Contents

1 Others

1.1 java

```java
//  Java

//  基礎運算判斷

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

// 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;
}
// ---

//  Scanner

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

// int 的 input 是 .nextInt()
int n = input.nextInt()

// boolean 的 input 是 .nextBoolean()
boolean flag = input.nextBoolean()
// ---

//  vector

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

    // 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) );
// ---

//  memset

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

//  Math

// 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 );
}
// ---

//  變數型態轉換


for( int i = 0; i < array.length; i++ ){
            // java 的 string to double 寫法
            tmp[i] = Double.parseDouble(array[i]);
        }
// ---

//  str.split

// 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");
// ---

// StringTokenizer

// 取運算子的部分用 StringTokenizer 更為方便
        StringTokenizer st = new
            StringTokenizer(str,"0123456789.");
        boolean flag = true;
        while(st.hasMoreTokens()){
            System.out.print( flag ? st.nextToken() :
                " " + st.nextToken());
            flag = false;
        }
// ---

//  string.contains()

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);
            }
        }
// ---

//  BigDecimal

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);
// -

// 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));
// ---


/*
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
*/

package com.company;

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);
    }
}
// ---

/*
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
輸入
為一介於0<= num <=1000的Integer
輸出
形態為Integer
*/

package com.company;

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);
    }
}

// ---

// 多筆測資輸入判斷
        double x1 = 0, y1 = 0, x2 = 0, y2 = 0;
        while(input.hasNext()){
            x1 = input.nextDouble();
            y1 = input.nextDouble();
            x2 = input.nextDouble();
            y2 = input.nextDouble();
        }
// ---

/*
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);

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

        // 讀取四個輸入
        x1 = input.nextDouble();
        y1 = input.nextDouble();
        x2 = input.nextDouble();
        y2 = input.nextDouble();

        // 利用 Math.pow 和 Math.sqrt 做運算
        System.out.println( Math.sqrt( Math.pow(x2 -
            x1, 2) + Math.pow(y2 - y1, 2) ) );
    }
}
// ---

/*
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
底 * 高 / 2

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;
```

```java
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));
        }


    }
}

// ---

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

package com.company;

import java.util.Arrays;
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();
    }
}
// ---

/*
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"。
*/

package com.company;

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

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");

    }
}
// ---

/*
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
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();
    }
}
// ---

//
    寫一個程式,使其依照輸入的列數來輸出一個金字塔,輸出結果
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();

        }
    }
}
// ---

/* 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 );

            }
        }

    }
}
// ---

/* 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 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) );

    }
}
// ---

/*
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
            // 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 );
            }
        }

    }
}
// ---

/*
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
程式要求 (未依規定，程式以0分計並扣學期總分2分)
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+,-,*,/,(,),%，

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();
```

```
667
668         flag = true;
669         for( String again : array){
670             if( token.contains(again) ){
671                 System.out.print( flag ? again : " "
                        + again);
672                 flag = false;
673             }
674         }
675         System.out.println();
676
677         // BigDecimal 四捨五入的寫法 .setScale(
                想要留取的位數, RoundingMode.HALF_UP)
678         System.out.println(ans.setScale(3,
                RoundingMode.HALF_UP));
679     }
680 }
681 // ---
682
683 /*
684 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
685 Input Format
686 每組測資為一整數 N，請注意本題輸入數字的數目不定。
687 Output Format
688 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
689 間以空格分開，但最後一個數字不用空格直接換行。
690 Technical Specification
691 • 0 ≤ N ≤ 100
692  */
693 import java.util.Arrays;
694 import java.util.Scanner;
695 import java.util.Vector;
696
697 public class Main {
698
699     public static void main(String[] args) {
700
701         // 直接在最開始就建好質數表
702         boolean[] prime = new boolean[100000+5];
703         Arrays.fill(prime,false);
704         Vector<String> v = new Vector<String>();
705
706         for( int i = 2; i <= 100000; i++ ){
707
708             if(!prime[i]){
709                 // 因為 100000 * 100000 會超出 int
                        所以用 long
710                 // i * i 是 int -> 強制轉型 (long)
711                 for(long j = (long) i * i; j <=
                        100000; j += i ) {
712                     // prime 的指標是放 int
713                     prime[(int) j] = true;
714                 }
715
716                 // 先把 i 轉成 String (
                        因為這樣比較方便 reverse )
717                 String str = String.valueOf(i);
718
719                 // 使用 StringBuilder 宣告 rev ->
                        因為 StringBuilder 才有
                        .reverse() 可以用
720                 StringBuilder rev = new
                        StringBuilder();
721                 // 這裡 append (附加) str 的值給 rev
722                 rev.append(str);
723                 // reverse
724                 rev.reverse();
725
726                 // 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
727                 // 然後因為 rev 是 StringBuilder 但
                        .equals 是 String 的 所以要
                        .toString()
728                 if( str.equals(rev.toString()) ){
729                     v.add(str);
730                 }
731
732             }
733         }
734
735         Scanner input = new Scanner(System.in);
736
737         // 重複輸入寫法
738         while( input.hasNextInt() ){
739             int n = input.nextInt();
740
741             if( n == 0 ){
742                 System.out.println();
743                 continue;
744             }
745
746             for( int i = 0, k = 0; i < n; i++, k++ ){
747                 if( k == 9 || i == n - 1 ){
748                     System.out.println(v.get(i));
749                     k = -1;
750                 }
751                 else{
752                     System.out.print(v.get(i) + " ");
753                 }
754             }
755         }
756     }
757 }
758 // ---
759
760
761 /*
762 Problem
763
764 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)。
765
766 Input
767
768 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)
769
770 Output
771
772 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，每個質數間
773  */
774
775  import java.math.BigInteger;
776 import java.util.Scanner;
777
778 public class Main {
779
780     public static void main(String[] args) {
781         Scanner input = new Scanner(System.in);
782         BigInteger M, N, G, L;
783
784         while( input.hasNextBigInteger() ) {
785
786             M = input.nextBigInteger();
787             N = input.nextBigInteger();
788
789             // BigInteger 內建函數 gcd
790             G = M.gcd(N);
791             // L = M * N / G
792             L = M.multiply(N).divide(G);
793             System.out.print(G + " " + L + "\n");
794
795             for( int i = 0; i < 5; i++){
796
```

```
797                 // .nextProbablePrime ->
                        返回一個大於該 BigInteger 的質數
798                 System.out.print( i == 4 ?
                        L.nextProbablePrime() + "\n" :
                        L.nextProbablePrime() + " ");
799                 L = L.nextProbablePrime();
800             }
801         }
802     }
803 }
```