

Contents

1	chapter practice	1
1.1	Cramer	1
1.2	calculation	1
1.3	distance	1
1.4	Heron's formula	2
1.5	year month day	2
1.6	circle distance	3
1.7	Regular polygon area	3
1.8	binary to decimal	3
1.9	substring	3
1.10	prime or gcd	3
1.11	pi calculation	4
1.12	factorial	4
1.13	Almanac	4
1.14	perfect number	5
1.15	maximum appear times	5
1.16	leap year	5
2	exam	5
2.1	BigDecimal Calculator	5
2.2	Magic Square	6
2.3	Prime Factorization	7
3	homeworks	8
3.1	Print Prime Numbers	8
3.2	P1 Infix Notation to Postfix Notation	8
3.3	P2 Infix Notation to Postfix Notation	9
3.4	P3 Infix Notation to Postfix Notation	10
4	old exams	11
4.1	PA Palindromic Prime	11
4.2	You can say 11	12
4.3	The BigInteger GCD LCM and Prime	13
4.4	Latin Square	13
5	practice	14
5.1	star pyramid	14
5.2	Nine Nine Multiplication Table	14
5.3	prime number	14
5.4	factor	15

1 chapter practice

1.1 Cramer

```

1  /*
2  請利用克萊姆(Cramer)公式，嘗試解出2*2的線性方程式
3  ax + by = e
4  cx + dy = f
5
6  輸入：
7  輸入a b e
8  與 c d f 分別代表兩個線性方程式的常數與答案
9  其形態皆為double
10
11 輸出：
12 請利用Cramer解出x與y二值
13 其形態皆為double
14 */
15
16 import java.util.Scanner;
17
18 public class Main {
19
20     public static void main(String[] args) {
21         Scanner input = new Scanner(System.in);
22
23         double x1 = 0, y1 = 0, a1 = 0, x2 = 0, y2 =
24             0, a2 = 0;
25
26         x1 = input.nextDouble();
27         y1 = input.nextDouble();
28         a1 = input.nextDouble();
29         x2 = input.nextDouble();
30         y2 = input.nextDouble();
31         a2 = input.nextDouble();

```

```

32     // 克萊姆公式
33     // [a b][x] = [e]
34     // [c d][y] = [f]
35     // x = ed - bf / ad - bc
36     // y = af - ec / ad - bc
37     double x = 0, y = 0;
38     x = ( ( a1 * y2 ) - ( y1 * a2 ) ) / ( ( x1 *
39         y2 ) - ( y1 * x2 ) );
40     y = ( ( x1 * a2 ) - ( a1 * x2 ) ) / ( ( x1 *
41         y2 ) - ( y1 * x2 ) );
42     System.out.println(x + " " + y);
43 }

```

1.2 calculation

```

1  /*
2  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。
3
4  輸入
5  為一介於0<= num <=1000的Integer
6
7  輸出
8  形態為Integer
9  */
10
11 import java.util.Scanner;
12
13 public class Main {
14
15     public static void main(String[] args) {
16         Scanner input = new Scanner(System.in);
17         String str = input.nextLine();
18
19         // 先把字串內的每位數單獨拆分進 array 內
20         String[] array = str.split("");
21
22         Integer ans = new Integer(1);
23         for( int i = 0; i < str.length(); i++){
24
25             // 用 Integer.parseInt() 把 array[i]
26             // 內的字符轉數字
27             ans = Integer.parseInt(array[i]) * ans;
28         }
29
30         System.out.println(ans);
31     }

```

1.3 distance

```

1  /*
2  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
3  點間的距離。
4  請注意，您可以利用Math.pow(a,0.5)來計算√a。
5
6  輸入
7  x1,y1: 1.5 -3.4
8  x2,y2: 4 5
9
10 輸出
11 8.764131445842194
12
13 計算兩點間距離的公式為
14 (x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) 開根號
15 */
16
17 import java.util.Scanner;
18
19 public class Main {
20
21     public static void main(String[] args) {
22         Scanner input = new Scanner(System.in);

```

```

21 String str = input.nextLine();
22
23 // 先把字串內的每位數單獨拆分進 array 內
24 String[] array = str.split("");
25
26 Integer ans = new Integer(1);
27 for( int i = 0; i < str.length(); i++ ){
28
29     // 用 Integer.parseInt() 把 array[i]
        內的字符轉數字
30     ans = Integer.parseInt(array[i]) * ans;
31 }
32
33 System.out.println(ans);
34 }
35 }

```

```

52 s = s + array[i];
53 }
54 s = s / 2;
55
56 double a = 0;
57 a = Math.sqrt( s * (s-array[0]) *
58     (s-array[1]) * (s-array[2]) );
59
60 // 利用 BigDecimal 去取小數點並四捨五入
61 BigDecimal ans = new BigDecimal(a);
62 System.out.println( ans.setScale(3,
63     RoundingMode.HALF_UP));
64 }
65 }
66 }

```

1.4 Heron's formula

```

1 /*
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)、(x3,y3)，
3 計算並輸出該三角形的面積。
4 公式：A = √ s(s-a)(s-b)(s-c)
5
6 輸入
7 三角形三個點： 1.5 -3.4 4.6 5 9.5 -3.4
8 輸出
9 面積為33.600
10
11 輸出格式為小數點後三位
12 若三點不為三角形請輸出 "Not Triangle"
13 */
14
15 import java.math.BigDecimal;
16 import java.math.RoundingMode;
17 import java.util.Arrays;
18 import java.util.Scanner;
19
20 public class Main {
21
22     private static Math Precision;
23
24     public static void main(String[] args) {
25         Scanner input = new Scanner(System.in);
26
27         double x1 = 0, y1 = 0, x2 = 0, y2 = 0, x3 = 0, y3 = 0;
28
29         x1 = input.nextDouble();
30         y1 = input.nextDouble();
31         x2 = input.nextDouble();
32         y2 = input.nextDouble();
33         x3 = input.nextDouble();
34         y3 = input.nextDouble();
35
36         double[] array = new double[3];
37         array[0] = Math.sqrt( Math.pow(x2 - x1, 2) +
38             Math.pow(y2 - y1, 2) );
39         array[1] = Math.sqrt( Math.pow(x3 - x2, 2) +
40             Math.pow(y3 - y2, 2) );
41         array[2] = Math.sqrt( Math.pow(x3 - x1, 2) +
42             Math.pow(y3 - y1, 2) );
43
44         // 判斷是否為三角形
45         // Arrays.sort(陣列名稱) -> 自動排序
46         Arrays.sort(array);
47
48         if ( array[0] + array[1] <= array[2] ){
49             System.out.println("Not Triangle");
50         }
51         else{
52             // 海龍公式
53             double s = 0;
54             for( int i = 0; i < 3; i++ ){

```

1.5 year month day

```

1 /*
2 請撰寫一程式，提示使用者輸入月份與年份，接著顯示該月份的夫數。
3 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
4 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"。
5
6 輸入
7 月份與年份： 2 2012
8 輸出
9 February 2012 had 29 days
10
11 輸出格式為：月份 年份 had 夫數
12 若月份不為1~12請輸出 "Invalid month"
13 */
14
15 import java.util.Scanner;
16
17 public class Main {
18
19     public static void main(String[] args) {
20         Scanner input = new Scanner(System.in);
21         int month = input.nextInt();
22         int year = input.nextInt();
23
24         // 分別存 30 天的月份 和 31 天的月份
25         // 以及他們的英文月份名
26         int[] thirty = {1,3,5,7,8,10,12};
27         String[] thirty_eng = {"January", "March",
28             "May", "July", "August", "October",
29             "December"};
30
31         int[] thirty_one = {4,6,9,11};
32         String[] thirty_one_eng = {"April", "June",
33             "September", "November"};
34
35         // 查詢他是否是 30 天的月份
36         // 最後放 return 後面的其他程式碼就不會執行了
37         for( int i = 0; i < thirty.length; i++){
38             if(month == thirty[i]){
39                 System.out.println(thirty_eng[i] + "
40                     " + year + " had 31 Days" );
41                 return;
42             }
43         }
44         for( int i = 0; i < thirty_one.length; i++){
45             if(month == thirty_one[i]){
46                 System.out.println(thirty_one_eng[i]
47                     + " " + year + " had 31 Days" );
48                 return;
49             }
50         }
51         if( year % 4 == 0){
52             System.out.println("February " + year + "
53                 had 29 Days");
54             return;
55         }
56         System.out.println("February " + year + " had
57             28 Days");
58     }
59 }

```

1.6 circle distance

```

1  /*
2  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
3  計算距離的公式
4
5  (x*x + y*y) 開根號
6
7  輸入
8  4 5
9  9 9
10
11 輸出
12 (4.0,5.0) is in the circle
13 (9.0,9.0) is not in the circle
14 */
15
16 import java.util.Scanner;
17
18 public class Main {
19
20     public static void main(String[] args) {
21         Scanner input = new Scanner(System.in);
22         double x = input.nextDouble();
23         double y = input.nextDouble();
24
25         //是否在半徑 10 以內
26         if( Math.sqrt( Math.pow(x, 2) + Math.pow(y,
27             2) ) <= 10 ){
28             // 如果要輸出 ("%f %d", a, b) 要用 ->
29             System.out.printf
30             // %.1f 控制小數點後 1 位
31             System.out.printf("(%.1f,%.1f) is in the
32             circle" ,x ,y);
33         }
34         else{
35             System.out.printf("(%.1f,%.1f) is not in
36             the circle" ,x ,y);
37         }
38         System.out.println();
39     }
40 }

```

1.7 Regular polygon area

```

1  /*
2  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
3   $A = (n*s^2) / 4 * \tan(\pi/n)$ 
4
5  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
6
7  輸入
8  n=邊數: 5
9  s=邊長: 6.5
10
11 輸出
12 74.69017017488385
13 */
14
15 import java.util.Scanner;
16
17 public class Main {
18
19     public static void main(String[] args) {
20         Scanner input = new Scanner(System.in);
21
22         double n = input.nextDouble();
23         double s = input.nextDouble();
24
25         // java 有內建 tan 和 PI 在 Math 裡 太酷了ㄟ
26         System.out.println( (n*s*s) /
27             (4*Math.tan(Math.PI/n)) );
28     }
29 }

```

1.8 binary to decimal

```

1  /*
2  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。
3  輸入 0111 1001 1100 輸出 7 9 12
4
5  */
6
7 import java.util.Scanner;
8 import java.util.StringTokenizer;
9
10 public class Main {
11
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14
15         String bin = input.nextLine();
16         StringTokenizer st = new StringTokenizer(
17             bin, " ");
18
19         while(st.hasMoreTokens()){
20             int two = 1, deci = 0;
21             String str = st.nextToken();
22             for( int i = str.length()-1; i >= 0; i-- )
23             {
24                 deci += ( str.charAt(i) - '0' ) * two;
25                 two *= 2;
26             }
27             System.out.println(deci);
28         }
29     }
30 }

```

1.9 substring

```

1  /*
2  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
3  串的字串
4  輸入 string s1:ABCD string s2:BC
5  輸出 BC is a substring of ABCD
6
7  輸入 string s1:ABCD string s2:CA
8  輸出 CA is not a substring of ABCD
9  */
10
11 import java.util.Scanner;
12
13 public class Main {
14
15     public static void main(String[] args) {
16         Scanner input = new Scanner(System.in);
17
18         String str = input.nextLine();
19
20         // 已確認：切割後的空白不會存進 array
21         String[] array = str.split(" ");
22
23         // 原來可以用 str.contains 去找子字串
24         // 我還以為只能找字元..
25         if(array[0].contains(array[1])){
26             System.out.println(array[1] + " is a
27             substring of " + array[0]);
28         }
29         else{
30             System.out.println(array[1] + " is not a
31             substring of " + array[0]);
32         }
33     }
34 }

```

1.10 prime or gcd

```

1  /*
2  請撰寫一程式，讀取一個整數，並以遞增的順序顯示所有的因數
   比方說，假設輸入整數為120，此程式的輸出結果應該是要2
3  但當輸入的整數為質數時，需輸出1與輸入的質數。
4  此題假設1為質數
5  */
6
7  import java.util.Scanner;
8  import java.util.Vector;
9
10 public class Main {
11
12     public static void main(String[] args) {
13         boolean[] prime = new boolean[10000000+5];
14         Scanner input = new Scanner(System.in);
15
16         int n = input.nextInt();
17
18         for( int i = 2; i <= n; i++ ){
19             if(!prime[i]){
20                 for( int j = i * i; j <= n; j += i ){
21                     prime[j] = true;
22                 }
23             }
24         }
25
26         if( !prime[n] ){
27             System.out.println("1 " + n );
28         }
29         else{
30             int num = n;
31             for( int i = 2; i < n; i++ ){
32                 while( num % i == 0 ){
33                     num /= i;
34                     System.out.print( num != 1 ? i +
35                                     " " : i + "\n" );
36                 }
37                 if( num == 1 ){
38                     break;
39                 }
40             }
41         }
42     }
}

```

1.11 pi calculation

```

1  /*
2  pi = 4( 1 - 1/3 + 1/5 - 1/7 + ... + (-1)^i+1/(2*i-1)
3  */
4
5  import java.util.Scanner;
6
7  public class Main {
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11         int n = input.nextInt();
12
13         double pi = 0;
14         for( int i = 1; i <= n; i++ ){
15             // 依據題意
16             pi += Math.pow(-1.0, (double)i+1)/(2*i-1);
17         }
18         System.out.println(pi*4);
19     }
20 }
21

```

1.12 factorial

```

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

```

```

3  */
4
5  import java.util.Scanner;
6
7  public class Main {
8
9     public static void main(String[] args) {
10         // 設個陣列存每階的值
11         double[] e = new double[10000000+5];
12         e[0] = 1;
13         Scanner input = new Scanner(System.in);
14         long n = input.nextLong();
15
16         double ans = 1;
17         for( long i = 1; i <= n; i++ ){
18             // i! = i*(i-1)!
19             e[(int)i] = i * e[(int)i-1];
20             ans += 1/e[(int)i];
21         }
22         System.out.println(ans);
23     }
24 }

```

1.13 Almanac

```

1  /*
2  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，
3  */
4
5  import java.util.Scanner;
6
7  public class Main {
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11         int y, week, k = 0;
12         String[] month = { "January", "February",
13                             "March", "April", "May", "June", "July",
14                             "August", "September", "October",
15                             "November", "December" };
16         int[] days = {31, 28, 31, 30, 31, 30, 31, 31,
17                       30, 31, 30, 31};
18
19         y = input.nextInt();
20         week = input.nextInt();
21
22         boolean flag = false;
23         for( int i = 0; i < 12; i++ ){
24             System.out.println( " " +
25                                 month[i] + " " + y + "\n" +
26                                 "-----\n"
27                                 + " Sun Mon Tue Wed Thu Fri
28                                   Sat");
29
30             k = 0;
31             for( int j = 1; j <= days[i]; j++){
32                 if( !flag && k != week ){
33                     System.out.print( " " );
34                     continue;
35                 }
36
37                 flag = true;
38                 if( k <= 6 ){
39                     System.out.printf( "%4d", j++);
40                 }
41                 else{
42                     System.out.printf( "\n%4d", j++);
43                     k = 0;
44                 }
45             }
46             flag = false;
47             if( k > 6 ){
48                 k = 0;
49             }
50             week = k;
51             System.out.print( i!= 11 ? "\n\n" : "\n");
52         }
53     }
54 }

```

```

44     }
45 }
46
47 }
30     }
31 }

```

1.14 perfect number

```

1  /*
2  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
3  本題不需任何輸入，請嘗試輸出所有 0 < num < 10000 的完美數
4  */
5
6  import java.util.Vector;
7
8  public class Main {
9
10     public static void main(String[] args) {
11
12         // 用 vector 存因數
13         Vector<Integer> v = new Vector<Integer>();
14         for( int i = 1; i < 10000; i++ ){
15             for( int j = 1; j < i; j++ ){
16                 if( i % j == 0 ){
17                     v.add(j);
18                 }
19             }
20             // 加總公因數
21             int count = 0;
22             for (Integer integer : v) {
23                 count += integer;
24             }
25             // 是否為完美數
26             if( count == i ){
27                 System.out.println(i);
28             }
29             v.clear();
30         }
31     }
32 }

```

1.15 maximum appear times

```

1  /*
2  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
3  */
4
5  import java.util.Scanner;
6  import java.util.Vector;
7
8  public class Main {
9
10     public static void main(String[] args) {
11         Scanner input = new Scanner(System.in);
12         Vector<Integer> num = new Vector<Integer>();
13
14         // 用 vector 來放輸出
15         while(input.hasNextInt()){
16             int n = input.nextInt();
17             num.add(n);
18         }
19         // 找最大值和出現次數
20         int maxi = 0, count = 1;
21         for (Integer integer : num) {
22             if (integer > maxi) {
23                 count = 1;
24                 maxi = integer;
25             } else if (integer == maxi) {
26                 count++;
27             }
28         }
29         System.out.println(maxi + " " + count);

```

1.16 leap year

```

1  /*
2  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 -
3  */
4
5  import java.util.Scanner;
6
7  public class Main {
8
9     public static void main(String[] args) {
10         Scanner input = new Scanner(System.in);
11
12         while(input.hasNextInt()){
13             int n = input.nextInt();
14             if( n < 0 ){
15                 break;
16             }
17             // 閏年的規律
18             else if( n % 4 == 0 && n % 100 != 0 ){
19                 System.out.println("It is leap
20                                     year.");
21             }
22             else {
23                 if (n % 400 == 0) {
24                     System.out.println("It is leap
25                                         year.");
26                 } else {
27                     System.out.println("It is not
28                                         leap year.");
29                 }
30             }
31         }
32     }
33 }

```

2 exam

2.1 BigDecimal Calculator

```

1  /*
2  Problem
3  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:
4  3.5 * 4 + 3 = 17
5
6  Requirement
7  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)
8
9  Input
10 There are many input lines. Each line has an input
   expression Exp. There are three operands and two
   operators in Exp.
11
12 Output

```

```

13  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)
14  */
15
16  import java.math.BigDecimal;
17  import java.math.RoundingMode;
18  import java.util.Arrays;
19  import java.util.Scanner;
20  import java.util.StringTokenizer;
21
22  public class Main {
23
24      public static BigDecimal calculate(String exp){
25
26          StringTokenizer st = new
              StringTokenizer(exp, "0123456789. ");
27          StringTokenizer ma = new StringTokenizer(exp,
              "+-*/% ");
28
29          String[] operator = new String[5];
30          String[] digits = new String[5];
31          Arrays.fill(operator, "0");
32          Arrays.fill(digits, "0");
33
34          int i = 0;
35          boolean flag = false;
36          while( st.hasMoreTokens() ){
37
38              if( exp.charAt(0) == '-' && !flag ){
39                  String haha = st.nextToken();
40                  flag = true;
41                  continue;
42              }
43              else{
44                  operator[i] = st.nextToken();
45                  i++;
46              }
47          }
48
49          int j = 0;
50          while( ma.hasMoreTokens() ){
51              digits[j] = ma.nextToken();
52              j++;
53          }
54
55          boolean e = false;
56          for( int k = 0; k < j-1; k++ ){
57              if(exp.charAt(0) == '-' ||
                  operator[k].equals("0") ){
58                  break;
59              }
60              System.out.print(digits[k] + " ");
61              System.out.print(operator[k] + " ");
62              e = true;
63          }
64          if( e ){
65              System.out.print(digits[j-1] + " = ");
66          }
67
68          boolean yep = false;
69          if( i == 2 ){
70              if( operator[1].equals("*") ||
                  operator[1].equals("/") ||
                  operator[1].equals("%") ){
71                  if( operator[0].equals("+") ||
                      operator[0].equals("-") ){
72                      yep = true;
73                      String tp;
74                      tp = operator[1];
75                      operator[1] = operator[0];
76                      operator[0] = tp;
77
78                      String str;
79                      str = digits[1];
80                      digits[1] = digits[0];

```

```

81                      digits[0] = str;
82
83                      str = digits[2];
84                      digits[2] = digits[1];
85                      digits[1] = str;
86                  }
87              }
88          }
89      }
90      BigDecimal sum = new BigDecimal(digits[0]);
91      if( exp.charAt(0) == '-' ){
92          BigDecimal tmp = BigDecimal.ZERO;
93          sum = tmp.subtract(sum);
94      }
95
96      i = 0;
97
98      for( int k = 1; k < j; k++ ){
99          BigDecimal b = new BigDecimal(digits[k]);
100         if( operator[i].equals("+") ){
101             sum = sum.add(b);
102         }
103         else if( operator[i].equals("-") ){
104             if(yep){
105                 sum = b.subtract(sum);
106             }
107             else{
108                 sum = sum.subtract(b);
109             }
110         }
111         else if( operator[i].equals("*") ){
112             sum = sum.multiply(b);
113         }
114         else if( operator[i].equals("/") ){
115             sum = sum.divide(b, 1,
116                 RoundingMode.CEILING);
117         }
118         else if( operator[i].equals("%") ){
119             sum = sum.remainder(b);
120         }
121         i++;
122     }
123
124     return sum;
125 }
126
127 public static void main(String[] args) {
128     Scanner input = new Scanner(System.in);
129
130     while( input.hasNext() ){
131         String exp = input.nextLine();
132         BigDecimal ans = calculate(exp);
133         // BigDecimal 四捨五入的寫法 .setScale(
            想要留取的位數, RoundingMode.HALF_UP)
134         System.out.println(ans.setScale(1,
            RoundingMode.HALF_UP));
135     }
136 }
137 }

```

2.2 Magic Square

```

1  /*
2  Problem
3  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.
4
5  For the case n = 3,

```

```

6  M. Square           Rows           Columns
   Diagonals
7  4 9 2              4+9+2 = 15      4+3+8 = 15
   2+5+8 = 15
8  3 5 7              3+5+7 = 15      9+5+1 = 15
   4+5+6 = 15
9  8 1 6              8+1+6 = 15      2+7+6 = 15
10
11 Input
12 Each line contains an Integer N denoting an N * N
   Magic Square.
13
14 Output
15 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。
16 */
17
18 import java.util.Arrays;
19 import java.util.Scanner;
20
21 public class Main {
22
23     public static void main(String[] args) {
24         Scanner input = new Scanner(System.in);
25         boolean flag = false;
26
27         while(input.hasNext()){
28             int n = input.nextInt();
29             if( flag ){
30                 System.out.println();
31             }
32
33             if( n % 2 == 0 ){
34                 System.out.println("It is not an odd
35                                     number.");
36                 flag = true;
37                 continue;
38             }
39
40             int[][] square = new int[100+5][100+5];
41             for( int i = 0; i < n; i++ ){
42                 Arrays.fill( square[i], 0);
43             }
44
45             int r = n-1, c = n/2;
46             square[r][c] = 1;
47             for( int i = 2; i <= n*n; i++ ){
48
49                 int rn = r+1;
50                 int cn = c+1;
51                 if( rn >= n ){
52                     rn = 0;
53                 }
54                 if( cn >= n ){
55                     cn = 0;
56                 }
57                 if( square[rn][cn] != 0 ){
58                     rn = r-1;
59                     cn = c;
60                     if( rn < 0 ){
61                         rn = n-1;
62                     }
63                 }
64                 square[rn][cn] = i;
65                 r = rn;
66                 c = cn;
67             }
68             int sum = 0;
69             for( int i = 0; i < n; i++ ){
70                 sum += square[0][i];
71             }
72             System.out.println(sum);
73             for( int i = 0; i < n; i++ ){
74                 for( int j = 0; j < n; j++ ){

```

```

74                 System.out.printf("%5d",
75                                     square[i][j]);
76             }
77             System.out.println();
78         }
79         flag = true;
80     }
81 }
82 }

```

2.3 Prime Factorization

```

1  /*
2  Problem
3  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。
4
5  Requirement
6  請撰寫以下兩個 static methods：(未依規定，以 0 分計)
7
8  1. boolean [] PrimeArray(long N) {...}
9     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;
10
11 2. String PrimeFactorization(long N) {...}
12     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" .
13
14 Input
15 輸入有多列，每列有個整數 N，最多 1000 列。
16
17 Output
18 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。
19 */
20
21 import java.util.Arrays;
22 import java.util.Scanner;
23
24 public class Main {
25
26     public static boolean [] PrimeArray(long N){
27
28         boolean[] prime = new boolean[1000000];
29         Arrays.fill(prime, true);
30
31         for( long i = 2; i <= N; i++ ){
32             if(prime[(int)i]){
33                 for( long j = i * i; j <= N; j += i ){
34                     prime[(int)j] = false;
35                 }
36             }
37         }
38         return prime;
39     }
40
41     public static String PrimeFactorization(long N){
42
43         int[] num = new int[1000000];
44         Arrays.fill( num, 0 );
45
46         long count = N;
47         for( int i = 2; i < N; i++ ){
48             while( count % i == 0 ){
49                 num[i]++;

```



```

49         count /= i;
50     }
51     if( count == 1 ){
52         break;
53     }
54 }
55
56 // 因數轉字串處理
57 String[] tp = new String[1000000];
58 int j = 0;
59 for( int i = 2; i < N; i++ ){
60     if( num[i] != 0 ){
61         tp[j++] = Integer.toString(i);
62         if( num[i] != 1 ){
63             tp[j++] = "^";
64             tp[j++] = Integer.toString(num[i]);
65         }
66         tp[j++] = " * ";
67     }
68 }
69
70 // 合併字串 用 str1.concat(str2)
71 // 因為最後會多一組 * 所以 j-1
72 String result = "";
73 for( int i = 0; i < j-1; i++ ){
74     result = result.concat(tp[i]);
75 }
76 return result;
77 }
78
79 public static void main(String[] args) {
80     Scanner input = new Scanner(System.in);
81     int max = 0;
82     int[] N = new int[1000+5];
83     int k = 0;
84
85     while( input.hasNextLong() ) {
86
87         N[k] = input.nextInt();
88         if (N[k] > max) {
89             max = N[k];
90         }
91         k++;
92     }
93
94
95     // 可以用 method 回傳陣列
96     // 但也要用陣列接他的回傳值
97     boolean[] A = PrimeArray(max);
98     System.out.println(max + " " +
99         (int)Math.sqrt(max) );
100     for( int o = 0; o < k; o++ ){
101         if( A[N[o]] ){
102             System.out.println(N[o]);
103         }
104         else{
105             String factorization =
106                 PrimeFactorization(N[o]);
107             System.out.println(factorization);
108         }
109     }

```

3 homeworks

3.1 Print Prime Numbers

```

1  /*
2  Write a program to input an integer N and print all
   prime numbers less than N.
3  Please print 10 prime numbers in each line.

```

```

4  */
5
6  import java.util.Scanner;
7  import java.util.Vector;
8
9  public class Main {
10
11     public static void main(String[] args) {
12         Scanner input = new Scanner(System.in);
13         int n = input.nextInt();
14
15         // 建質數表
16         // java 中沒有全域變數
17         // 所以這邊建表目前是每次輸入都重建一次
18         // 但不夠漂亮
19         boolean[] prime = new boolean[10000000+5];
20
21         // 用 vector 存質數們
22         Vector<Integer> v = new Vector<Integer>();
23         for( int i = 2; i <= n; i++ ){
24             if(!prime[i]){
25                 v.add(i);
26                 for( int j = i * i; j <= n; j += i ){
27                     prime[j] = true;
28                 }
29             }
30         }
31
32         for( int i = 0, k = 0; i < v.size(); i++, k++ ){
33             if( k == 9 || i == v.size() - 1 ){
34                 System.out.println(v.get(i));
35                 // 把 k 歸回去 -1 是因為下次輪迴會先
36                 // k++
37                 k = -1;
38             }
39             else{
40                 System.out.print(v.get(i) + " ");
41             }
42         }
43         v.clear();
44     }
45 }

```

3.2 P1 Infix Notation to Postfix Notation

```

1  /*
2  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
3  1. 使用String類別 & 字串輸入
4  2. 使用StringTokenizer類別 或
   String類別之split()method來切割中序運算式。
5  3. 使用String [] array 儲存所有tokens。
6  4. 將數字token轉換成實數，加總後輸出總和。Hint: Use
   BigDecimal in java.math package
7
8  輸入
9  每組測資會給予不一定長度的算式，符號包含+,-,*,/,(),%,算式中數
10
11  輸出
12  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。
13  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。
14  第三行印出運算元的總和，小事點以下印三位。
15  */
16
17  import java.math.BigDecimal;
18  import java.math.RoundingMode;
19  import java.util.Scanner;
20
21  public class Main {
22
23     public static void main(String[] args) {
24         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));
}
}

```

```

15 import java.util.Scanner;
16 import java.util.Stack;
17
18 public class Main {
19
20     public static int icpfunc(String check){
21         // icp 是該 token 的順序
22         // icp -> +- 1 -> */% 2 -> ( 4
23         int icp;
24         if(check.equals("(")){
25             icp = 4;
26             return icp;
27         }
28         else if(check.equals("*") ||
29             check.equals("/") || check.equals("%") ){
30             icp = 2;
31             return icp;
32         }
33         else{
34             icp = 1;
35             return icp;
36         }
37     }
38     public static int ispfunc(String check){
39         // isp 是堆疊最上層 stack.peek 中的順序
40         // isp -> ( 0 -> +- 1 -> */% 2
41         int isp;
42         if(check.equals("(")){
43             isp = 0;
44             return isp;
45         }
46         else if(check.equals("*") ||
47             check.equals("/") || check.equals("%") ){
48             isp = 2;
49             return isp;
50         }
51         else{
52             isp = 1;
53             return isp;
54         }
55     }
56
57     public static void main(String[] args) {
58         Scanner input = new Scanner(System.in);
59         String str = input.nextLine();
60         String token = "+-*/%";
61
62         // 用 split 切割字串
63         String[] array =
64             str.split("(\\+|\\-|\\*|\\/|\\%)|(?<=\\+|\\-|\\*|\\/|\\%)");
65
66         // 使用 stack 儲存運算子
67         Stack<String> stack = new Stack<>();
68         int icp = 0, isp = 0;
69
70         for (String s : array) {
71             // 如果是 ) 則輸出所有 ( 以前的 stack
72             // 內的運算子
73             // 但 ( 都不輸出
74             if (s.equals(")")) {
75                 while (!stack.peek().equals("(")) {
76                     System.out.print(stack.peek() + "
77 ");
78                     stack.pop();
79                 }
80                 // 把 ) 也 pop 掉
81                 stack.pop();
82             }
83             // 如果是 + - * / %
84             else if (token.contains(s)) {
85                 // 運算此運算子的 icp 和 當前 stack
86                 // 頂端的運算子的 isp
87                 icp = icpfunc(s);
88                 if (stack.empty()) {
89                     stack.push(s);
90                 }
91             }
92         }
93     }
94 }

```

3.3 P2 Infix Notation to Postfix Notation

```

1  /*
2  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
3
4  程式要求
5  1. 必須使用 java.util.Stack 實作，否則不予計分。
6  2. 請利用 part1 求得的 token 字串陣列轉後序
7
8  輸入
9  每組測資會給予不一定長度的算式，符號包含 +, -, *, /, (, ), %, 算
10
11  輸出
12  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。
13  格式請詳看 sample output
14  */

```

```

72         System.out.print(stack.peek() + "
73         ");
74         stack.pop();
75     }
76     // 把 ) 也 pop 掉
77     stack.pop();
78 }
79 // 如果是 + - * / %
80 else if (token.contains(s)) {
81     // 運算此運算子的 icp 和 當前 stack
82     // 頂端的運算子的 isp
83     icp = icpfunc(s);
84     if (stack.empty()) {
85         stack.push(s);
86     }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```



```

97 Stack<String> stack = new Stack<>();
98 int icp = 0, isp = 0, i = 0;
99
100 for (String str : postfixexpr) {
101     // 如果是 ) 則輸出所有 ( 以前的 stack
    內的運算子
102     // 但 ( 都不輸出
103     if (str.equals("(")) {
104         while (!stack.peek().equals("(")) {
105             ans[i++] = stack.peek();
106             stack.pop();
107         }
108         // 把 ) 也 pop 掉
109         stack.pop();
110     }
111     // 如果是 + - * / %
112     else if (token.contains(str)) {
113         // 運算此運算子的 icp 和 當前 stack
        頂端的運算子的 isp
114         icp = icpfunc(str);
115         if (stack.empty()) {
116             stack.push(str);
117         }
118         else {
119             isp = ispfunc(stack.peek());
120             if (icp > isp) {
121                 stack.push(str);
122             } else {
123                 while (icp <=
                    ispfunc(stack.peek())) {
124                     ans[i++] = stack.peek();
125                     stack.pop();
126                     if (stack.empty()) {
127                         break;
128                     }
129                 }
130                 stack.push(str);
131             }
132         }
133     }
134     // 如果是運算元則直接輸出
135     else {
136         ans[i++] = str;
137     }
138 }
139 // 最後因為有可能 array
    在切割的時候最後一個值是運算元
140 // 那樣就還會有運算子在 stack 還沒輸出
    所以這裡在確認一次
141 while(!stack.empty()){
142     ans[i++] = stack.peek();
143     stack.pop();
144 }
145 return ans;
146 }
147
148 // 相當於老師要求的 cccc
149 public static BigDecimal caculating (String s){
150     String answer = "";
151     String[] frompostfixe;
152     Stack<String> stack = new Stack<>();
153     String token = "+-*/%";
154
155     // 呼叫 postfix method 去排序後續
156     // 而 postfix method 會先去呼叫 splitmethod
        method 去切割字串
157     // 所以 caculating 可以 call postfixe 和
        splitmethod
158     frompostfixe = postfixe(s);
159
160     for (String str : frompostfixe) {
161
162         // 如果跑到 frompostfixe
        沒放後續字串的地方了 就 break 掉
163         if (str == null ){
164             break;
165         }
166
167         // 當碰到運算子時 先到 stack
        去找最上面的兩個運算元
168         // 分別讀取後 將兩者皆 pop 出來 (
        因為如此一來他們用過了 )
169         // 對他們進行該運算子的運算
170         // 最後將目前的運算結果 push 進 stack 裡
171         //
        這樣就算之後有更要先乘除後加減的數也不會影響到了
172         if (token.contains(str)) {
173
174             BigDecimal top = BigDecimal.valueOf(
                Double.parseDouble(stack.peek())
            );
175             stack.pop();
176             BigDecimal sec = BigDecimal.valueOf(
                Double.parseDouble(stack.peek())
            );
177             stack.pop();
178
179             if (str.charAt(0) == '+'){
180                 sec = sec.add( top );
181                 stack.push(sec.toString());
182             }
183             else if (str.charAt(0) == '-'){
184                 sec = sec.subtract( top );
185                 stack.push(sec.toString());
186             }
187             else if (str.charAt(0) == '*'){
188                 sec = sec.multiply( top );
189                 stack.push(sec.toString());
190             }
191             else if (str.charAt(0) == '/'){
192                 sec = sec.divide( top, 2,
                    RoundingMode.HALF_UP );
193                 stack.push(sec.toString());
194             }
195             else if (str.charAt(0) == '%'){
196                 sec = sec.remainder( top );
197                 stack.push(sec.toString());
198             }
199             // 判斷目前運算的結果是多少 (
        無論是暫時的 還是最終結果
200             answer = stack.peek();
201         }
202         // 如果是運算子就直接 push 進 stack
203         else{
204             stack.push(str);
205         }
206     }
207     // 最後回傳 Bigdecimal 型態的 answer
208     // 根據老師要求的 method 型別
209     return BigDecimal.valueOf(
        Double.parseDouble(answer) );
210 }
211 }

```

4 old exams

4.1 PA Palindromic Prime

```

1 /*
2 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
3
4 Input Format
5 每組測資為一整數 N，請注意本題輸入數字的數目不定。
6
7 Output Format

```

```

8 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
9
10 Technical Specification
11 •  $0 \leq N \leq 100$ 
12 */
13
14 import java.util.Arrays;
15 import java.util.Scanner;
16 import java.util.Vector;
17
18 public class Main {
19
20     public static void main(String[] args) {
21
22         // 直接在最開始就建好質數表
23         boolean[] prime = new boolean[100000+5];
24         Arrays.fill(prime, false);
25         Vector<String> v = new Vector<String>();
26
27         for( int i = 2; i <= 100000; i++ ){
28
29             if(!prime[i]){
30                 // 因為 100000 * 100000 會超出 int
31                 // 所以用 long
32                 // i * i 是 int -> 強制轉型 (long)
33                 for(long j = (long) i * i; j <=
34                     100000; j += i ) {
35                     // prime 的指標是放 int
36                     prime[(int) j] = true;
37                 }
38
39                 // 先把 i 轉成 String (
40                 // 因為這樣比較方便 reverse )
41                 String str = String.valueOf(i);
42
43                 // 使用 StringBuilder 宣告 rev ->
44                 // 因為 StringBuilder 才有
45                 // .reverse() 可以用
46                 StringBuilder rev = new
47                     StringBuilder();
48                 // 這裡 append (附加) str 的值給 rev
49                 rev.append(str);
50                 // reverse
51                 rev.reverse();
52
53                 // 兩字串判斷相等在 java 要用 .equals
54                 // 不能直接 == (原因跟記憶體位址有關
55                 // 然後因為 rev 是 StringBuilder 但
56                 // .equals 是 String 的 所以要
57                 // .toString()
58                 if( str.equals(rev.toString()) ){
59                     v.add(str);
60                 }
61             }
62         }
63
64         Scanner input = new Scanner(System.in);
65
66         // 重複輸入寫法
67         while( input.hasNextInt() ){
68             int n = input.nextInt();
69
70             if( n == 0 ){
71                 System.out.println();
72                 continue;
73             }
74
75             for( int i = 0, k = 0; i < n; i++, k++ ){
76                 if( k == 9 || i == n - 1 ){
77                     System.out.println(v.get(i));
78                     k = -1;
79                 }
80             }
81         }
82     }
83 }

```

```

73         System.out.print(v.get(i) + " ");
74     }
75 }
76 }
77 }
78 }

```

4.2 You can say 11

```

1  /*
2  給你一個正整數 N，判定它是否是 11 的倍數。提示：
3  在這裡，我們需要用到一個關於 11 倍
4  數的小常識，那就是：11
5  倍數的「奇數位數字和」與「偶數位數字和」兩者的差必定為
6  11 的
7  倍數。
8
9  Input Format
10 每列資料有一個正整數 N，N 最大可能到 1000 位數。若 N
11  = 0 代表輸入結束
12
13  Output Format
14 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
15 11 的倍數。輸出格式請
16
17 本題必須使用 char[] or String 儲存數入之正整數。
18  */
19
20 import java.util.Scanner;
21
22 public class Main {
23
24     public static void main(String[] args) {
25         Scanner input = new Scanner(System.in);
26
27         while( input.hasNext() ){
28             String str = input.nextLine();
29             if( str.equals("0") ){
30                 break;
31             }
32
33             int even = 0, odd = 0;
34             for( int i = str.length()-1; i >= 0; i -=
35                 2 ){
36                 // 要抓字串中的每個 char 要
37                 // str.charAt()
38                 // 要把 char 轉 int 蠻複雜的
39                 // 要先 String.valueOf() 然後
40                 // Integer.parseInt
41                 odd += Integer.parseInt(
42                     String.valueOf( str.charAt(i) ) );
43             }
44             for( int i = str.length()-2; i >= 0; i -=
45                 2 ){
46                 even += Integer.parseInt(
47                     String.valueOf( str.charAt(i) ) );
48             }
49
50             int sub = Math.abs( even - odd );
51             if( sub % 11 != 0 ){
52                 System.out.println( odd + " " + even
53                     + "\n" + str + " is not a
54                     multiple of 11." );
55             }
56             else{
57                 System.out.println( odd + " " + even
58                     + "\n" + str + " is a multiple of
59                     11." );
60             }
61         }
62     }
63 }

```

4.3 The BigInteger GCD LCM and Prime

```

1  /*
2  請利用 BigInteger 解題，不然不予給分
3
4  Problem
5  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)。
6
7  Input
8  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)
9
10 Output
11 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
12 */
13
14 import java.math.BigInteger;
15 import java.util.Scanner;
16
17 public class Main {
18
19     public static void main(String[] args) {
20         Scanner input = new Scanner(System.in);
21         BigInteger M, N, G, L;
22
23         while( input.hasNextBigInteger() ) {
24
25             M = input.nextBigInteger();
26             N = input.nextBigInteger();
27
28             // BigInteger 內建函數 gcd
29             G = M.gcd(N);
30             // L = M * N / G
31             L = M.multiply(N).divide(G);
32             System.out.print(G + " " + L + "\n");
33
34             for( int i = 0; i < 5; i++){
35
36                 // .nextProbablePrime ->
37                 // 返回一個大於該 BigInteger 的質數
38                 System.out.print( i == 4 ?
39                     L.nextProbablePrime() + "\n" :
40                     L.nextProbablePrime() + " ");
41                 L = L.nextProbablePrime();
42             }
43         }
44     }
45 }

```

4.4 Latin Square

```

1  /*
2  Description
3  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.
4  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.
5
6  Input
7  Enter number n, and enter n rows of letters separated
    by spaces.
8  0 < n <= 26, all the letters in square are A ~ Z.
9  Input is terminated by a set where n = -1. This set
    should not be processed.
10
11 Output

```

```

12 Recognize if it is Latin Square or not. There are 2
    different kinds of your answers:
13 The input array is a Latin square
14 The input array is not a Latin square
15 Overall, just check if it is Latin square or not.
16
17 Sample Input
18 4
19 A B C D
20 B A D C
21 C D B A
22 D C A B
23
24 3
25 B C A
26 B A C
27 C B A
28
29 3
30 B A C
31 A C B
32 C B A
33
34 4
35 D C B A
36 J A V A
37 U C C U
38 P P A P
39
40 -1
41
42 Sample Output
43 The input array is a Latin square
44
45 The input array is not a Latin square
46
47 The input array is a Latin square
48
49 The input array is not a Latin square
50 */
51
52 import java.util.Arrays;
53 import java.util.Scanner;
54
55 public class Main {
56
57     public static void main(String[] args) {
58         Scanner input = new Scanner(System.in);
59         boolean newline = false;
60
61         while( input.hasNextInt() ){
62
63             int n = input.nextInt();
64             if( n < 0 ){
65                 break;
66             }
67             // 控制最後不要換行
68             if( newline ){
69                 System.out.println();
70             }
71
72             int[] count = new int[26+5];
73             char[][] square = new char[26+5][26+5];
74             boolean[][] row = new boolean[26+5][26+5];
75             boolean[][] col = new boolean[26+5][26+5];
76
77             // 陣列初始化
78             for( int i = 0; i < 26; i++){
79                 Arrays.fill(count, 0);
80                 Arrays.fill(square[i], '0');
81                 Arrays.fill(row[i], false);
82                 Arrays.fill(col[i], false);
83             }
84
85             for( int i = 0; i < n; i++ ){
86                 for( int j = 0; j < n; j++ ){

```

```

87 // char 的讀取寫法
88     .next().charAt(0) -> 這個 0
89     是固定的
90     因為單字元的索引值就是 0
91     square[i][j] =
92         input.next().charAt(0);
93 // 計算每位是否接是出現 n 次 ->
94     每行都出現一次
95     count[ square[i][j] - 'A' ]++;
96 }
97 // 吃最後的換行
98 String space = input.nextLine();
99
100 boolean flag = false;
101 for( int i = 0; i < n; i++ ) {
102     for (int j = i; j < n; j++) {
103         if (row[i][square[i][j] - 'A'] ||
104             row[j][square[j][i] - 'A'] ||
105             col[i][square[j][i] - 'A'] ||
106             col[j][square[i][j] - 'A']) {
107             flag = true;
108             break;
109         }
110         row[i][square[i][j] - 'A'] = true;
111         row[j][square[j][i] - 'A'] = true;
112         col[i][square[j][i] - 'A'] = true;
113         col[j][square[i][j] - 'A'] = true;
114     }
115     if (flag) {
116         break;
117     }
118 }
119 for( int i = 0; i < 26; i++ ){
120     if( count[i] != n && count[i] != 0 ){
121         flag = true;
122         break;
123     }
124 }
125 if(flag){
126     System.out.println("The input array
127         is not a Latin square");
128 }
129 else{
130     System.out.println("The input array
131         is a Latin square");
132 }
133 newline = true;
134 }
135 }
136 }

```

5 practice

5.1 star pyramid

```

1  /*
2  3
3   *
4  ***
5 *****
6  */
7
8  import java.util.Scanner;
9
10 public class Main {
11
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14         int n = input.nextInt();
15

```

```

16         for( int i = 1; i <= n; i++ ){
17
18             for( int j = n-i; j > 0; j-- ){
19                 System.out.printf(" ");
20             }
21
22             for( int j = 0; j < i; j++ ){
23                 System.out.printf("*");
24             }
25             for( int j = 1; j < i; j++ ){
26                 System.out.printf(" ");
27             }
28
29             System.out.println();
30
31         }
32     }
33 }

```

5.2 Nine Nine Multiplication Table

```

1  /*
2  請列出所有九九乘法表。
3  example:
4  1 * 1 = 1 1 * 2 = 2 1 * 3 = 3 1 * 4 = 4 1 * 5 = 5 ...
5  1 * 9 = 9\n
6  *每個公式中間都有一個空格
7  *每行最後都沒有空格，但有換行符號
8  *每個公式的答案都是 %2d <-重點
9  *可以利用 System.out.printf(String , value ,.....);
10 */
11
12 public class Main {
13
14     public static void main(String[] args) {
15
16         for( int i = 1; i < 10; i++ ){
17             for( int j = 1; j < 10; j++ ){
18
19                 if( j == 9 ){
20                     System.out.printf( "%d * %d =
21                         %2d\n", i, j, i * j );
22                     continue;
23                 }
24
25                 System.out.printf( "%d * %d = %2d ",
26                     i, j, i * j );
27             }
28         }
29     }

```

5.3 prime number

```

1  /*
2  請輸入一個 Integer 並輸出小於其的所有質數
3  每行答案中間都有空格
4  每行最後都為換行符號(沒有空格)
5  */
6
7  import java.util.*;
8
9  public class Main {
10
11     public static void main(String[] args) {
12
13         Scanner input = new Scanner(System.in);
14         int n = input.nextInt();
15
16         boolean prime[] = new boolean[1000];
17

```

```

18 // java 的 vector 寫法似乎是如此 :D
19 // <> 內不能寫 int 要寫 Integer
20 Vector<Integer> v = new Vector();
21
22 // Arrays.fill 類似 C++ 的 memset( a, false,
23 // sizeof(a) );
24 Arrays.fill( prime, false );
25
26 for( int i = 2; i < n; i++ ){
27     if( !prime[i] ){
28         for( int j = i * i; j < n; j += i ){
29             prime[j] = true;
30         }
31         // 像 C++ 的 v.push_back(i)
32         v.add(i);
33     }
34 }
35
36 int i;
37 for( i = 0; i < v.size()-1; i++ ){
38     // 不能直接寫 v[i] 要寫 v.get(i)
39     System.out.printf( "%d ", v.get(i) );
40 }
41
42 System.out.printf( "%d\n", v.get(i) );
43 }
44 }
45 }
46 }
47 }

```

```

37 first += Integer.parseInt(
38     String.valueOf( str.charAt(i)
39     ) );
40 flag = false;
41 }
42 else{
43     second += Integer.parseInt(
44         String.valueOf( str.charAt(i)
45         ) );
46     flag = true;
47 }
48 }
49
50 // java 的 abs 前面要加 Math.
51 if( Math.abs( first - second ) % 11 == 0
52 ){
53     System.out.printf( "%s is a multiple
54         of 11.\n", str );
55 }
56 else{
57     System.out.printf( "%s is not a
58         multiple of 11.\n", str );
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

5.4 factor

```

1 /*
2 你的任務是，給你一個正整數 N，判定它是否是 11 的倍數。
3
4 Input
5 每列資料有一個正整數 N，N 最大可能到 1000 位數。
6 若 N = 0 代表輸入結束。
7
8 Output
9 對每個輸入的數，輸出是否為 11 的倍數。輸出格式請參考
10 Sample Output。
11 */
12 import java.util.Scanner;
13
14 public class Main {
15
16     public static void main(String[] args) {
17
18         Scanner input = new Scanner(System.in);
19
20         // java 中要用 .hasNext() 來判斷是否輸入結束
21         while( input.hasNext() ){
22
23             // String 的 input 是 .nextLine()
24             String str = input.nextLine();
25
26             // java 用 ==
27             // 進行比較的時候，比較的是他們在記憶體中的存放地址
28             // 而 String,Integer,Date 這些類中 equals
29             // 可以用來做比較
30             if( str.equals("0") ){
31                 break;
32             }
33
34             boolean flag = false;
35             int first = 0, second = 0;
36             for( int i = 0; i < str.length(); i++ ){
37
38                 if( flag ){
39

```