


```

121         System.out.println(ans.setScale(3,
122             // ---
123             RoundingMode.HALF_UP));
124
125     /*
126     請利用克萊姆(Cramer)公式，嘗試解出2*2的線性方程式
127     ax + by = e
128     cx + dy = f
129     輸入：
130     輸入 a b e
131     與 c d f 分別代表兩個線性方程式的常數與答案
132     其形態皆為 double
133     輸出：
134     請利用 Cramer 解出 x 與 y 二值
135     其形態皆為 double
136     */
137
138     package com.company;
139
140     import java.util.Scanner;
141
142     public class Main {
143
144         public static void main(String[] args) {
145             Scanner input = new Scanner(System.in);
146
147             double x1 = 0, y1 = 0, a1 = 0, x2 = 0, y2 =
148                 0, a2 = 0;
149
150             x1 = input.nextDouble();
151             y1 = input.nextDouble();
152             a1 = input.nextDouble();
153             x2 = input.nextDouble();
154             y2 = input.nextDouble();
155             a2 = input.nextDouble();
156
157             // 克萊姆公式
158             // [a b][x] = [e]
159             // [c d][y] = [f]
160             // x = ed - bf / ad - bc
161             // y = af - ec / ad - bc
162
163             double x = 0, y = 0;
164             x = ( ( a1 * y2 ) - ( y1 * a2 ) ) / ( ( x1 *
165                 y2 ) - ( y1 * x2 ) );
166             y = ( ( x1 * a2 ) - ( a1 * x2 ) ) / ( ( x1 *
167                 y2 ) - ( y1 * x2 ) );
168
169             System.out.println(x + " " + y);
170         }
171     }
172     // ---
173
174     /*
175     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
176     為一介於 0 <= num <= 1000 的 Integer
177     輸出
178     形態為 Integer
179     */
180
181     package com.company;
182
183     import java.util.Scanner;
184
185     public class Main {
186
187         public static void main(String[] args) {
188             Scanner input = new Scanner(System.in);
189             String str = input.nextLine();
190
191             // 先把字串內的每位數單獨拆分進 array 內
192             String[] array = str.split("");
193
194             Integer ans = new Integer(1);

```

```

193         for( int i = 0; i < str.length(); i++ ){
194
195             // 用 Integer.parseInt() 把 array[i]
196             內的字符轉數字
197             ans = Integer.parseInt(array[i]) * ans;
198         }
199         System.out.println(ans);
200     }
201 }
202
203 // ---
204
205 // 多筆測資輸入判斷
206 double x1 = 0, y1 = 0, x2 = 0, y2 = 0;
207 while(input.hasNext()){
208     x1 = input.nextDouble();
209     y1 = input.nextDouble();
210     x2 = input.nextDouble();
211     y2 = input.nextDouble();
212 }
213 // ---
214
215 /*
216 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
217 點座標，您可以利用 Math.pow(a,0.5) 來計算√a。
218 輸入
219 x1,y1: 1.5 -3.4
220 x2,y2: 4 5
221 輸出
222 8.764131445842194
223 計算兩點間距離的公式為
224 (x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) 開根號
225 */
226
227 import java.util.Scanner;
228
229 public class Main {
230
231     public static void main(String[] args) {
232         Scanner input = new Scanner(System.in);
233
234         double x1 = 0, y1 = 0, x2 = 0, y2 = 0;
235
236         // 讀取四個輸入
237         x1 = input.nextDouble();
238         y1 = input.nextDouble();
239         x2 = input.nextDouble();
240         y2 = input.nextDouble();
241
242         // 利用 Math.pow 和 Math.sqrt 做運算
243         System.out.println( Math.sqrt( Math.pow(x2 -
244             x1, 2) + Math.pow(y2 - y1, 2) ) );
245     }
246 }
247 // ---
248
249 /*
250 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)
251 及(x3,y3)，計算其面積。
252 底 * 高 / 2
253 輸入
254 三角形三個點: 1.5 -3.4 4.6 5 9.5 -3.4
255 輸出
256 面積為 33.600
257
258 -----*輸出格式為小數點後三位*-----
259 -----*若三點不為三角形請輸出 "Not Triangle"*-----
260 */
261
262 import java.math.BigDecimal;
263 import java.math.RoundingMode;
264 import java.util.Arrays;

```

```

265 import java.util.Scanner;
266
267 public class Main {
268
269     private static Math Precision;
270
271     public static void main(String[] args) {
272         Scanner input = new Scanner(System.in);
273
274         double x1 = 0, y1 = 0, x2 = 0, y2 = 0, x3 =
            0, y3 = 0;
275
276         x1 = input.nextDouble();
277         y1 = input.nextDouble();
278         x2 = input.nextDouble();
279         y2 = input.nextDouble();
280         x3 = input.nextDouble();
281         y3 = input.nextDouble();
282
283         double[] array = new double[3];
284         array[0] = Math.sqrt( Math.pow(x2 - x1, 2) +
            Math.pow(y2 - y1, 2) );
285         array[1] = Math.sqrt( Math.pow(x3 - x2, 2) +
            Math.pow(y3 - y2, 2) );
286         array[2] = Math.sqrt( Math.pow(x3 - x1, 2) +
            Math.pow(y3 - y1, 2) );
287
288         // 判斷是否為三角形
289         // Arrays.sort(陣列名稱) -> 自動排序
290         Arrays.sort(array);
291
292         if ( array[0] + array[1] <= array[2] ){
293             System.out.println("Not Triangle");
294         }
295         else{
296             // 海龍公式
297             double s = 0;
298             for( int i = 0; i < 3; i++ ){
299                 s = s + array[i];
300             }
301             s = s / 2;
302
303             double a = 0;
304             a = Math.sqrt( s * (s-array[0]) *
                (s-array[1]) * (s-array[2]) );
305
306             // 利用 BigDecimal 去取小數點並四捨五入
307             BigDecimal ans = new BigDecimal(a);
308             System.out.println( ans.setScale(3,
                RoundingMode.HALF_UP));
309         }
310     }
311 }
312
313 // ---
314
315 /*
316 Write a program to input an integer N and print all
317 prime numbers less than N.
318 Please print 10 prime numbers in each line.
319 */
320
321 package com.company;
322
323 import java.util.Arrays;
324 import java.util.Scanner;
325 import java.util.Vector;
326
327 public class Main {
328
329     public static void main(String[] args) {
330         Scanner input = new Scanner(System.in);
331         int n = input.nextInt();
332
333         // 建質數表

```

```

335         // java 中沒有全域變數
336         // 所以這邊建表目前是每次輸入都重建一次
337         // 但不夠漂亮
338         boolean[] prime = new boolean[1000000+5];
339
340         // 用 vector 存質數們
341         Vector<Integer> v = new Vector<Integer>();
342         for( int i = 2; i <= n; i++ ){
343             if(!prime[i]){
344                 v.add(i);
345                 for( int j = i * i; j <= n; j += i ){
346                     prime[j] = true;
347                 }
348             }
349         }
350
351         for( int i = 0, k = 0; i < v.size(); i++, k++ ){
352             if( k == 9 || i == v.size() - 1 ){
353                 System.out.println(v.get(i));
354                 // 把 k 歸回去 -1 是因為下次輪迴會先
355                 k++;
356                 k = -1;
357             }
358             else{
359                 System.out.print(v.get(i) + " ");
360             }
361         }
362         v.clear();
363     }
364     // ---
365     /*
366     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。
367     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 "2012 had 29
368     days"。如果使用者月份輸入3，年份輸入2015，程式顯示 "March
369     2015 had 31 days"。
370     */
371     package com.company;
372
373     import java.util.Arrays;
374     import java.util.Scanner;
375     import java.util.Vector;
376
377     public class Main {
378
379         public static void main(String[] args) {
380             Scanner input = new Scanner(System.in);
381             int month = input.nextInt();
382             int year = input.nextInt();
383
384             // 分別存 30 天的月份 和 31 天的月份
385             // 以及他們的英文月份名
386             int[] thirty = {1,3,5,7,8,10,12};
387             String[] thirty_eng = {"January", "March",
388                 "May", "July", "August", "October",
389                 "December"};
390
391             int[] thirty_one = {4,6,9,11};
392             String[] thirty_one_eng = {"April", "June",
393                 "September", "November"};
394
395             // 查詢他是否是 30 天的月份
396             // 最後放 return 後面的其他程式碼就不會執行了
397             for( int i = 0; i < thirty.length; i++){
398                 if(month == thirty[i]){
399                     System.out.println(thirty_eng[i] + "
400                         " + year + " had 31 Days" );
401                     return;
402                 }
403             }
404             for( int i = 0; i < thirty_one.length; i++){

```

```

398         if(month == thirty_one[i]){
399             System.out.println(thirty_one_eng[i]
400                 + " " + year + " had 31 Days" );
401             return;
402         }
403         if( year % 4 == 0){
404             System.out.println("February " + year + "
405                 had 29 Days");
406             return;
407         }
408         System.out.println("February " + year + " had
409             28 Days");
410     }
411     // ---
412     /*
413     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
414     計算距離的公式
415     (x*x + y*y) 開根號
416     輸入
417     4 5
418     9 9
419     輸出
420     (4.0,5.0) is in the circle
421     (9.0,9.0) is not in the circle
422     */
423     import java.util.Scanner;
424     public class Main {
425         public static void main(String[] args) {
426             Scanner input = new Scanner(System.in);
427             double x = input.nextDouble();
428             double y = input.nextDouble();
429
430             //是否在半徑 10 以內
431             if( Math.sqrt( Math.pow(x, 2) + Math.pow(y,
432                 2) ) <= 10 ){
433                 // 如果要輸出 ("%f %d", a, b) 要用 ->
434                 System.out.printf
435                     ("%f %d", x, y);
436                 // %.1f 控制小數點後 1 位
437                 System.out.printf("(%.1f,%.1f) is in the
438                     circle" ,x ,y);
439             }
440             else{
441                 System.out.printf("(%.1f,%.1f) is not in
442                     the circle" ,x ,y);
443             }
444             System.out.println();
445         }
446     }
447     // ---
448     //
449     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
450     import java.util.Scanner;
451     public class Main {
452         public static void main(String[] args) {
453             Scanner input = new Scanner(System.in);
454             int n = input.nextInt();
455
456             for( int i = 1; i <= n; i++ ){
457                 for( int j = n-i; j > 0; j-- ){
458                     System.out.printf(" ");
459                 }
460                 for( int j = 0; j < i; j++ ){
461                     System.out.printf("*");
462                 }
463             }
464         }
465     }
466     }
467     for( int j = 1; j < i; j++ ){
468         System.out.printf("*");
469     }
470     System.out.println();
471 }
472 }
473 }
474 }
475 }
476 // ---
477 /* 請列出所有九九乘法表。
478 example:
479 1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 1 * 4 = 4 1 * 5 = 5 ...
480 1 * 9 = 9\n
481 *每個公式中間都有一個空格
482 *每行最後都沒有空格，但有換行符號
483 *每個公式的答案都是 %2d <-重點
484 *可以利用System.out.printf(String , value ,.....);
485 */
486 public class Main {
487     public static void main(String[] args) {
488         for( int i = 1; i < 10; i++ ){
489             for( int j = 1; j < 10; j++ ){
490                 if( j == 9 ){
491                     System.out.printf( "%d * %d =
492                         %2d\n", i, j, i * j );
493                     continue;
494                 }
495                 System.out.printf( "%d * %d = %2d ",
496                     i, j, i * j );
497             }
498         }
499     }
500 }
501 }
502 }
503 }
504 }
505 // ---
506 /* 請輸入一個Integer並輸出小於其的所有質數
507 每行答案中間都有空格
508 每行最後都為換行符號(沒有空格)
509 */
510 import java.util.*;
511 public class Main {
512     public static void main(String[] args) {
513         Scanner input = new Scanner(System.in);
514         int n = input.nextInt();
515         boolean prime[] = new boolean[1000];
516
517         // java 的 vector 寫法似乎是如此 :D
518         // <> 內不能寫 int 要寫 Integer
519         Vector<Integer> v = new Vector();
520
521         // Arrays.fill 類似 C++ 的 memset( a, false,
522             sizeof(a) );
523         Arrays.fill( prime, false );
524
525         for( int i = 2; i < n; i++ ){
526             if( !prime[i] ){
527                 for( int j = i * i; j < n; j += i ){
528                     prime[j] = true;
529                 }
530                 // 像 C++ 的 v.push_back(i)
531                 v.add(i);
532             }
533         }
534     }
535 }
536 }
537 }

```

```

538     }
539 }
540
541 int i;
542 for( i = 0; i < v.size()-1; i++ ){
543
544     // 不能直接寫 v[i] 要寫 v.get(i)
545     System.out.printf( "%d ", v.get(i) );
546
547 }
548 System.out.printf( "%d\n", v.get(i) );
549
550 }
551 }
552 // ---
553
554 /*
555 你的任務是，給你一個正整數 N，判定它是否是 11 的倍數。
556
557 Input
558 每列資料有一個正整數N，N 最大可能到 1000 位數。
559 若 N = 0 代表輸入結束。
560
561 Output
562 對每個輸入的數，輸出是否為 11 的倍數。輸出格式請參考
563     Sample Output。
564 */
565 import java.util.Scanner;
566
567 public class Main {
568
569     public static void main(String[] args) {
570
571         Scanner input = new Scanner(System.in);
572
573         // java 中要用 .hasNext() 來判斷是否輸入結束
574         while( input.hasNext() ){
575
576             // String 的 input 是 .nextLine()
577             String str = input.nextLine();
578
579             // java 用 ==
580             // 進行比較的時候，比較的是他們在記憶體中
581             // 而 String,Integer,Date 這些類中 equals
582             // 可以用來做比較
583             if( str.equals("0") ){
584                 break;
585             }
586
587             boolean flag = false;
588             int first = 0, second = 0;
589             for( int i = 0; i < str.length(); i++ ){
590
591                 if( flag ){
592                     first += Integer.parseInt(
593                         String.valueOf( str.charAt(i)
594                     ) );
595                     flag = false;
596                 }
597                 else{
598                     second += Integer.parseInt(
599                         String.valueOf( str.charAt(i)
600                     ) );
601                     flag = true;
602                 }
603             }
604
605             // java 的 abs 前面要加 Math.
606             if( Math.abs( first - second ) % 11 == 0
607             ){
608                 System.out.printf("%s is a multiple
609                     of 11.\n", str );
610             }
611             else{
612                 System.out.printf("%s is not a
613                     multiple of 11.\n", str );
614             }
615         }
616     }
617 }

```

```

604     }
605 }
606
607 }
608 }
609 // ---
610
611 /*
612 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組
613 程式要求（未依規定，程式以0分計並扣學期總分2分）
614 1. 使用String類別 & 字串輸入
615 2. 使用StringTokenizer類別 或
        String類別之split()method來切割中序運算式。
616 3. 使用String [] array 儲存所有tokens。
617 4. 將數字token轉換成實數，加總後輸出總和。Hint: Use
        BigDecimal in java.math package
618
619
620 輸入
621
622     每組測資會給予不一定長度的算式，符號包含+,-,*,/,(),%,^
623
624 輸出
625     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。
626     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。
627     第三行印出運算元的總和，小事點以下印三位。
628
629 */
630
631 import java.math.BigDecimal;
632 import java.math.RoundingMode;
633 import java.util.Scanner;
634
635 public class Main {
636
637     public static void main(String[] args) {
638         Scanner input = new Scanner(System.in);
639         String str = input.nextLine();
640         String token = "+-*/%/()%^";
641
642         // 當字串中帶有 + * / \ 等符號時要寫 \
        // 轉義，因為他們在正則表達式中有相應的不同意義
643         // limit is 0; array contains all substrings
644         // Positive Lookahead or Lookbehind 觀念
645         // “((?=)|(?<=))” -> 才會把 運算子 還有
        // 運算元 都單獨切開
646         String[] array =
            str.split("((?=\+|-|\\*|/|\\\\)\\\\(|%)|(?<=\+|-|/|\\\\)\\\\)");
647
648         // 預設 ans 賦值 可控制小數點位數 ex 0.000 ->
        // 小數點後三位
649         BigDecimal ans = new BigDecimal("0.000");
650
651         boolean flag = true;
652         for( String check : array){
653             // string.contains
654             // 一次查詢多個關鍵字 的寫法
655             // 要另外存一個數組 token
656             // 且在查詢時要以此 for ( String check :
657             // array ) 去做掃描
658             if( !token.contains(check) ){
659                 // 將字串型態的數字轉成實數並相加
660                 BigDecimal b = new BigDecimal(check);
661                 // (boolean 判斷式) ? (true的輸出) :
662                 // (false 的輸出)
663                 System.out.print( flag ? check : " "
664                     + check);
665                 flag = false;
666
667                 // BigDecimal 內建 .add 可以直接相加
668                 ans = ans.add(b);
669             }
670         }
671         System.out.println();
672     }
673 }

```

```

667     flag = true;
668     for( String again : array){
669         if( token.contains(again) ){
670             System.out.print( flag ? again : " "
671                 + again);
672             flag = false;
673         }
674     }
675     System.out.println();
676
677     // BigDecimal 四捨五入的寫法 .setScale(
678     // 想要留取的位數, RoundingMode.HALF_UP)
679     System.out.println(ans.setScale(3,
680         RoundingMode.HALF_UP));
681 }
682 // ---
683 /*
684 Palindromic Prime
685 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
686 Ex: 757 與 313 為迴文質數
687 Input Format
688 每組測資為一整數 N，請注意本題輸入數字的數目不定。
689 Output Format
690 請輸出前 N 個 Palindromic Prime(由 2
691 開始)。輸出的方式為每行顯示 10 個數字，每行數字
692 間以空格分開，但最後一個數字不用空格直接換行。
693 Technical Specification
694 • 0 ≤ N ≤ 100
695 */
696 import java.util.Arrays;
697 import java.util.Scanner;
698 import java.util.Vector;
699
700 public class Main {
701
702     public static void main(String[] args) {
703
704         // 直接在最開始就建好質數表
705         boolean[] prime = new boolean[100000+5];
706         Arrays.fill(prime, false);
707         Vector<String> v = new Vector<String>();
708
709         for( int i = 2; i <= 100000; i++ ){
710             if(!prime[i]){
711                 // 因為 100000 * 100000 會超出 int
712                 // 所以用 long
713                 // i * i 是 int -> 強制轉型 (long)
714                 for(long j = (long) i * i; j <=
715                     100000; j += i ) {
716                     // prime 的指標是放 int
717                     prime[(int) j] = true;
718                 }
719
720                 // 先把 i 轉成 String (
721                 // 因為這樣比較方便 reverse )
722                 String str = String.valueOf(i);
723
724                 // 使用 StringBuilder 宣告 rev ->
725                 // 因為 StringBuilder 才有
726                 // .reverse() 可以用
727                 StringBuilder rev = new
728                     StringBuilder();
729                 // 這裡 append (附加) str 的值給 rev
730                 rev.append(str);
731                 // reverse
732                 rev.reverse();
733
734                 // 兩字串判斷相等在 java 要用 .equals
735                 // 不能直接 == (原因跟記憶體位址有關
736                 // 然後因為 rev 是 StringBuilder 但
737                 // .equals 是 String 的 所以要

```

```

728         .toString()
729         if( str.equals(rev.toString()) ){
730             v.add(str);
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 Problem
762 請輸入兩個數字 M、N，找出 M、N 的最大公因數 (GCD -
763 Greatest Common Divisor) G 及最小公倍數 (LCM -
764 Least Common
765 Multiple)L，並找出大於 L 的前 5 個可能質數 (Prime)。
766 Input
767 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
768 (數值可能會超過 long 可儲存的範圍)
769
770 Output
771 第一列先輸出兩數的最大公因數
772 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++){

```

```
797 |         // .nextProbablePrime ->
798 |         返回一個大於該 BigInteger 的質數
799 |         System.out.print( i == 4 ?
800 |             L.nextProbablePrime() + "\n" :
801 |             L.nextProbablePrime() + " ");
802 |         L = L.nextProbablePrime();
803 |     }
804 | }
```