


```
61         System.out.println();
62
63         // BigDecimal 四捨五入的寫法 .setScale(
64             想要留取的位數, RoundingMode.HALF_UP)
65         System.out.println(ans.setScale(3,
66             RoundingMode.HALF_UP));
67     }
68 }
```

1.3 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 */
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     public static void main(String[] args) {
57         Scanner input = new Scanner(System.in);
58         String str = input.nextLine();
59         String token = "+-*/(%)";
60
61         // 用 split 切割字串

```

```
61 String[] array =  
    str.split("(\\?=\\\\+/-\\\\\\\\*/\\\\\\\\)\\\\\\\\(/%)/(?<=\\\\+/-\\\\\\\\|\\\\\\\\)");  
  
62  
63 // 使用 stack 儲存運算子  
64 Stack<String> stack = new Stack<>();  
65 int icp = 0, isp = 0;  
66  
67 for (String s : array) {  
68     // 如果是 ) 則輸出所有 ( 以前的 stack  
        內的運算子  
69     // 但 ( 都不輸出  
70     if (s.equals("(")) {  
71         while (!stack.peek().equals("(")) {  
72             System.out.print(stack.peek() + "  
式3           ");  
73             stack.pop();  
74         }  
75         // 把 ) 也 pop 掉  
76         stack.pop();  
77     }  
78     // 如果是 + - * / %  
79     else if (token.contains(s)) {  
80         // 運算此運算子的 icp 和 當前 stack  
            頂端的運算子的 isp  
81         icp = icpfunc(s);  
82         if (stack.empty()) {  
83             stack.push(s);  
84         }  
85         else {  
86             isp = ispfunc(stack.peek());  
87             if (icp > isp) {  
88                 stack.push(s);  
89             } else {  
90                 while (icp <=  
                    ispfunc(stack.peek())) {  
91                     System.out.print(stack.peek()  
                        + " ");  
92                     stack.pop();  
93                     if (stack.empty()) {  
94                         break;  
95                     }  
96                 }  
97                 stack.push(s);  
98             }  
99         }  
100     }  
101     // 如果是運算元則直接輸出  
102     else {  
103         System.out.print(s + " ");  
104     }  
105 }  
106 // 最後因為有可能 array  
    在切割的時候最後一個值是運算元  
107 // 那樣就還會有運算子在 stack 還沒輸出  
    所以這裡在確認一次  
108 while(!stack.empty()){  
109     System.out.print(stack.peek() + " ");  
110     stack.pop();  
111 }  
112 System.out.println();  
113 }  
114 }
```

1.4 P3 Infix Notation to Postfix Notation

```

1  /*
2  請利用作業一(Part
   2)中序轉後序-轉後序的答案輸出後續的算式，並輸出運算結果
3
4  輸入
   每組測資會給予不一定長度的算式，符號包含 +, -, *, /, (, ), %。
5
6
7  程式要求 (未依規定以0分記)

```

```

8 Write a Class XXXX (name it yourself) (which is not
  the Main class) for computing the input
  expression.
9 Class XXXX must include the following three static
  methods
10 String[] AAAA(String s) which returns an array of
  string tokens for the input expression s.
11 String[] BBBB(String s) which returns an array of
  string tokens (Postfix Expression) for the input
  expression s. Note that BBBB() can call AAAA().
12 BigDecimal CCCC(String s) which returns the computing
  result of input expression s. Note that CCCC()
  can call AAAA() and BBBB().
13 Please give a meaningful name for AAAA, BBBB, and
  CCCC.
14
15 輸出
16 輸出算式的答案(BigDecimal)
17 */
18
19 import java.math.BigDecimal;
20 import java.math.RoundingMode;
21 import java.util.Scanner;
22 import java.util.Stack;
23
24 public class Main {
25
26     public static void main(String[] args) {
27         Scanner input = new Scanner(System.in);
28         String str = input.nextLine();
29
30         BigDecimal output;
31         // 呼叫並使用另一個叫做 computing 的 class
32         // 的內部 method
33         output = computing.caculating(str);
34
35         System.out.println(output.setScale(2,
36             RoundingMode.HALF_UP));
37     }
38 }
39 // 同一份 java 檔案內只能有一個 public class
40 // 所以第二個直接叫 class 就好
41 class computing {
42
43     // 用來判斷後續先後順序的兩個 method
44     public static int icpfunc(String check){
45         // icp 是該 token 的順序
46         // icp -> +- 1 -> */% 2 -> ( 4
47         int icp;
48         if(check.equals("(")){
49             icp = 4;
50             return icp;
51         }
52         else if(check.equals("*") ||
53             check.equals("/") || check.equals("%")){
54             icp = 2;
55             return icp;
56         }
57         else{
58             icp = 1;
59             return icp;
60         }
61     }
62
63     public static int ispfunc(String check){
64         // isp 是堆疊最上層 stack.peek 中的順序
65         // isp -> ( 0 -> +- 1 -> */% 2
66         int isp;
67         if(check.equals("(")){
68             isp = 0;
69             return isp;
70         }
71         else if(check.equals("*") ||
72             check.equals("/") || check.equals("%")){
73             isp = 2;
74             return isp;
75         }
76     }
77 }

```

```

70     else{
71         isp = 1;
72         return isp;
73     }
74 }
75
76 // 等同於老師要求的第一個 method AAAA
77 // 把輸入的 s 做切割後放入陣列 arr 中
78 public static String[] splitmethod (String s){
79     // 用 split 切割字串
80     String[] arr =
81         s.split("(\\?=\\\\+|-|\\\\*|/|\\\\)\\\\(|%)|(\\?<=\\\\+|-|\\\\*/|");
82     return arr;
83 }
84
85 // 等同於老師要求的 BBBB
86 // 呼叫 splitmethod method
87 // 去切割字串並按照後續排序
88 public static String[] postfixe (String s){
89     String[] postfixearr;
90     String token = "+-*/(%)";
91
92     // 呼叫 splitmethod method 切割字串
93     // 用 postfixearr 接回傳的陣列裡
94     postfixearr = computing.splitmethod(s);
95
96     String[] ans = new String[postfixearr.length];
97
98     // 使用 stack 儲存運算子
99     Stack<String> stack = new Stack<>();
100     int icp = 0, isp = 0, i = 0;
101
102     for (String str : postfixearr) {
103         // 如果是 ) 則輸出所有 ( 以前的 stack
104         // 內的運算子
105         // 但 ( 都不輸出
106         if (str.equals(")") ) {
107             while (!stack.peek().equals("(")) {
108                 ans[i++] = stack.peek();
109                 stack.pop();
110             }
111             // 把 ) 也 pop 掉
112             stack.pop();
113         }
114         // 如果是 + - * / %
115         else if (token.contains(str)) {
116             // 運算此運算子的 icp 和 當前 stack
117             // 頂端的運算子的 isp
118             icp = icpfunc(str);
119             if (stack.empty()) {
120                 stack.push(str);
121             }
122             else {
123                 isp = ispfunc(stack.peek());
124                 if (icp > isp) {
125                     stack.push(str);
126                 } else {
127                     while (icp <=
128                         ispfunc(stack.peek())) {
129                         ans[i++] = stack.peek();
130                         stack.pop();
131                         if (stack.empty()) {
132                             break;
133                         }
134                     }
135                     stack.push(str);
136                 }
137             }
138         }
139         // 如果是運算元則直接輸出
140         else {
141             ans[i++] = str;
142         }
143     }

```

```

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 postfixe method 去排序後續
156     // 而 postfixe 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之前運算過的數
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 }

```

2 note

2.1 array

```

1 #define inf 10000
2
3
4 // Array 自動排序
5 Arrays.sort(arr);
6
7 // System.arraycopy(來源, 起始索引, 目的, 起始索引,
    複製長度)
8 System.arraycopy(arr1, 0, arr2, 0, arr1.length);

```

2.2 BigDecimal

```

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

```

2.3 date

```

1 /**
2  java 中獲取當前日期和時間方法
3  使用 Date 和 toString 去打印當前日期和時間
4  */
5
6 import java.util.Date;
7
8 public class DateDemo {
9     public static void main(String[] args) {
10         // 初始化 Date 对象
11         Date date = new Date();
12
13         // 使用 toString() 函数显示日期时间
14         System.out.println(date.toString());
15     }
16 }
17
18 // Output:
19 // Mon May 04 09:51:52 CDT 2013
20
21 /**
22  日期比较
23  Java使用以下三种方法来比较两个日期：
24
25  1. 使用 getTime()
26      方法获取两个日期（自1970年1月1日经历的毫秒数值），然后比较它们
27  2. 使用方法 before(), after() 和 equals()
28      例如，一个月的12号比18号早，则 new Date(99, 2,
        12).before(new Date (99, 2, 18)) 返回true。

```

```

28 3. 使用 compareTo() 方法，它是由 Comparable
    接口定义的，Date 类实现了这个接口。

```

```

29 /**
30 使用 SimpleDateFormat 格式化日期
31 SimpleDateFormat
32 是一个以语言环境敏感的方式来格式化和分析日期的类
33 SimpleDateFormat
34 允许你选择任何用户自定义日期时间格式来运行
35 例如：
36 */
37 import java.util.*;
38 import java.text.*;
39
40 public class DateDemo {
41     public static void main(String[] args) {
42
43         Date dNow = new Date( );
44         SimpleDateFormat ft = new SimpleDateFormat
45             ("yyyy-MM-dd hh:mm:ss");
46
47         System.out.println("當前時間為： " +
48             ft.format(dNow));
49     }
50 }
51 // Output:
52 // 當前時間為： 2018-09-06 10:16:34
53 /*
54 實際範例
55 */
56
57 import java.util.Date;
58
59 public class DateDemo {
60
61     public static void main(String[] args) {
62         // 初始化 Date 对象
63         Date date = new Date();
64
65         // c的使用
66         System.out.printf("全部日期和時間：%tc%n",date);
67
68         //f的使用
69         System.out.printf("年-月-日格式：%tF%n",date);
70
71         //d的使用
72         System.out.printf("月/日/年格式：%tD%n",date);
73
74         //r的使用
75         System.out.printf("HH:MM:SS
76             PM格式（12時制）：%tr%n",date);
77
78         //t的使用
79         System.out.printf("HH:MM:SS格式（24時制）：%tT%n",date);
80
81         //R的使用
82         System.out.printf("HH:MM格式（24時制）：%tR",date);
83     }
84 }

```

2.4 Math

```

1 /* 三角函數 */
2 sin(double a)
3 cos(double a)
4 tan(double a)
5 Math.PI
6
7 /*四捨五入*/
8
9 // 無條件向上取整數
10 Math.ceil(2.1) = 3.0

```

```

11 // 無條件向下取整數
12 Math.floor(2.1) = 2.0
13 // 取最接近整數 如果一樣就近往偶數的那個取
14 Math rint(2.5) = 2.0
15 Math rint(-2.5) = -2.0
16
17 /* 取亂數 */
18 Math.random()

```

2.5 pascal triangle

```

1 import java.math.BigInteger;
2 import java.util.Arrays;
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Scanner input = new Scanner(System.in);
9         int n = input.nextInt();
10
11         BigInteger[] triangle = new BigInteger[n+5];
12         Arrays.fill(triangle, BigInteger.ONE);
13
14         for( int i = 0; i <= n; i++ ){
15             for( int j = i; j > 0; j-- ){
16                 triangle[j] =
17                     triangle[j].add(triangle[j-1]);
18             }
19             for( int j = n-i; j > 0; j-- ){
20                 System.out.print(" ");
21             }
22             for( int j = 0; j <= i; j++ ){
23                 System.out.printf("%d", triangle[j]);
24                 if( i % 2 == 0 && i != 0 && j == i ){
25                     break;
26                 }
27             }
28             if( i != 0 ){
29                 System.out.print(" ");
30             }
31             System.out.println();
32         }
33     }
34 }

```

2.6 point2D

```

1 import java.awt.geom.Point2D;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double x1 = 2.0, y1 = 6.0;
7         Point2D pt = new Point2D.Double(x1, y1);
8
9         System.out.println(pt.distance(0,0));
10     }
11 }
12
13 // Output:
14 // 6.324555320336759

```

2.7 split StringTokenizer

```

1 /*
2 當字串中帶有 `+ * / \` 等符號時要寫 `\\`
3 轉義，因為他們在正則表達式中有相應的不同意義
4 最後放 0 的原因 -> limit is 0; array contains all
5 substrings
6 Positive Lookahead or Lookbehind 觀念

```

```

5 | "((?=)|(?<=))" -> 才會把 運算子 還有 運算元 }
   | 都單獨切開
6 | */
7 | String[] array =
   |     str.split("(\\(|\\)|\\.|\\/|\\\\|\\}|\\{|\\%|\\?|\\<|=|+|-|\\*|\\/|\\\\|\\}\\(|%)") , 0);
8 |
9 | /*
10 | 預設 ans 賦值 可控制小數點位數 ex 0.000 ->      2.9 StringB
    | 小數點後三位
11 | */
12 | BigDecimal ans = new BigDecimal("0.000");
13 |
14 | /*
15 | 取運算子的部分用 StringTokenizer 更為方便
16 | */
17 | StringTokenizer st = new
    |         StringTokenizer(str, "0123456789.");
18 | boolean flag = true;
19 | while(st.hasMoreTokens()){
20 |     System.out.print( flag ? st.nextToken() : " " +
        |                     st.nextToken());
21 |     flag = false;
22 | }

```

2.8 str contain

```

1 boolean flag = true;
2 for( String check : array){
3
4     // string.contains 一次查詢多個關鍵字的寫法
5     // 要另外存一個數組 token
6     // 且在查詢時要以此 for ( String check : array )
7     // 去做掃描
8     if( !token.contains(check) ){
9
10        // 將字串型態的數字轉成實數並相加
11        BigDecimal b = new BigDecimal(check);
12
13        // (boolean 判斷式) ? (true的輸出) : (false
14        // 的輸出)
15        System.out.print( flag ? check : " " + check);
16        flag = false;
17
18        // BigDecimal 內建 .add 可以直接相加
19        ans = ans.add(b);
20    }
21 }
22
23 /*
24 找 BC 是 ABCD 的子字串
25 可以用 String.contains 找
26 */
27
28 import java.util.Scanner;
29
30 public class Main {
31
32     public static void main(String[] args) {
33         Scanner input = new Scanner(System.in);
34
35         String str = input.nextLine();
36
37         // 已確認：切割後的空白不會存進 array
38         String[] array = str.split(" ");
39
40         // 原來可以用 str.contains 去找子字串
41         // 我還以為只能找字元..
42         if(array[0].contains(array[1])){
43             System.out.println(array[1] + " is a
44                                 substring of " + array[0]);
45         }
46         else{
47             System.out.println(array[1] + " is not a
48                                 substring of " + array[0]);
49         }
50     }
51 }

```

```

43     }
44
45     }
46 }
//\\)/\\(\\(%))",0);

```

2.9 StringBuilder

```

1 // 先把 i 轉成 String ( 因為這樣比較方便 reverse )
2 String str = String.valueOf(i);
3
4 // 使用 StringBuilder 宣告 rev -> 因為 StringBuilder
   才有 .reverse() 可以用
5 StringBuilder rev = new StringBuilder();
6
7 // 這裡 append (附加) str 的值給 rev
8 rev.append(str);
9
10 // reverse
11 rev.reverse();
12
13 /* 置換 */
14
15 // replace
16 import java.lang.*;
17
18 public class StringBuilderDemo {
19
20     public static void main(String[] args) {
21
22         StringBuilder str = new StringBuilder("Java Util
           Package");
23         System.out.println("string = " + str);
24
25         // 從 index 5 到 9 換掉
26         str.replace(5, 9, "Lang");
27
28         System.out.println("After replacing: " + str);
29     }
30 }
31
32 // string = Java Util Package
33 // After replacing: Java Lang Package

```

2.10 this

```

1  /**
2  你看到在建構式參數與物件資料成員同名時，可用 this 加以區別
3  */
4
5  public class CashCard {
6
7      private String number;
8      private int balance;
9      private int bonus;
10
11     public CashCard(String number, int balance, int
12         bonus) {
13         this.number = number;
14         // 參數 number 指定給這個物件的 number
15
16         this.balance = balance;
17         // 參數 balance 指定給這個物件的 balance
18
19         this.bonus = bonus;
20         // 參數 bonus 指定給這個物件的 bonus
21     }
22     ...
23 }

```

2.11 type changing

```
1  /*
2  變數型態轉換
3  */
4
5  // Integer to String
6  int num;
7  Integer.toString(num);
8  String.valueOf(num);
9
10 // String to Integer
11 String str;
12 Integer.parseInt(str);
13 Integer.valueOf(str);
14
15 // String to Array
16 // but will store like [1,2,3]
17 char[] ch = str.toCharArray();
18
19 // Integer to Double
20 int num;
21 double d1 = double.valueOf(num);
22 double d2 = new double(num);
23
24 // Double to
```