
        // 最後回傳 Bigdecimal 型態的 answer
        // 根據老師要求的 method 型別
        return BigDecimal.valueOf(
            Double.parseDouble(answer) );
    }
}
```

# 4 old exams

## 4.1 PA Palindromic Prime

```
/*
Palindromic Prime
    指數字為一質數並且迴文，請撰寫一個找出迴文質數的程式。
    Ex: 757 與 313 為迴文質數

Input Format
每組測資為一整數 N，請注意本題輸入數字的數目不定。

Output Format
```

```java
/*
請輸出前 N 個 Palindromic Prime(由 2 開始)。輸出的方式為每行顯示 10 個數字，每行數字間以空格分開，但最後一個數字不用空格。

Technical Specification
• 0 ≤ N ≤ 100
*/

import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String[] args) {

        // 直接在最開始就建好質數表
        boolean[] prime = new boolean[100000+5];
        Arrays.fill(prime,false);
        Vector<String> v = new Vector<String>();

        for( int i = 2; i <= 100000; i++ ){

            if(!prime[i]){
                // 因為 100000 * 100000 會超出 int 所以用 long
                // i * i 是 int -> 強制轉型 (long)
                for(long j = (long) i * i; j <= 100000; j += i ) {
                    // prime 的指標是放 int
                    prime[(int) j] = true;
                }

                // 先把 i 轉成 String ( 因為這樣比較方便 reverse )
                String str = String.valueOf(i);

                // 使用 StringBuilder 宣告 rev -> 因為 StringBuilder 才有 .reverse() 可以用
                StringBuilder rev = new StringBuilder();
                // 這裡 append (附加) str 的值給 rev
                rev.append(str);
                // reverse
                rev.reverse();

                // 兩字串判斷相等在 java 要用 .equals 不能直接 == ( 原因跟記憶體位址有關
                // 然後因為 rev 是 StringBuilder 但 .equals 是 String 的 所以要 .toString()
                if( str.equals(rev.toString()) ){
                    v.add(str);
                }

            }
        }

        Scanner input = new Scanner(System.in);

        // 重複輸入寫法
        while( input.hasNextInt() ){
            int n = input.nextInt();

            if( n == 0 ){
                System.out.println();
                continue;
            }

            for( int i = 0, k = 0; i < n; i++, k++ ){
                if( k == 9 || i == n - 1 ){
                    System.out.println(v.get(i));
                    k = -1;
                }
                else{
                    System.out.print(v.get(i) + " ");
                }
            }
        }
    }
}
```

## 4.2 You can say 11

```java
/*
給你一個正整數 N，判定它是否是 11 的倍數。提示:
在這裡，我們需要用到一個關於 11 倍數的小常識，那就是：11 倍數的「奇數位數字和」與「偶數位數字和」兩者的差必定為 11 的倍數。

Input Format
每列資料有一個正整數 N，N 最大可能到 1000 位數。若 N = 0 代表輸入結束

Output Format
先輸出奇數的和與偶數位的和，再對每一個輸入的數，輸出是否為 11 的倍數。輸出格式請

本題必須使用 char[] or String 儲存數入之正整數。
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        while( input.hasNext() ){
            String str = input.nextLine();
            if( str.equals("0") ){
                break;
            }

            int even = 0, odd = 0;
            for( int i = str.length()-1; i >= 0; i -= 2 ){
                // 要抓字串中的每個 char 要 str.charAt()
                // 要把 char 轉 int 蠻複雜的
                // 要先 String.valueOf() 然後 Integer.parseInt
                odd += Integer.parseInt( String.valueOf( str.charAt(i) ) );
            }
            for( int i = str.length()-2; i >= 0; i -= 2 ){
                even += Integer.parseInt( String.valueOf( str.charAt(i) ) );
            }

            int sub = Math.abs( even - odd );
            if( sub % 11 != 0){
                System.out.println( odd + " " + even + "\n" + str + " is not a multiple of 11." );
            }
            else{
                System.out.println( odd + " " + even + "\n" + str + " is a multiple of 11." );
            }
        }
    }
}
```

## 4.3 The BigInteger GCD LCM and Prime

```
/*
請利用BigInteger解題，不然不予給分

Problem
請輸入兩個數字M、N，找出M、N的最大公因數(GCD - Greatest Common Divisor) G及最小公倍數(LCM - Least Common
Multiple)L，並找出大於L的前5個可能質數(Prime)。

Input
輸入有多筆測資，每列代表一組測資，每組測資會有兩個數字M、N (數值可能會超過long可儲存的範圍)

Output
第一列先輸出兩數的最大公因數G及最小公倍數L，第二列再輸出大於L的前5個可能質數，每個
*/

import java.math.BigInteger;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        BigInteger M, N, G, L;

        while( input.hasNextBigInteger() ) {

            M = input.nextBigInteger();
            N = input.nextBigInteger();

            // BigInteger 內建函數 gcd
            G = M.gcd(N);
            // L = M * N / G
            L = M.multiply(N).divide(G);
            System.out.print(G + " " + L + "\n");

            for( int i = 0; i < 5; i++){

                // .nextProbablePrime ->
                    返回一個大於該 BigInteger 的質數
                System.out.print( i == 4 ?
                    L.nextProbablePrime() + "\n" :
                    L.nextProbablePrime() + " ");
                L = L.nextProbablePrime();
            }
        }
    }
}
```

## 4.4 Latin Square

```
/*
Description
A Latin square is an n-by-n array filled with n different Latin letters, each occurring exactly once in each row and once in each column.
Write a program that prompts the user to enter the number n and the array of characters, as shown in the sample output, and checks if the input array is a Latin square. The characters are the first n characters starting from A.

Input
Enter number n, and enter n rows of letters separated by spaces.
0 < n <= 26, all the letters in square are A ~ Z.
Input is terminated by a set where n = -1. This set should not be processed.

Output
Recognize if it is Latin Square or not. There are 2 different kinds of your answers:
The input array is a Latin square
The input array is not a Latin square
Overall, just check if it is Latin square or not.

Sample Input
4
A B C D
B A D C
C D B A
D C A B

3
B C A
B A C
C B A

3
B A C
A C B
C B A

4
D C B A
J A V A
U C C U
P P A P

-1

Sample Output
The input array is a Latin square

The input array is not a Latin square

The input array is a Latin square

The input array is not a Latin square
*/

import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        boolean newline = false;

        while( input.hasNextInt() ){

            int n = input.nextInt();
            if( n < 0 ){
                break;
            }
            // 控制最後不要換行
            if( newline ){
                System.out.println();
            }

            int[] count = new int[26+5];
            char[][] square = new char[26+5][26+5];
            boolean[][] row = new boolean[26+5][26+5];
            boolean[][] col = new boolean[26+5][26+5];

            // 陣列初始化
            for( int i = 0; i < 26; i++ ){
                Arrays.fill(count, 0);
                Arrays.fill(square[i], '0');
                Arrays.fill(row[i], false);
                Arrays.fill(col[i], false);
            }

            for( int i = 0; i < n; i++ ){
                for( int j = 0; j < n; j++ ){
```

```
                    // char 的讀取寫法
                        .next().charAt(0) -> 這個 0
                        是固定的
                        因為單字元的索引值就是 0
                    square[i][j] =
                        input.next().charAt(0);
                    // 計算每位是否接是出現 n 次 ->
                        每行都出現一次
                    count[ square[i][j] - 'A' ]++;
                }
            }
            // 吃最後的換行
            String space = input.nextLine();

            boolean flag = false;
            for( int i = 0; i < n; i++ ) {
                for (int j = i; j < n; j++) {

                    if (row[i][square[i][j] - 'A'] ||
                        row[j][square[j][i] - 'A'] ||
                        col[i][square[j][i] - 'A'] ||
                        col[j][square[i][j] - 'A']) {
                        flag = true;
                        break;
                    }
                    row[i][square[i][j] - 'A'] = true;
                    row[j][square[j][i] - 'A'] = true;
                    col[i][square[j][i] - 'A'] = true;
                    col[j][square[i][j] - 'A'] = true;

                }
                if (flag) {
                    break;
                }
            }
            for( int i = 0; i < 26; i++ ){
                if( count[i] != n && count[i] != 0 ){
                    flag = true;
                    break;
                }
            }

            if(flag){
                System.out.println("The input array
                    is not a Latin square");
            }
            else{
                System.out.println("The input array
                    is a Latin square");
            }
            newline = true;
        }
    }
}
```

# 5 practice

## 5.1 star pyramid

```
/*
3
  *
 ***
*****
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        for( int i = 1; i <= n; i++ ){

            for( int j = n-i; j > 0; j-- ){
                System.out.printf(" ");
            }

            for( int j = 0; j < i; j++ ){
                System.out.printf("*");
            }
            for( int j = 1; j < i; j++ ){
                System.out.printf("*");
            }

            System.out.println();

        }
    }
}
```

## 5.2 Nine Nine Multiplication Table

```
/*
請列出所有九九乘法表。
example:
1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 1 * 4 = 4 1 * 5 = 5 ...
    1 * 9 = 9\n
*每個公式中間都有一個空格
*每行最後都沒有空格，但有換行符號
*每個公式的答案都是 %2d <-重點
*可以利用System.out.printf(String , value ,......);
*/

public class Main {

    public static void main(String[] args) {

        for( int i = 1; i < 10; i++ ){
            for( int j = 1; j < 10; j++ ){

                if( j == 9 ){
                    System.out.printf( "%d * %d =
                        %2d\n", i, j, i * j );
                    continue;
                }

                System.out.printf( "%d * %d = %2d ",
                    i, j, i * j );

            }
        }

    }
}
```

## 5.3 prime number

```
/*
請輸入一個Integer並輸出小於其的所有質數
每行答案中間都有空格
每行最後都為換行符號(沒有空格)
*/

import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int n = input.nextInt();

        boolean prime[] = new boolean[1000];

```

```java
        // java 的 vector 寫法似乎是如此 :D
        // <> 內不能寫 int 要寫 Integer
        Vector<Integer> v = new Vector();

        // Arrays.fill 類似 C++ 的 memset( a, false, sizeof(a) );
        Arrays.fill( prime,false );

        for( int i = 2; i < n; i++ ){
            if( !prime[i] ){

                for( int j = i * i; j < n; j += i ){
                    prime[j] = true;
                }
                // 像 C++ 的 v.push_back(i)
                v.add(i);

            }
        }

        int i;
        for( i = 0; i < v.size()-1; i++ ){

            // 不能直接寫 v[i] 要寫 v.get(i)
            System.out.printf( "%d ", v.get(i) );

        }
        System.out.printf( "%d\n", v.get(i) );

    }
}
```

## 5.4 factor

```java
/*
你的任務是，給你一個正整數 N，判定它是否是 11 的倍數。

Input
每列資料有一個正整數N，N 最大可能到 1000 位數。
若 N = 0 代表輸入結束。

Output
對每個輸入的數，輸出是否為 11 的倍數。輸出格式請參考 Sample Output。
*/

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        // java 中要用 .hasNext() 來判斷是否輸入結束
        while( input.hasNext() ){

            // String 的 input 是 .nextLine()
            String str = input.nextLine();

            // java 用 == 進行比較的時候，比較的是他們在記憶體中的存放地址
            // 而 String,Integer,Date 這些類中 equals 可以用來做比較
            if( str.equals("0") ){
                break;
            }

            boolean flag = false;
            int first = 0, second = 0;
            for( int i = 0; i < str.length(); i++ ){

                if( flag ){
                    first += Integer.parseInt( String.valueOf( str.charAt(i) ) );
                    flag = false;
                }
                else{
                    second += Integer.parseInt( String.valueOf( str.charAt(i) ) );
                    flag = true;
                }
            }

            // java 的 abs 前面要加 Math.
            if( Math.abs( first - second ) % 11 == 0 ){
                System.out.printf("%s is a multiple of 11.\n", str );
            }
            else{
                System.out.printf("%s is not a multiple of 11.\n", str );
            }
        }

    }
}
```